Search and rendezvous by mobile robots in continuous domains

by

Ryan Killick

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Carleton University
Ontario, Canada

©2021
Ryan Killick
Abstract

A collection of $n \geq 1$ mobile robots/agents are located within a continuous domain $\mathcal{D}$. In the search problem the agents are tasked with finding a stationary target located at an unknown position within $\mathcal{D}$. In the rendezvous problem the agents are required to all converge to any single location within $\mathcal{D}$. In either problem the capabilities of the agents are described by a robot model $\mathcal{M}$ (describing, for example, speed constraints, the fault model, knowledge available to the agents, memory capacity), and the cost of a solution algorithm $S$ is described by a cost function $\mathcal{C}$ which assigns a real number $\mathcal{C}(S)$ to each $S$. The ultimate goal of either problem is for the agents to complete the desired task whilst minimizing $\mathcal{C}$ subject to $\mathcal{M}$. This thesis will be of the integrated form and will be composed of several published results concerning the problems of search and rendezvous by mobile agents in continuous domains.
Preface

Publications included for credit


This is the journal version of a paper first appearing in the proceedings of SIROCCO2019. The paper has been reformatted but otherwise has not been modified from its published version.


The paper has been reformatted but otherwise has not been modified from its published version.


This is the journal version of a paper first appearing in the proceedings of FUN2018 under the title “God Save the Queen”. The paper has been reformatted but otherwise has not been modified from its published version.


This is the journal version of a paper first appearing in the proceedings of SIROCCO2018 under the title “Priority Evacuation from a Disk Using Mobile Robots”. The paper has been reformatted but otherwise has not been modified from its published version.


The paper has been reformatted but otherwise has not been modified from its published version.


This is the journal version of a paper first appearing in the proceedings of SIROCCO2018. The paper has been reformatted but otherwise has not been modified from its published version.


The paper has been reformatted but otherwise has not been modified from its published version.
To whom it may concern.

This is to confirm that Ryan Killick has made significant contribution to all the papers listed in his PhD Thesis document. Ryan engaged in all aspects of the research from the mathematical discussions by often devising the primary algorithms and suggesting innovative solutions to the writing of the final paper. His participation was outstanding and helped trace new research directions.

Sincerely,

Evangelos Kranakis
Chancellor’s Professor
Acknowledgements

I thank everybody that helped in the completion of this thesis. In particular, I cannot offer
enough gratitude to my friend and supervisor Dr. Evangelos Kranakis who initially provided
the hook to start me on this line of research. More than this, Evangelos’ love and dedication
to the subject has only been an inspiration to me – as I’m sure it has for many students
preceding me. I am truly privileged to have been able to work with him during the course of
my degree. Many, many thanks Evangelos!

I also give special thanks to Dr. Jurek Czyzowicz and the rest of the “Montreal gang”
(Dr.s Konstantinos Georgiou, Danny Krizanc, Lata Narayanan, Jarda Opatrny, Manuel
Lafond, Sunil Shende, and Denis Pankratov) for all of our enjoyable and productive meetings
at Concordia where much of the research in this thesis was born. I look forward to the end
of the pandemic and to the next time that we can meet in person. My sincerest thanks goes
to my parents and my partner in crime Maria Paula who offer never ending support, and a
sanctuary from the stresses of the world. Finally, I would like to thank my dog Duke who
still possesses the exceptional ability to break my concentration so that I can stretch my legs
with a nice relaxing walk.
**Contents**

1 Introduction .................................................. 16
   1.1 Search problems ........................................... 18
      1.1.1 Linear search with faults .......................... 20
      1.1.2 Search and evacuation in other continuous domains .. 22
   1.2 Rendezvous problems ................................. 23
      1.2.1 Deterministic rendezvous .......................... 25
      1.2.2 Gathering ........................................... 26
      1.2.3 Randomized rendezvous .......................... 27
   1.3 Related work ........................................... 27
   1.4 Outline ................................................. 31

I Search problems ............................................. 33

2 Linear search with faults .................................. 34
   2.1 Introduction ........................................... 35
      2.1.1 Model ............................................. 36
      2.1.2 Preliminaries and notation ......................... 37
      2.1.3 Related work ..................................... 39
      2.1.4 Results and outline ................................ 40
   2.2 Crash faulty evacuation ......................... 41
      2.2.1 Proportional schedules .......................... 41
   2.3 Generalized (proportional) schedules ........... 47
      2.3.1 Many faults ..................................... 48
      2.3.2 Three agents, one fault .......................... 60
   2.4 Byzantine search with one fault ................. 66
   2.5 Conclusions ........................................... 70

3 Time-Energy Tradeoffs for Evacuation by Two Robots in the Wireless Model 73
   3.1 Introduction ........................................... 75
10.1.2 Main Contributions .......................................................... 271
10.1.3 Related work ................................................................. 272
10.2 Model .............................................................................. 274
10.3 Election with knowledge of both \( n \) and \( \ell \) ......................... 275
  10.3.1 Exact knowledge of \( n \) and \( \ell \) ........................................... 275
  10.3.2 Inexact knowledge of \( n \) and/or \( \ell \) ............................... 278
10.4 Election with knowledge of either \( n \) or \( \ell \) ............................. 279
  10.4.1 Exact knowledge of \( n \) or \( \ell \) .......................................... 279
  10.4.2 Inexact knowledge of \( n \) or \( \ell \) ...................................... 282
10.5 Impossibility results ............................................................. 283
10.6 Extensions and Open Questions ........................................... 285
  10.6.1 Discussions and Open Problems ...................................... 286

Appendices
10.A Description of Karp’s theorem .............................................. 289
10.B Pseudocode for algorithms of Section 10.3.1 ....................... 290
10.C Pseudocode for algorithms of Section 10.3.2 ....................... 291
10.D Pseudocode for algorithms of Section 10.4.1 ....................... 291
10.E Pseudocode for algorithms of Section 10.4.2 ....................... 293

11 Conclusions ........................................................................ 294
List of Figures
1.1

The doubling strategy which achieves a 9d search time for the basic LSP. . . 19

2.1

Example space-time trajectories when r = 2 and n = 5, 7, 9. Note that each
r+1
. . . . . . . . . . . .
of the turning-points lie along a cone with slope ± r−1
Illustrating the points (ρi,j,k , τi,j,k ) and their respective cones. Only (ρi,j,1 , τi,j,1 )
are indicated. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
The competitive ratio as a function of the number of faults f for the proportional schedule algorithm. The blue line indicates the optimized competitive
ratio; the red lines indicate the bounds of Theorem 2.2.1; the green line
indicates the competitive ratio of evacuation when we choose r to optimize
√
only Sfx ; the lower black line indicates the asymptotic limit of 4 + 2 2. . .
+
◦
Illustration of the intersection points (ρ◦i,j,k , τi,j,k
) (left side) and (ρ+
i,j,k , τi,j,k )
(right side) of agents i and i + k. In this example j is even. . . . . . . . . .
The optimized trajectories of the agents. . . . . . . . . . . . . . . . . . . .
The optimized competitive ratio of the generalized schedules as a function
of the number of faults. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Illustrating the two worst case scenarios for the generalized proportional
schedule for n = 3 and f = 1 when the parameters a and q are chosen
according to (2.18) and (2.19). Scenario A on the left occurs when the target
is just beyond the turning point di,j and agent i + 1 is faulty. Scenario B on
(1)
the right occurs when the target is just beyond the turning point di+1,j−1 and
agent i is faulty. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Setup for the case that the first announcement claims that the target is at
(1)
location x ∈ (di,j , di+2,j−1 ] and the actual target is on the left side at location
x0 . Agent i is in red, agent i + 1 in blue, and agent i + 2 in green. On the left
side it is agent i + 1 that falsely announces the target and on the right side
it is agent i + 2 that falsely announces the target. In either case the dashed
lines show where the agents would be if the announcement were not made
and the solid lines show the actual trajectories of the agents. . . . . . . . .

2.2
2.3

2.4
2.5
2.6
2.7

2.8

11

41
45

47
53
60
61

62

67


2.9 Setup for the case that the first announcement claims that the target is at location \( x \in (d_{i+2,j-1}, d_{i+1,j}] \) and the actual target is on the left side at location \( x' \). Agent \( i \) is in red, agent \( i+1 \) in blue, and agent \( i+2 \) in green. On the left side it is agent \( i+1 \) that falsely announces the target and on the right side it is agent \( i+2 \) that falsely announces the target. In either case the dashed lines show where the agents would be if the announcement were not made and the solid lines show the actual trajectories of the agents.

3.1 The competitive ratio of \( N_{s,r} \) for the choices of Theorem 3.2.4.

4.1 A representation of position (x-axis, vertical dashed line is 0) and time (y-axis), and the trajectory followed by the two robots (solid lines). The two diagonal dashed lines form the “1/3 cone” of Theorem 4.4.1.

4.2 The robots’ behavior when the exit is found by \( L \) is indicated by the bold line. In the first case (left), the catch-up speed is slower than 1 (and the rendezvous is realized at the turning point of the non-finder), whereas it is 1 in the second case (right).

4.3 The competitive ratio of algorithm \( N_{s,r,k} \) (vertical axis) for the entire spectrum of \( cb \geq 9 \) (horizontal axis). Red curve corresponds to the case \( cb \leq \gamma_1 \), blue curve to the case \( cb \in (\gamma_1, \gamma_2) \) and green curve to the case \( cb \geq \gamma_2 \). The curve is continuous and differentiable for all \( cb \geq 9 \).

4.4 Comparison between the competitive ratio achieved by using optimal speed parameters to \( f_{NLP}^b \) of Theorem 4.3.2 (calculated numerically using software) and the competitive ratio achieved by the choices of Theorem 4.3.4.

4.5 The eigenvalues of matrix (4.3) as a function of \( q \in (0, 1/3] \) (and scaled by \( b^3/(b - s_0)^3 \)).

5.1 An illustration of trajectories \( S(t), Q(t) \), and their critical angles at some fixed time \( \tau \), with \( S(\tau) = S, Q(\tau) = Q, S'(\tau) = u, Q'(\tau) = v \).

5.2 Algorithm \( \text{SEARCH}_1(\alpha, \beta) \) depicted for the optimal parameters of the algorithm. In all subsequent figures, as well as here, the orange points on the perimeter of the disc correspond to the worst adversarial placements of the treasure, which due to our optimality conditions induce the same evacuation cost. The orange points in \( Q \)’s trajectories correspond to the \( Q \)’s positioning when the treasures are reported, in the worst cost induced cases. The green dashed line depicts \( Q \)’s trajectory after \( Q \) abandons her trajectory and moves toward the reported exit following a straight line.
5.3 Algorithm SEARCH$_2(\alpha, \beta)$ depicted for the optimal parameters of the algorithm. .......................................................... 146

5.4 Algorithm SEARCH$_3(\alpha, \beta, \rho)$ depicted for the optimal parameters of the algorithm. .......................................................... 149

5.5 Algorithm SEARCH$_3(\alpha, \beta, \rho)$ depicted for the optimal parameters of the algorithm. .......................................................... 154

5.6 The queen must be in region $R$ at time $f(s_3)$. Here $s_3 = E$ and $q_3 = F$. .......................................................... 157

5.7 Blue trajectory: servant and red trajectory: queen. At point $H$, if the queen hears of an exit at $E$, she goes there, otherwise she goes to $F$. .......................................................... 159

5.8 Setup for the proof of Lemma 5.4.4. The boundary of the disk $D_P$ is indicated in blue. The arc of $U$ which is excluded from $D_P$ is highlighted in red and has length $\theta$. .......................................................... 161

6.1 Evacuation times $T$ of Algorithm 2 for $n \in [4, 2^7]$ (left) and $n \in [4, 2^{18}]$ (right). The upper bound of $2 + 4(\sqrt{2} - 1) \frac{\pi}{n}$ (Theorem 6.3.1), the lower bound of $1 + \frac{2}{n} \cos^{-1} \left( \frac{\pi}{n} \right) + \sqrt{1 - \frac{4}{n^2}}$ (Theorem 6.4.1) are also provided. For comparison, a naive upper bound and lower bound of $2 + \frac{2\pi}{n}$ (see Section 6.2.2) and $2 + \frac{\pi}{n+1}$ (see Section 6.4) are included. .......................................................... 175

6.2 Depiction of the two trivial algorithms each achieving an evacuation time of $2 + \frac{2\pi}{n}$. Both algorithms are in the class $S$ and the algorithm on the right is also in the class $S_{sym}$. The queen is indicated by the blue point and the servants by the red points. A red arc indicates points that have been discovered. .......................................................... 177

6.3 Illustration of the queen’s trajectory $Q(t)$ (blue point) and the motion of the intercepts $A_+(t)$ and $A_-(t)$. The blue circle represents the circle $C_Q(t)$ and the black circle represents the circle $U$. A red arc indicates those positions of $U$ that must be discovered at the indicated time. Time flows from right to left. .......................................................... 179

6.4 Example configuration of Algorithm 2 when $n = 8$. The configuration is only shown for the 4 servants on the upper half of the circle $U$. In this diagram all servants move counter-clockwise. The servants $S_1$ and $S_2$ have already finished their search and are located at the starting positions of the respective servants $S_2$ and $S_3$. The servant $S_3$ is just about to finish its search. The point $A_+(\tau)$ marks the location where the intercept $A_+(t)$ slows to a speed of 1. The servant $S_4$ will reach the point $A_+(\tau)$ at the exact moment the intercept does. .......................................................... 181

6.5 Setup for the proof of Lemma 6.4.6 .......................................................... 186

7.1 Illustration of the idea of containment. On the left $T(x)$ contains $T'(x)$ and on the right $T'(x)$ contains $T(x)$. In both cases the containment point is indicated. .......................................................... 202
7.2 The two worst case possibilities in $D$-Model using Algorithm 3 with $|X_k| = 2^k$. The point of rendezvous is indicated in green and in both cases the robots rendezvous as $R$ is approaching its $j_k$th turning-point and when $R'$ is approaching its $j^\text{th}$ turning-point. Left: The case that $k + j$ is even. Right: The case that $k + j$ is odd. The absolute worst-case is depicted and occurs when $d = 3$ and $v = \frac{1}{10}$ such that the competitive ratio is 9.55. Note that the scales of the depicted trajectories are not the same for the two cases.

8.1 Three rounds of the algorithm.
8.2 Structure of the active phase of round $n$.
8.3 Illustration of how the active phase of $R$ can overlap with the inactive phase of $R'$. The intervals of overlap are shaded, the active/inactive phases of $R$ are indicated by solid lines, and those of $R'$ are indicated by dashed lines.

9.1 Competitive ratio bounds for various regions of the space of possible $n$ and $F$ pairs.
9.2 Inputs for example analysis of competitive ratio. In both cases the robots $A$, $B$, and $C$ move directly towards the center of the minimum enclosing circle of $S = \{A, B, C\}$.
9.3 Setup for the proof of Lemma 9.2.6.
9.4 Setup for the proof of Theorem 9.2.11. The shaded gray region indicates those positions of $A$ such that $a \leq b \leq c$. Top: $\tan \beta \geq \sin \gamma$. Bottom: $\tan \beta \geq \sin \gamma$.
9.5 Setup for the proofs of Theorem 9.3.1 (left) and Theorem 9.3.3 (right).
9.6 Configuration with three robots $A$, $B$, and $C$ that maximizes our competitive ratio.
9.7 Setup for the proof of Theorem 9.3.3.

10.1 Left: The instance $I$ with two robots $r_1$ and $r_2$ on a cycle of length $\ell$. Right: The instance $I'$ with four robots $r_1, r_2, r_1',$ and $r_2'$ on a cycle of length $2\ell$. 
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A summary of the best known upper and lower bounds on the search time for linear search with byzantine faults. Bounds for ( n &gt; 3 ) are from ([70]), ([136]), and ([156]). The upper bound for ((n, f) = (3, 1)) was derived as part of this thesis (see Chapter 2).</td>
<td>21</td>
</tr>
<tr>
<td>2.1</td>
<td>Best known upper and lower bounds on the competitive ratio of evacuation by ( n = 2f + 1 ) agents. The lower bounds all derive from lower bounds on crash faulty search.</td>
<td>40</td>
</tr>
<tr>
<td>2.2</td>
<td>Optimal parameter choices for the generalized schedules and the corresponding competitive ratio. The competitive ratio of the optimized proportional schedules is included for comparison.</td>
<td>61</td>
</tr>
<tr>
<td>5.1</td>
<td>Upper bounds (UB) and lower bounds (LB) for priority evacuation.</td>
<td>138</td>
</tr>
<tr>
<td>6.1</td>
<td>Evacuation times ( T ) of the queen using Algorithm 2 (numerical results). The upper bound of ( 2 + 4(\sqrt{2} - 1) \frac{n}{\pi} ) (Theorem 6.3.1), and the lower bound of ( 1 + \frac{2}{n} \cos^{-1} \left( \frac{2}{n} \right) + \sqrt{1 - \frac{4}{n^2}} ) (Theorem 6.4.1) are also provided. For comparison, the naive upper bound and lower bound of ( 2 + \frac{2n}{n+1} ) (see Section 6.2.2) and ( 2 + \frac{\pi}{n+1} ) (see Section 6.4) are included.</td>
<td>175</td>
</tr>
<tr>
<td>9.1</td>
<td>Summary of competitive ratio bounds for various algorithms.</td>
<td>244</td>
</tr>
<tr>
<td>10.1</td>
<td>Results according to the knowledge of the robots (“Ex.” = exact, “-” = no knowledge, “UB” = upper bound). ( T_{\text{exp}} ) (resp. ( B_{\text{exp}} )) represents the expected time (resp. random-bit) complexity. The column “Type” gives the type of randomized algorithm (LV = Las Vegas, MC = Monte Carlo). The last column gives the corresponding algorithm label in the text. When an upper bound on ( \ell ) (resp. ( n )) is known it is represented by ( L ) (resp. ( N )); and the constructed upper bound on ( n ) is ( N = \frac{Ln}{\ell} ).</td>
<td>271</td>
</tr>
<tr>
<td>10.2</td>
<td>Same as Table 10.1 for time and bit complexities with high probability.</td>
<td>272</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Search and rendezvous problems are a fundamental research topic in theoretical computer science, robotics, and operations research. Search traditionally concerns the problem of a searcher/agent/robot whose goal is to locate a hidden target in as little time as possible. However, a large number of variations of this basic problem have, and can be considered. The search may take place in a continuous environment (line, plane, perimeter of a closed region, etc.) or in a discrete environment (e.g. graph). There may be one searcher or many; the searchers may have different capabilities (e.g. speeds); may communicate with each other in a number of ways (wireless, face-to-face); or may be unreliable in the sense that a subset of the searchers may exhibit errors/faults as they perform the search. The target that is being sought after may itself be mobile/immobile and may be represented by a point, line, or other geometric object. Finally, one may consider goals for the searchers other than, or in addition to, minimizing the search time (e.g. minimizing the total distance travelled by the searchers).

The search problem naturally extends itself to the rendezvous problem whereby the agents are now seeking each other out. In the simplest case two agents are required to meet within a given search domain, however, many of the same variations considered for search can be considered here. In contrast to search, with the rendezvous problem arise many additional considerations that deal with the intrinsic properties of the searchers themselves. Indeed, if the searchers are each equipped with unique IDs or some other distinguishing property then a rendezvous algorithm might assign a different strategy to each searcher. However, if the searchers are indistinguishable then we can only assign a single strategy to all searchers. Thus, the rendezvous problem leads into the realm of distributed computing where a primary goal is to determine the minimum set of capabilities of the searchers in order to complete the rendezvous task.

The thesis will be comprised of a collection of published results concerning three variations of the basic search problem and three variations of the basic rendezvous problem. Informal descriptions of each of the studied problems follow.
**Search and evacuation by faulty robots**  A group of \( n \) unit speed robots are placed at the origin of the infinite line and wish to evacuate from a target placed at unknown initial distance \( d \) from the robots. A robot will only know that it has found the target if it is colocated with it, however, once a robot finds the target it can communicate the location of the target instantaneously to the others. The task is complicated by the fact that a subset of the robots exhibit crash faults and may fail to announce the target when they are on top of it. The identity of the faulty agents is unknown and it can be assumed that these agents will fail in such a way to delay the search for as long as possible. How should the agents move about in order to minimize the competitive ratio of the search – the supremum, over all possible target locations, of the ratio of the time taken to evacuate divided by the initial distance to the target?

**Minimum energy evacuation from the line**  Two robots with maximum speed \( b \) are placed at the origin of the infinite line and are required to reach a target placed at unknown initial distance \( d \) from the robots in time not exceeding \( c \cdot d \) for a given \( c > 0 \). A robot will only know that it has found the target if it is immediately on top of it. Moreover, in order to move at speed \( s \) across distance \( x \) a robot consumes energy \( s^2 x \). The objective is to devise a strategy for the agents to complete the given task using the minimum amount of energy possible.

**Priority evacuation from a disk**  A queen and her \( n \) servants are placed at the origin of a unit disk and can move about with unit speed. An exit is known to be placed on the perimeter of the disk, however, the exact location of the exit is unknown. The exit will only be discovered by the queen/servants if they are colocated with it. The objective is to devise strategies for the queen and her servants in order to minimize the time required for the queen to reach the exit.

**Symmetric rendezvous with asymmetric clocks**  Two agents are placed at arbitrary positions on the infinite plane and wish to rendezvous (i.e. meet). The agents are anonymous (i.e. have no distinguishing identification), cannot communicate, and must execute the same rendezvous strategy. Each robot has its own individual walking speed, compass (which it uses to orient itself and define a handedness), and clock (which it uses to define a unit of time in order to measure its movements). The speeds, compasses, clocks of the agents may or may not be the same, however each agent believes that it travels one unit distance in one unit time and believes that its compass points “north”. The objective is: a) to determine under what conditions is the rendezvous possible, and, b) to construct competitive algorithms to achieve the rendezvous (when it is possible). The same problem is also studied on the infinite line.
Gathering faulty robots in the plane  A collection of \( n \) robots are distributed arbitrarily in the plane. The robots wish to gather (i.e. all reach the same location). The robots are anonymous, however each is equipped with a GPS device allowing them to know their coordinate positions within the plane, and each is able to communicate (broadcast) wirelessly to all others. The robots’ task is complicated by the presence of some number of “spies” in their midst – a subset of malicious robots whose goal is to prolong the gathering for as long as possible. Fortunately, the reliable robots each possess a key such that, once all keys are gathered, they will know precisely which of the robots are the spies. The objective is to gather the set of reliable robots in as little time as possible.

Gathering (and leader election) on the continuous cycle  A collection of \( n \) unit speed robots are placed at arbitrary locations on a unit cycle (i.e. continuous boundary of a unit circle). The robots wish to gather (i.e. all reach the same location). The robots are anonymous, have zero visibility, and can communicate only when they are face-to-face, however, they each possess a fair coin allowing them to make decisions in a randomized way. The objective is to construct a strategy for the robots allowing them to gather: a) in minimum time, and, b) using the fewest number of coin flips.

During the remainder of this introduction chapter each of these problems will be motivated and introduced in the context of the larger body of research that the problems derive from. However, as the extent of the literature on search/rendezvous is quite massive, the discussion will be limited as much as possible to only those details necessary to introduce the six problems mentioned above. A much more detailed consideration of related work will be provided in Section 1.3.

1.1 Search problems

The first search type problem was posed in the mid 1960’s by Bellman [25] and (independently) by Beck [22] and concerned the search for a target placed on an infinite line according to a known probability distribution \( F \). More interesting to this research is the form of the problem proposed by Beck in [23] and later popularized as the Lost Cow Problem by Baeza-Yates in [14]. This problem is summarized below and will be referred to as the basic Linear Search Problem (LSP):

**Basic Linear Search Problem:** An immobile target is placed at initial and unknown distance \( d \) from a unit speed searcher on the infinite line. The searcher will only locate the target if it is directly on top of it. Devise a strategy for the searcher in order to find the target as soon as possible.
This problem has, of course, been solved for some time. A complete solution was first described by Beck in [23] and again later (and seemingly independently) by Baeza-Yates in [14][13]. In either case it was demonstrated that: a) the searcher can always find the target in at most $9d$ time, and, b) no strategy can achieve a search time better than $9d$ (up to lower order terms). A strategy assuring the $9d$ search time is the following “doubling” strategy (also called linear spiral search by Baeza-Yates, see Figure 1.1): move to the right 1 unit and back to the origin, then move left 2 units and back to the origin, then right 4 units and back to the origin, . . .

Figure 1.1: The doubling strategy which achieves a $9d$ search time for the basic LSP.

The main idea of this strategy is that the searcher moves between a set of “turning points” located at distances from the origin which follow a geometric sequence. Informally, the use of a geometric sequence of turning points implies that the searcher never wastes too much time searching since the total distance travelled can only be a constant multiple of the actual distance to the target. The idea of using a strategy that geometrically expands its search area plays a central role in the solution of many search (and indeed rendezvous) problems.

Possibly the most natural extension of the LSP is to consider the effect of additional searchers. However, even in the simplest extension to two searchers there are important distinctions that arise depending on the goals and capabilities of the searchers. In the basic Linear Group Search Problem (LGSP) the searchers need only to receive knowledge of the target’s location (a motivating example might be the search for a missing person). In the basic Linear Evacuation Problem (LEP) one requires that all searchers actually reach the target’s location (a motivating example here might be a group of people trying to evacuate from a building).

Solutions to both of these problems depend strongly on the mode of communication (or lack-there-of) assumed for the searchers. The two most popular communication models (and the two considered in this work) are the wireless and face-to-face (F2F) models. In the wireless communication model the searchers can communicate with each other instantaneously and across any distance. In the F2F model the searchers can only communicate when they are located at the same position at the same time. We summarize these two basic group search problems below:

**Basic Linear Group Search Problems:** An immobile target is placed at initial and unknown distance $d$ from $n$ unit speed searchers located at the origin of the infinite line. Searchers will only locate the target if they are directly on top of it and can communicate either: a) wireless, or, b) face-to-face. Devise strategies for the searchers in order to minimize
the time until: 1) all searchers are aware of the target’s location (LGSP), or, 2) all searchers reach the target’s location (LEP).

In the wireless communication model the solution of the basic LGSP becomes trivial – send (at least) one searcher to the left and (at least) one searcher to the right and have the first searcher that finds the target communicate this to the rest. It is clear that this strategy achieves a best possible search time of $d$ and, moreover, that more than two searchers are not required in order to achieve this search time. Interesting questions arise, however, if one considers the effect of faulty searchers.

### 1.1.1 Linear search with faults

The two most common faults considered in the literature are crash faults – the faulty searcher stops working, i.e. it crashes – and byzantine faults – the faulty searcher actively tries to disrupt the search. The study of linear search with faults was initiated in [77] (crash faults) and [70] (byzantine faults). In [77] the crash fault model assumed that faulty searchers would fail to send messages in the case that they found the target. In [70] the byzantine model assumed that faulty robots may fail to send a message if they found the target or may initiate a message even if they did not find the target. In both cases the goal is to derive upper and lower bounds on the search time as a function of $n$ and $f$ – the number of searchers and faulty searchers respectively.

For the crash model the only non-trivial cases occur when $f + 1 \leq n \leq 2f + 1$. Indeed, when $n = f$ the problem has no solution and when $n \geq 2f + 2$ the optimal solution splits the robots into two groups of at least $f + 1$ robots each and sends each group in opposite directions. When $f + 1 \leq n \leq 2f + 1$ the authors of [77] provide an algorithm with search time $2d\frac{\rho^\rho\rho - \rho + \rho - 1}{(\rho - 1)^{\rho - 1}} + d$ where $\rho = \frac{2(f + 1)}{n}$. This search time was later proven to be optimal in [136], thereby closing the problem of crash faulty search.

Turning now to the byzantine model one can observe that the only non-trivial cases occur when $2f + 1 \leq n \leq 4f + 1$, since, for $n < 2f + 1$ the problem has no solution (the majority of the robots are byzantine faulty), and for $n > 4f + 1$ the problem has the trivial solution of sending two groups of at least $2f + 1$ searchers each in opposite directions. In [70] bounds on the search time are given for various small values of $n$ and $f$ and the authors also provide results (some optimal) for larger values of $n$ as a function of the ratio $\beta = f/n$. A number of these upper bounds were later improved in [156]. In Chapter 2 of this thesis a novel algorithm will be introduced which improves the upper bound for the specific case of $(n, f) = (3, 1)$. The current best known upper and lower bounds on the search time for a subset of $n$ and $f$ pairs are provided in Table 1.1.

Since there is at least one reliable robot in each group the searchers are guaranteed to find the target in best possible time $d$.  

20
Table 1.1: A summary of the best known upper and lower bounds on the search time for linear search with byzantine faults. Bounds for $n > 3$ are from [70], [136], and [156]. The upper bound for $(n, f) = (3, 1)$ was derived as part of this thesis (see Chapter 2).

<table>
<thead>
<tr>
<th>$n, f$</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3, 1$</td>
<td>$5.23d$</td>
<td>$7.44d$</td>
</tr>
<tr>
<td>$4, 1$</td>
<td>$3d$</td>
<td>$3d$</td>
</tr>
<tr>
<td>$5, 1$</td>
<td>$2d$</td>
<td>$2d$</td>
</tr>
<tr>
<td>$6, 1$</td>
<td>$d$</td>
<td>$d$</td>
</tr>
<tr>
<td>$5, 2$</td>
<td>$4.43d$</td>
<td>$9d$</td>
</tr>
<tr>
<td>$6, 2$</td>
<td>$3d$</td>
<td>$3.68d$</td>
</tr>
</tbody>
</table>

For a long time it was unknown whether or not byzantine search for $(n, f) = (3, 1)$ could be completed with a competitive ratio better than 9, however, this question was finally answered in [156] where the authors made use of an algorithm from [70] in order to achieve an upper bound of 8.65. Part of the difficulty for the case that $(n, f) = (3, 1)$ – and more generally when $n = 2f + 1$ – is that it is possible that all agents need to reach the target in order to confirm its location (e.g. when the first $f$ agents that visit the target are faulty and fail to announce it). As a result, one must solve as a subproblem the problem of evacuation by $n$ agents, at most $f$ of which are crash faulty. In Chapter 2 of this thesis a detailed analysis of upper bounds for the crash faulty evacuation problem is presented, thus forming the first step to solving the byzantine search problem for the difficult case of $n = 2f + 1$.

We now turn our attention towards the basic Linear Evacuation Problem with just two searchers. If communication is wireless then the problem was solved in [15] where it was demonstrated that the optimal evacuation time (i.e. the time of arrival of the last searcher to the target) is $3d$. This evacuation time is achieved by the obvious extension to the optimal strategy for the basic Linear Group Search problem, i.e. send two groups of searchers in opposite directions and have them immediately travel to the target once it is found.

In the case of F2F communication the authors of [15] demonstrated the somewhat surprising fact that two searchers cannot evacuate in time less than $9d$. Even more surprising is that this result holds no matter how many extra searchers there are [40]. That is, in the case of evacuation with F2F communication, multiple searchers can do no better than what a single searcher can achieve. The same work [40] demonstrating this surprising result also initiated for the first time the study of the evacuation problem when the robots can travel at different speeds. Among other things, they demonstrate the further interesting observation that, in the F2F model, two searchers – one with maximum speed 1 and one with maximum

Note that it is again the case that more than two searchers are not required to achieve the optimal evacuation time.
 speed 1/3 – can also evacuate in time 9d (the strategy to achieve this essentially has each of the two searchers independently perform the doubling strategy, keeping in mind that one searcher is travelling three times slower than the other). In addition to all of this, it was shown in [15] that there is yet another strategy that achieves 9d for two searchers that can travel up to a maximum unit speed. To achieve this each robot is directed to move in opposite directions at speed 1/3. As soon as one searcher finds the target (which will happen at time 3d) it reverses its direction and moves at full speed until it catches the other searcher (which it will do at time 6d). Finally, both searchers move at full speed back towards the target reaching it at exactly time 9d.

The fact that there are multiple strategies all achieving the same optimal evacuation time of 9d naturally leads one to consider other cost measures (instead of or in addition to evacuation time) in order to define a “best” evacuation strategy. For example, one might want to minimize the total distance travelled by the searchers in addition to the evacuation time. It can easily be shown that, under this new cost measure, the three different 9d evacuation strategies are no longer equivalent. Chapters 5 and 4 correspond to two published works ([60] and [62]) which consider in detail the evacuation of two unit speed searchers under a novel cost measure that is based on the energy consumed during the search. The two Chapters respectively treat the evacuation under the wireless and F2F communication models.

1.1.2 Search and evacuation in other continuous domains

Another natural extension to the basic linear search/evacuation problems is to consider search domains other than the infinite line. For example, simply enlarging the dimension of the space considered leads to the search/evacuation problems in the infinite plane and, in general, infinite k-dimensional space. Even sticking to lower dimensional spaces one can still find non-trivial results by changing the topological properties of the space considered (e.g. bounded vs. unbounded spaces).

Most relevant to the proposed thesis is the problem considered in [57, 72] concerning the evacuation of n robots from a unit disk (i.e. the set of points \( \{(x, y) : x^2 + y^2 \leq 1\} \)). In this problem there are n searchers located at the center of the disk and the goal of the searchers is to evacuate from an exit at an unknown location on the perimeter of the disk in as little time as possible. The searchers can travel up to a maximum unit speed, and they will only locate the exit if they are directly on top of it. In the case of a single searcher the solution is trivial and has a (worst-case) evacuation time of \( 1 + 2\pi \). In the case of multiple searchers we again find large differences in solutions depending on the communication model (wireless vs. F2F) of the searchers. Indeed, in the case of wireless evacuation from a disk, it was demonstrated in [57] that two searchers can evacuate in time at most \( 1 + \frac{2\pi}{3} + \frac{\sqrt{3}}{2} \) and that this time is optimal. For the case of F2F communication the best known evacuation time is 5.6234 [91], however, it is still unknown whether or not this is optimal (the best known lower bound
is $5.255 \allowbreak \begin{equation} [72] \end{equation}$). When there are more than two searchers, optimal strategies are unknown for both the F2F and wireless models, although in the case of wireless communication, only asymptotically optimal bounds are known $[57]$.

In this thesis a novel variant of this evacuation problem is considered wherein one of the searchers is given priority over the rest. The problem is best described and motivated by the following short story. A queen is asleep at the center of a circular room surrounded by $n$ of her most loyal servants. Suddenly they are awakened by cries of “Fire! Fire!”.” The room is pitch black and they are disoriented, however both the queen and her servants know there is exactly one exit located somewhere along the perimeter of the room. How are the queen and her servants to find and evacuate the queen from the exit in as little time possible in order to save the queen’s life?

More formally, the Priority Evacuation Problem is defined as follows:

**Priority Evacuation Problem**  
$n + 1$ maximum unit speed searchers are located at the center of a unit disk and there is an exit placed at an unknown location on the perimeter of the disk. The searchers can communicate wirelessly and will only locate the exit if they are immediately on top of it. One of the searchers is distinguished from the rest. The goal is to devise a strategy for the $n + 1$ searchers in order to evacuate the distinguished searcher in as little time as possible.

Chapters 5 and 6 of this thesis will consider this priority evacuation problem in detail.

At this point we have introduced all three of the search problems that will be considered in the thesis and we will now turn our attention to the problem of rendezvous. Further related work concerning search will be discussed in Section 1.3

### 1.2 Rendezvous problems

Rendezvous concerns the task faced by two or more agents that must meet each other within some environment. In many ways this problem resembles the search problem – solution algorithms must describe trajectories for the agents and the objective of minimizing the time at which the meeting occurs is also shared by both problems. Moreover, the early development of the rendezvous problem very much reflected that of the search problem. The first formal treatment (in a continuous environment) was conducted by Alpern in $[7]$ who considered the problem faced by two agents randomly placed on the infinite line according to a known probability distribution function $F$. Following this many of the same variations considered for the search problem were considered for rendezvous and much of this early research on the problem culminated in the full length text $[8]$ treating in great detail both of the search and rendezvous problems.
Despite these many similarities, there is one major difference that sets the search problem distinctly apart from the rendezvous problem. To illustrate this consider the following basic Linear Rendezvous Problem (LRP):

**The Basic Linear Rendezvous Problem:** Two unit speed agents are placed at unknown positions on the infinite line. The initial separation of the agents is $d$, and the agents will only find each other if they are located directly on top of one another. Devise a strategy for the agents such that they meet in as little time as possible.

Now consider solutions to this problem under the two following scenarios: a) different strategies may be assigned to the two agents, and, b) the same strategy must be assigned to each agent. The first scenario reflects the situation where the agents can coordinate with each other prior to the rendezvous taking place (e.g. the agents may intentionally separate and agree before hand on how they should meet up again). The second scenario reflects the situation where the agents must rendezvous without any prior coordination (e.g. the agents are unintentionally separated and thus cannot agree on how they should meet).

It is relatively simple to devise a strategy for the agents that guarantees a meeting in the first scenario. In contrast, it is not even clear that a meeting between the agents is possible in the second scenario (and indeed rendezvous is not possible unless further assumptions are made).

Alpern called the first scenario where the agents may use different strategies the (player) asymmetric rendezvous problem, and the second scenario the (player) symmetric rendezvous problem. Of the two versions of the rendezvous problem, the symmetric version is arguably the more difficult to solve. Indeed, it is often the case that the objective when tackling a symmetric rendezvous problem is to simply demonstrate whether or not the rendezvous is feasible. This compares to the asymmetric version where optimization of the rendezvous strategies is most often the goal. The proposed thesis considers variations only of the symmetric rendezvous problem and thus we will focus our discussion on this case. Thus, in the sequel, when we refer to the rendezvous problem we will be referring only to its symmetric version.

Consider again the basic LRP when the agents must use the same strategy. If there are no further assumptions made on the capabilities/properties of the agents then it is easy to see that the agents will forever remain locked in the same relative positions with respect to one another (and this is independent of the strategy employed). In order to solve the problem there must exist some inherent amount of asymmetry in its description. In other words, there must be a mechanism available to the agents in order to break symmetry in the problem. This concept of symmetry breaking is central to the question of the feasibility of rendezvous and is what sets the rendezvous problem apart from the search problem.

\(^3\)The following strategy that Alpern called “Wait For Mommy” will suffice. One agent (the Child) is instructed to stay still, and the other agent (the Mommy) performs the optimal doubling search strategy.
There are, broadly speaking, two separate ways symmetry may be broken in order to achieve rendezvous. The first and most obvious method of symmetry breaking is to allow the use of randomized algorithms. Indeed, this was the solution first employed in the study of rendezvous [7]. Alternatively, one can make use of asymmetries that are intrinsic to the problem itself in order to break symmetry. We will discuss these deterministic methods of breaking symmetry first and return to randomized methods shortly.

1.2.1 Deterministic rendezvous

If the agents are forced to make deterministic decisions then, perhaps, the most obvious way to break symmetry is to simply assume that the agents are distinguishable. This assumption can be implemented, for example, by providing the agents with unique labels/identifiers (IDs). This was the approach taken in [81, 89]. Alternatively, the agents may have access to a GPS device allowing them to know their coordinate location in the rendezvous environment. This clearly permits rendezvous since both agents can simply move to, say, the origin (although this is certainly not the most efficient strategy [50]). In any of these cases, the idea is that the agents adopt the same parameterized algorithm and use different parameters (IDs, locations) as inputs to this algorithm. A major drawback of this type of approach is that it necessitates the assumption that the values of the parameters which allow the symmetry breaking to occur are known to the agents.

A second method to achieve deterministic symmetry breaking is by providing extra tools/capabilities to the agents. For example, if the agents are able to “see” each other then rendezvous is clearly achievable. Alternatively, each agent may be equipped with a light that is visible by the other and whose color can be controlled by the agent [158], or the agents may be equipped with a token that they may drop in order to mark certain positions within the environment [19].

If the agents do not have any extra capabilities and are not provided with knowledge of symmetry breaking parameters then it is not clear that a rendezvous can take place. This thesis considers this problem and proposes a novel method by which symmetry may be broken even in this restrictive setting in nearly all cases where there is some minimum difference between the agents. This difference may take the form of differences in the agents’: speeds, clocks (i.e. how the agents measure time/distance), and/or compasses (i.e. how the agents orient themselves within their environment). In particular, we derive the conditions on the speeds/clocks/compasses of the agents in order for rendezvous to occur and also provide algorithms to achieve the rendezvous when it is possible. This problem is treated in Chapters 7 and 8 of the thesis.
1.2.2 Gathering

So far we have focused on rendezvous by only two agents. As in the case of search it is only natural to extend this problem with the addition of extra “rendezvouers”. The multi-agent rendezvous problem – or the Gathering Problem as it is more often presently referred to – was considered first in [157] where the problem was viewed from the more general perspective of mobile agent pattern formation problems. Most research in this setting studies the gathering problem under the traditional look-compute-move computation model of distributed computing. This model assumes that each robot/agent is forever cycling between the three states look, compute, and move – in the look state the robot takes a snapshot of its surroundings; in the compute state the robot makes use of the snapshot obtained in the look state and based on this (and possibly additional information if the robots have memory) computes a destination to move to; finally, in the move state the robot moves to the computed destination. This model of computation assumes that time proceeds in discrete rounds and solutions under this model depend strongly on the amount of synchronicity between the agents (i.e. do rounds last the same amount of time for each agent? is each agent active in each round? etc.).

More relevant to the proposed research is the problem of gathering as viewed from the perspective of continuous swarm robotics. In this model of computation robots can move continuously throughout their environment, and time flows continuously and uniformly for all agents. Pioneering work in this area was carried out in [116] where the problem of gathering so called ants (for a(ge)nts) was considered. In this context an ant is an extremely basic computational entity that lacks any kind of distinguishing feature (it is anonymous), has no communication ability, lacks any kind of memory (it is oblivious), and has a limited sensing range (it is myopic). Regardless of the details of the specific model used, the focus in swarm robotics is to construct protocols to be used by large numbers of agents with extremely weak capabilities. Indeed, much of this research has in mind the goal of applying the constructed protocols to real life settings involving large numbers of mass produced and inexpensive robots. In such a system it can be expected that a number of the robots will exhibit faults, and thus the question of fault tolerance of such systems will be an important consideration.

In this thesis the problem of gathering a number of agents in the presence of faulty agents is considered using a model that is a close variant of the continuous time model of swarm robotics. In particular, the problem of gathering $n$ agents in the plane when at most $f$ of the agents exhibit byzantine faults (i.e. the faulty agents can be expected to act in such a way so as to prolong the gathering for as long as possible) is considered when the agents are equipped with a GPS device allowing them to know their coordinate locations within the plane. Further details of this problem and the model used will be presented in Chapter 9.
1.2.3 Randomized rendezvous

If agents are allowed to make randomized decisions then symmetry may be broken, for example, by considering the sequence of outcomes of the toss of a fair coin. In many ways this method of breaking symmetry is similar to the method of using parametrized algorithms described earlier. The difference now is that the input “parameters” of the agents will be randomly generated during the rendezvous algorithm.

An important difference between randomized and deterministic algorithms is that one does not always require correctness guarantees in the case of randomized algorithms. Indeed, sometimes the only “solution” algorithms that exist for a given problem only guarantee a minimum probability of correctness. These algorithms that “gamble with correctness” are often referred to be of the Monte Carlo type. In contrast, algorithms of the Las Vegas type guarantee correctness, however, they can only guarantee probabilistic bounds on the resources used by the algorithms. In other words, Las Vegas algorithms “gamble with resources”.

In this thesis a variation of the gathering problem is considered and analyzed under the assumption that the agents can generate random bits. In particular, the problem of gathering $n$ identical agents on a cycle (i.e. continuous perimeter of a circle) of length $\ell$ is studied and both Monte Carlo and Las Vegas solution algorithms are proposed under a variety of assumptions on the knowledge available to the agents. Further details on this problem are provided in Chapter 10.

1.3 Related work

There is a large body of work on the problems of search and rendezvous. In the case of search the earliest relevant work was that of Beck [22, 21, 23, 24] concerning search on the infinite line, however, also relevant was the work of Heath and Fristedt [106], Fristedt [105], and Gal [109]. These works viewed for the first time the search problem through the lens of game theory. In particular, the search was treated as a zero sum game between a searcher and a hider. The searcher wishes to minimize the time until the target is found, and the hider wishes to hide the target in order to extend the search time as much as possible. Further research in this direction ensued, eventually resulting in a full length text by Gal [108] and by Alpern and Gal in [8].

More recently, the search problem went through a sort of revival in the computer science community with the work of Baeza-Yates et. al. [14] who introduced the “Lost Cow Problem” (or the “cow-path” problem as it is more often referred to today). The most basic version of this problem has already been discussed, however, in [14, 13] the more general $w$-path lost cow problem was studied wherein the lost cow is initially placed at a crossroads with $w$ different paths leading away from it (i.e. $w$ concurrent rays), and only one of these paths contains the target being sought after. The basic LSP introduced in Section 1.1 can be
viewed as the special case of this problem when \( w = 2 \). Many additional relevant results by Baeza-Yates et. al. were presented in [14, 13, 15], including, but not limited to: search for a line in the infinite plane; the effect on the search time when the searcher is provided some amount of knowledge of the target; search in a lattice; search with multiple (two) searchers.

The cow-path problem, and variations of it, have been considered in many works following that of Baeza-Yates et. al. In [128] the authors consider randomized algorithms for the cow-path problem and devise an algorithm that is optimal for any number of paths. In the case of two paths (i.e. the basic LSP), the optimal search time is 4.5911d (which seems to replicate a much earlier result by Beck [23] who also derived a randomized algorithm with search time 4.59d). In [127] the authors consider a two-dimensional version of the cow-path problem wherein the objective is to seek out a horizontal/vertical line in the two dimensional plane. Linear search with terrain-dependent speeds was considered in [78], and in [32] the authors consider the LSP when upper and lower bounds are known on the distance to the target.

The evacuation problem first appeared as the problem of evacuating a group of searchers from a disk [57]. Disk evacuation under the F2F model of communication was considered in [36, 57, 72, 137] and the wireless model was considered in [55, 57, 144]. Evacuation in geometric domains other than the disk was considered in [79] where the problem was studied for squares and equilateral triangles. The priority evacuation problem that is presented in this work has never been studied previously (for a disk nor any other environment).

The problem of group search and evacuation on the line was initiated in [40], where, among other things, the authors demonstrate the surprising result that the optimal evacuation time of any number of searchers is 9d – the same as one searcher. They also initiate for the first time the study of group search by different speed searchers. In [17] the authors continue the study of group search on the line of two distinct speed robots and derive optimal search strategies in both the wireless and F2F models for the full spectrum of speeds for the two searchers. Linear group search with faulty searchers was initiated in [70, 77] and has already been discussed in detail in Section 1.1.1. Evacuation from a disk by possibly faulty searchers was considered in [59].

The vast majority of the work mentioned thus far has concerned itself with the minimization of the search/evacuation time. Research concerning search/evacuation with alternate/additional cost functions is minimal. In [84] the LSP is considered with turn costs (i.e. an additional cost incurred when the searcher changes direction). In [106] the authors considered the basic LSP with various generalized cost functions, and in [15] there was a brief mention of using the total distance travelled by the searchers as an alternate cost function. The problem of minimizing the energy consumption of two person linear evacuation has never been studied before.

The rendezvous problem is a younger problem than search, however it has been around long enough to yield a large number of theoretically interesting results. It was first introduced informally in 1976 by Steve Alpern in [5] who later also formulated and formalized the
continuous time version of the problem in [7]. A further impetus to the problem was given by the seminal book treatment [8] where rendezvous was viewed as a search-game between two players having the converging goal in that they are aiming to find one another as quickly as possible.

Numerous papers on rendezvous followed, covering several cases depending on various parameters of the model: type of environment (graph or geometric), robot’s knowledge about the environment (partial or complete), anonymity of the robot (labeled or not), robot movement mode (synchronous, semi-synchronous or asynchronous), algorithm type (deterministic or randomized), reliability issues related to robot instruments, etc. A survey covering deterministic rendezvous algorithms is presented in [145] while the monograph [133] is dedicated to the ring and torus.

Many papers on rendezvous adopt the discrete model, e.g., where the robots may meet only at graph nodes (e.g., [152]). In the continuous model for graphs, it is possible to consider the agents’ meeting in the interior of graph edges (e.g., [89]). However, when the continuous environment is a two-dimensional plane it is necessary to equip robots with devices permitting non-zero visibility, i.e. the rendezvous arises when the robots belong to each other’s visibility range (e.g., see [10]).

Extending the rendezvous to more than two agents results in what is more commonly known as the gathering problem. This problem was originally introduced in [157] as a version of pattern formation (see also [82]). In distributed computing, the problem of gathering identical robots has been the focus of intensive investigations under a variety of assumptions on the computational power and communication capabilities of the robots (e.g., [44, 47, 101, 139]).

The fundamental issue of the rendezvous problem is symmetry breaking, for example by exploiting some specific parameter(s) of the model that permit robots to act differently and not to be trapped in the same relative position to one another (cf. [145]). If there is no such parameter available it may be shown that the rendezvous is infeasible (e.g., [87]). For rendezvous in graph environments, the symmetry may be broken using asymmetry in the graph topology or the robots’ positions within it (see [76]). In the case of rendezvous in the two-dimensional plane this is not possible and a symmetry-breaking procedure may exploit, for example, the difference of the robots’ labels [81, 89] or robot’s knowledge of its own position in the Cartesian plane [50]. In all these cases the designer of the rendezvous algorithm needs to know what is this parameter of the studied scenario that permits the symmetry breaking. In the work presented in Chapter 8 it is shown that the knowledge to which of the parameters of the studied scenario makes the rendezvous possible is not necessary.

The rendezvous/gathering problems have been studied previously for robots with different speeds [33, 95], inconsistent compasses [46, 125] and chirality or sense of direction [18, 34]. In [124] the authors study the feasibility of gathering by mobile robots that have \( \phi \)-absolute error dynamic compasses, which allows the angle difference between a local
coordinate system and the global coordinate system to vary with time in the range of $[0, \phi]$. In [130] a gathering problem is discussed for robots equipped with inaccurate (incorrect) compasses which may point a different direction from other robots’ compasses. However, in the studies previously mentioned, these differences were obstacles that needed to be circumvented by the suggested algorithms, rather than used for the benefit of the proposed approach, which is the case of the work presented in Chapter 8.

Similarly to rendezvous, the problems of gathering and/or electing a leader and their relationship to asymmetry have been observed, investigated, and discussed when studying solvability of a variety of problems by autonomous mobile robots, in particular pattern formations (e.g., [90, 100, 107]). Indeed, a great deal of research has been devoted to the link between degree of symmetries and deterministic problem solving; see [97] and chapters therein for a recent account, in particular [159]. Almost all of this work is on deterministic solutions, with few exceptions (e.g., [115]).

Robots operating specifically in a continuous cycle have been studied in the context of rendezvous and gathering, but only with robots having different motorial capabilities [95, 121]. Other investigated problems in a continuous cycle are: patrolling, studied both when the robots are identical and when they have different motorial capabilities (e.g. see [58, 69, 80]); and scattering, where the robots must place themselves at uniform distance on the cycle [99].

The geometric continuous settings in which the mobile entities can move freely are in general more suitable than discrete settings for distributed computing applications in robotics [39]. This is further enforced by the fact that after a system shut-down in a robot application the participating robots cannot be guaranteed to occupy the vertices of a graph but rather might be placed at arbitrary locations in the underlying geometric domain. Nevertheless, settings of identical mobile entities operating in discrete spaces (i.e., in graphs) are also extremely important as they naturally describe a wide variety of computational environments, including networked systems supporting mobile software agents, and ad-hoc wireless networks. In these settings, the analogue of a set of mobile robots in a continuous cycle is a set of identical mobile agents in a ring of identical nodes. Interestingly, this discrete setting has been extensively studied, especially for rendezvous and gathering; e.g., see the monograph [133]. In absence of distinct features of the agents and of the nodes (e.g., ids, markers, tokens), solutions are necessarily randomized, and their development has been the object of several investigations. In particular Ooshita et al. studied the gathering problem in anonymous unidirectional ring networks for multiple (mobile) agents with limited knowledge and characterized the relation between probabilistic solvability and termination detection [142]. Izumi et al. investigated the feasibility of polynomial-expected-round randomized gathering for $n$ robots and show that any randomized algorithm has $\Omega(\exp(n))$ expected-round lower bound [123].

In the computational universe of static (or stationary) entities connected via a communication network (i.e. the traditional message-passing universe in distributed computing), the
computational entities coincide with the network nodes (i.e., the nodes are the active agents). Note that, in this universe, the gathering problem does not exist; on the other hand, leader election is a fundamental problem. When the entities are identical, the system is known as an anonymous network, and several researchers have focused on computing in an anonymous ring (e.g., [11, 12, 96]). The problem of electing a leader in an anonymous network, known also as symmetry breaking and for which clearly only probabilistic solutions exist, has been investigated in an anonymous ring network (e.g., [16, 104, 122]). In particular, Itai and Rodeh proposed probabilistic algorithms for both the synchronous and asynchronous case; they considered both cases when the size of the ring may be either known or unknown to the nodes and studied its impact on termination with a nonzero probability [122].

Interestingly, of all the related work, the one closest in spirit to the study of gathering/election presented in Chapter 9 is that of symmetry breaking in an anonymous ring, in spite of the fact that the computational universes are completely different: static entities and discrete space in one while mobile entities and continuous space in ours.

Fault tolerance in mobile agent algorithms has already been discussed in the context of the search problem. Faults or imperfections in mobile agents performing gathering were investigated in [1, 46, 88, 125, 153]. Research in [46, 125] and [153] considered the gathering problem in the presence of inaccurate or faulty robot perception components. In [1] the initial positions of the collection is known to all robots, which operate in so called look-compute-move cycle. The feasibility of the problem, as a function of faulty robots, is investigated in [1] for crash and byzantine faults. In [88], the gathering problem is studied in an unknown graph environment and the feasibility question for byzantine faults in the strong and weak sense are investigated. The results of [88] depend on the knowledge of the upper bound on the size of the graph environment (or the absence of such knowledge).

Most relevant to the problem of gathering faulty robots presented in Chapter 9 is the work [42] where the authors studied the same problem except on a line. We note, however, that the robot movements along the line are much easier to analyze than the setting studied in this work. Indeed, in the case of a line, the robots move inside a corridor forcing them to meet.

1.4 Outline

The remainder of the thesis will be comprised of the author’s published work concerning the search and rendezvous problems previously introduced. Chapters 2 through 6 concern search-like problems, and Chapters 7 through 10 concern rendezvous-like problems. Chapter 2 details the problem of search and evacuation by \( n = 2f + 1 \) agents when at most \( f \) are faulty; Chapters 3 and 4 concern the problem of evacuation on the line by two searchers looking to minimize their energy consumption in the wireless and F2F communication models; Chapters 5 and 6 consider the problem of priority evacuation from a disk. Chapters 7 and
analyze the feasibility of rendezvous by two agents on the infinite line and plane when those agents have asymmetric clocks/speeds/orientations; Chapter 9 presents an asymmetric gathering problem of a group of agents in the plane when \( f \) of them could be faulty; finally, Chapter 10 investigates a randomized problem of gathering agents on the perimeter of a circle.
Part I

Search problems
Chapter 2

Linear search with faults

This chapter considers the problem of evacuation on the line by \( n = 2f + 1 \) agents when \( f \) of them are crash faulty (i.e. they may not announce the exit when they find it). It is the only chapter of the thesis that has not yet been published.
Search and evacuation with a near majority of faulty agents

Jurek Czyzowicz¹, Ryan Killick², Evangelos Kranakis², Grzegorz Stachowiak³

¹Université du Québec en Outaouais, Gatineau, Québec, Canada
²School of Computer Science, Carleton University, Ottawa, Ontario, Canada
³Institute of Computer Science, University of Wroclaw, Wroclaw, Poland.

Abstract

There are \( n \geq 3 \) unit speed mobile agents placed at the origin of the infinite line. In as little time as possible, the agents must find and evacuate from an exit placed at an initially unknown location on the line. The agents can communicate in the wireless mode in order to facilitate the evacuation (i.e. by announcing the target’s location when it is found). However, among the agents are a subset of at most \( f \) crash faulty agents who may fail to announce the target when they visit its location.

In this paper we study this aforementioned problem for the specific case that \( n = 2f + 1 \). We introduce a novel type of search algorithm and analyze its competitive ratio – the supremum, over all possible target locations, of the ratio of the time the agents take to evacuate divided by the initial distance between the agents and the target. In particular, we demonstrate that the competitive ratio of evacuation is at most \( \frac{7}{437011} \) for \((3, 1)\); at most \( \frac{7}{253767} \) for \((5, 2)\) and \((7, 3)\); and at most \( \frac{7}{147026} \) for \((9, 4)\). For larger values of \( n = 2f + 1 \) we prove an asymptotic upper bound of \( 4 + 2\sqrt{2} \). We also adapt our evacuation algorithm for \((3, 1)\) to the problem of search by three agents with one byzantine fault, i.e. the faulty agent may also lie about finding the target. In doing so we improve the best known upper bound on this search problem from 8.653055 to 7.437011.

2.1 Introduction

Problems of search and exploration are central to many areas of computer science and mathematics and, accordingly, have received much attention in the literature. Perhaps the simplest search type problem considers the optimal trajectory of a single mobile agent tasked with finding a target placed at an unknown location on the infinite line. The goal of the agent is to minimize the competitive ratio – the supremum over all possible target locations of the
ratio of the time the agent takes to find the target and the initial distance between the agent and target. Independently studied by Bellman and Beck in the 1960’s, it is now well known that the optimal trajectory for this single agent search uses a doubling strategy whereby the agent, starting at the origin, moves between points on the line at alternating positions $1, -2, 4, -8, \ldots$. It is a simple task to show that this trajectory ensures a competitive ratio of 9.

Search by multiple agents on the line is a natural extension of the single agent search. Of course, with more than one agent also arise questions about how search is affected by the presence of agents with differing capabilities/attributes. For example, one can consider agents with different speeds, and or communication abilities. A particularly interesting and important topic in group search is the development of fault tolerant search algorithms.

In this paper we study a version of fault tolerant group search on the line. Specifically, we consider the problem of evacuation by $n = 2f + 1$ mobile agents when at most $f$ of these agents are faulty. The agents all begin the search at the same time from a common location and the goal is for the agents to find and exit from a target placed at an unknown location on the line. To achieve this goal the agents can co-operate by exchanging messages with one another in wireless mode (i.e. instantaneously and across any distance). This goal is impeded by the presence of $f$ crash faulty agents who may fail to announce that they have found the target when they detect it.

We also use the results from the crash evacuation problem to improve upper bounds on the problem of search by three agents at most one of which is byzantine faulty. A byzantine faulty agent is similar to a crash faulty agent except that byzantine agents can also lie about finding the target. When $n = 2f + 1$ it can happen that all agents are required to reach the target in order for the search to complete and so crash evacuation can be viewed as a sub-problem of the more difficult byzantine search problem.

2.1.1 Model

We have $n = 2f + 1$ mobile agents with at most $f$ of them faulty. Robots/agents all begin at a common location referred to as the origin. The agents can move up to a maximum unit speed in either the positive direction (referred to as moving to the right) or the negative direction (referred to as moving to the left) and an agent may change its travel direction arbitrarily often. There is no time cost associated with an agent changing its movement direction.

The agents are labelled with unique identifiers taken from the set $\{0, \ldots, n - 1\}$ and can communicate with each other in the wireless mode. A parallel search algorithm specifies a unique trajectory for each agent and all agents are assumed to have full knowledge of these trajectories. Since agents know of all other trajectories it follows that the only kind of message broadcast by an agent will be a notification that it has detected the target at its current location. If an agent does not broadcast a message while visiting a location then it is
assumed that the agent did not detect the target at that location.

Each agent is aware of the number $f$ of faults, however, the identity of the faulty agents is unknown. The fault model considered for the agents is that of crash or silent faults. In this model an agent may fail to announce the target when it is detected, however it cannot send a message falsely claiming that it has found the target when it has not (this is known as the byzantine model). The presence of faulty agents thus implies that an agent cannot necessarily trust that the target is not at a location previously visited by another agent. In order to be sure that a target is not at a particular location $x$, it will be required that the location $x$ has been visited by at least one provably reliable agent. With at most $f$ faults, at least $f + 1$ agents must visit $x$ in order to have this guarantee.

It is possible that the faulty agents do not follow the trajectories assigned to them. However, as all agents are aware of the trajectories of the other agents, any agent that is found to be not following its assigned trajectory can be reliably identified as faulty. We will assume that the identity and behaviour of the faulty agents is controlled by an adversary who will always act in a way to maximize the competitive ratio. We may therefore safely assume that any such premature identification will not occur. Each agent will therefore follow the trajectory initially assigned to it until they either find the target or they receive an announcement that the target has been found. Since announcements can always be trusted, agents will immediately move to the announced location in order to complete the evacuation.

2.1.2 Preliminaries and notation

We begin with some definitions.

**Definition 2.1.1.** The evacuation time $E^f_x$ of a parallel search algorithm for $n = 2f + 1$ agents, at most $f$ of which are faulty, is the worst case time required until the last reliable agent reaches a target at location $x$.

**Definition 2.1.2.** The search time $S^f_x$ of a parallel search algorithm for $n = 2f + 1$ agents, at most $f$ of which are faulty, is the worst case time required for the first reliable agent to reach $x$.

**Definition 2.1.3.** The competitive ratio is defined as $R_f = \sup_x \frac{E^f_x}{|x|}$ and represents the worst case ratio of the evacuation time to the lower bound $|x|$ on the time required to find the target.

Since crash faulty agents fail silently (i.e. they cannot lie about finding the target), it follows that any announcement made by an agent must be truthful. As a result the only sensible thing for the (reliable) agents to do once an announcement has been made is to immediately move to the announced target’s location. This observation has two important implications. First, it implies that we can define a parallel evacuation algorithm entirely by the trajectories of the agents. Second, it implies that we can express the evacuation time as
the sum of \( S_f^x \) and the distance between the target and the agent most distant from the target at the time \( S_f^x \). This last point leads us to make the following definition.

**Definition 2.1.4.** Given a parallel search algorithm for \( n = 2f + 1 \) agents, at most \( f \) of which are faulty, define \( i_f^x \) and \( \Delta_f^x \) as the identity of, and distance between, the agent most distant from location \( x \) at the time \( S_f^x \).

With this definition we can express the evacuation time as follows

\[
E = S_f^x + \Delta_f^x.
\] (2.1)

We consider agent trajectories defined by sets of turning points – points on the line at which agents change their movement direction, and between which the agents move at constant unit speed. We use the notation \( d_{i,j} \) to refer to the turning point \( j \) of agent \( i \). We will assume that the turning points alternate on either side of the origin with increasing absolute values, i.e. if \( d_{i,j} > 0 \) then \( d_{i,j+1} < 0 \) and \( |d_{i,j+1}| > |d_{i,j}| \).

For these types of trajectories one must make additional assumptions in order to achieve a constant competitive ratio. To see why this is, imagine we have a set of trajectories with first turning points \( d_{i,0} \) and assume that for the majority of the agents we have \( d_{i,0} > \delta \) for some \( \delta > 0 \). Then the target can be placed at location \( -\epsilon \) with \( \epsilon > 0 \) arbitrarily small and all agents that initially moved to the left are made to be faulty. The first time a reliable agent can reach the target is then \( 2\delta \) and the competitive ratio is at least \( 2\delta/\epsilon \).

To overcome this problem one usually makes the assumption that the agents are aware of a lower bound on the distance to the target. Then, by making the first turning points much smaller than this lower bound, a finite competitive ratio is possible. Alternatively, one can assume that the agents do not have first turning points. In other words, one assumes that the turning point sequence \( d_{i,j} \) extends to \( j = -\infty \) and the agents have always been moving. Although less realistic, we find the latter assumption to be more elegant mathematically and we will take this approach here.

We end this section with a lemma which specifies how the target will be placed in the worst case.

**Lemma 2.1.5.** The supremum of \( E_f^x/|x| \) always occurs when \( x \) is a turning point.

**Proof.** Consider an evacuation algorithm defined by turning point trajectories. Suppose that the competitive ratio of this algorithm is \( \alpha \) and consider a location \( x \) that defines this ratio. We will make use of the work of [136] which states that we have \( \alpha > 3 \).

Without loss of generality assume that \( x > 0 \), and, for the sake of deriving a contradiction, suppose that \( x \) is not a turning point of one of the agents. Since \( x \) is not at a turning point there exists \( u > 0 \) such that the interval \( [x - u, x] \) does not contain any turning points.

Let \( S \) represent the first time the target at \( x \) is announced; let \( i_x \) represent the identity of the agent most distant from \( x \) at time \( S \); and let \( \Delta \) represent the distance between agent
\( i_s \) and \( x \) at the time \( S \). Similarly define \( S', i'_s, \) and \( \Delta' \) for the situation that the target is at \( x' = x - u \) instead of \( x \). By assumption we have
\[
\alpha = \frac{S + \Delta}{x} > \frac{S' + \Delta'}{x - u}.
\]
Consider how \( \Delta \) and \( S \) would change if the target were instead placed at \( x' = x - u \). The agent that announces the location \( x \) must have been moving in the positive direction when it reached \( x \). Since agents move at constant unit speed between turning points it is clear that \( S' = S - u \). For \( \Delta' \) there are a few cases to consider. If we have \( i_s = i'_s \) and agent \( i_s \) was moving towards \( x \) at time \( S \) then it is clear that \( \Delta' = \Delta \). If \( i_s = i'_s \) and agent \( i_s \) is moving away from \( x \) at time \( S \) then \( \Delta' = \Delta - 2u \). If \( i_s \neq i'_s \) agents \( i_s \) and \( i'_s \) crossed paths during the time interval \([S', S]\). In any case, it is simple to see that we will have \( \Delta - 2u \leq \Delta' \leq \Delta \). We can thus conclude that
\[
\frac{S' + \Delta'}{x - u} \geq \frac{S + \Delta - 3u}{x - u} = \frac{\alpha x - 3u}{x - u} = \frac{\alpha(x - u) + (\alpha - 3)u}{x - u} = \alpha + \frac{(\alpha - 3)}{x - u} > \alpha
\]
where the last step results from the fact that \( \alpha > 3 \). We have therefore arrived at a contradiction and must conclude that the lemma holds.

2.1.3 Related work

Search problems are optimization problems generally concerned with minimizing the time required for a set of mobile agents to find a hidden target in a given environment. One usually assumes that the environment is known in advance and the focus is on studying the effects on the search time under different assumptions on the agent capabilities. Searching in an unknown environment implies exploration where quite often there are additional/alternative goals the agents are required to achieve, e.g. mapping and/or positioning the searchers within the environment [3, 4, 85, 118].

When the environment is known, search is a pure optimization problem. The study of search by a single agent on the infinite line was initiated independently by Bellman [25] and Beck [22, 21, 23] where, among other things, the authors demonstrate the now well known result that a single searcher cannot find a hidden target at initial distance \( d \) from the searcher in time less than \( 9d \). The work by Bellman and Beck gave rise to a number of variants of search on the line. Notable is the work of Heath and Fristedt [106], Fristedt [105], and Gal [109]. Also notable is the works of Baeza-Yates et. al. [13, 15] where, among other things, the authors study problems of search by agents in environments different from the line, e.g. in the plane or at the origin of \( w \) concurrent rays (known as the “Lost Cow” problem). Group search was initiated in [40] where the problem of evacuation by multiple agents that can communicate face-to-face was studied. More recently, search on the line
was considered when: the agents have distinct speeds \[17\]; turning costs are included \[84\]; the concern is to minimize the energy consumed during the search \[60, 62\].

Search on the line with possibly faulty searchers was initiated in \[77\] wherein the authors introduce optimal trajectories – the \textit{proportional schedules} – for search by \(n\) agents at most \(f\) of which are crash faulty. This work is particularly relevant to the problem we study here. It should also be noted that the optimality of the proportional schedules for search was only established at a later time in \[136\]. Search with byzantine faults was first studied in \[70\] wherein the authors prove a number of lower bounds and upper bounds on the problem. Many of these upper bounds were later improved in \[156\], where, in particular, the authors demonstrate that the proportional schedules of \[77\] can be used to achieve an upper bound of 8.653055 on the problem of search by three agents, one of which is byzantine faulty.

### 2.1.4 Results and outline

Our main result is the development and analysis of a novel search type algorithm for the evacuation problem with crash faulty agents. A summary of the resulting upper bounds are listed in Table 2.1 along with the best known lower bounds. We also prove an asymptotic upper bound on the evacuation by \(n = 2f + 1\) agents of \(4 + 2\sqrt{2}\) (Theorem 2.2.8) and improve the upper bound on search by three agents, at most one of which is byzantine faulty, from 8.653055 to 7.437011 (Theorem 2.4.1). The best known lower bound on this search problem is 5.233069.

In Section 2.2.1 we analyze the competitive ratio of evacuation for the proportional schedules – a family of trajectories first developed in \[77\] for the purpose of search by crash faulty agents. We use this section to prove our asymptotic upper bound (Theorem 2.2.8) and also to build intuition on how we can improve upon these trajectories. In Section 2.3 we introduce a generalization of the proportional schedules and analyze separately the cases that \(f > 1\) (Subsection 2.3.1) and \(f = 1\) (Subsection 2.3.2). In Section 2.4 we show that our evacuation algorithm for three agents also leads to an improvement on the competitive ratio of search by three agents, one of which is byzantine faulty. Finally, in Section 2.5 we conclude with a brief discussion of open problems.
2.2 Crash faulty evacuation

With the evacuation time expressed as in (2.1) it is clear that in order to optimize the evacuation time one needs to consider a trade-off between the search time \( S_f \) and the distance \( \Delta x_f \). This is in contrast to the normal search problem which aims only to optimize \( S_f \). Nevertheless, one can imagine that an algorithm that optimizes \( S_f \) would still provide a good starting point for studying the evacuation problem. Since it just so happens that an optimal algorithm for crash faulty search is known, we will use this approach to study the evacuation problem.

2.2.1 Proportional schedules

An optimal algorithm for the crash faulty search problem was introduced\(^1\) in [77]. This algorithm is referred to as a proportional schedule and is defined by the collection of \( n \) trajectories represented by the sequences of turning points

\[
d_{i,j} = r^{2i/n}(r-j)
\]

where \( r > 1 \) is a real number parameter, and \( i \) and \( j \) index the robots and turning points respectively. Figure 2.1 depicts example proportional schedules for \( n = 5, 7, 9 \) using a space-time diagram which plots an agent’s position on the \( x \) axis with time on the \( y \)-axis.

\[\text{Figure 2.1: Example space-time trajectories when } r = 2 \text{ and } n = 5, 7, 9.\]

Note that each of the turning-points lie along a cone with slope \( \frac{r+1}{r-1} \).

Our strategy for analyzing this algorithm derives from equation (2.1). We will first compute \( S_f^x = S_f^x(r) \) and \( \Delta x_f^y = \Delta x_f^y(r) \). Our goal will be to prove the following Theo-

\(^1\)The algorithm was later proven to be optimal in [136].
Note that we will not analyze the exact evacuation time at this point since as we will eventually describe a better algorithm.

**Theorem 2.2.1.** For all $\epsilon > 0$ the evacuation time of the proportional schedules satisfies

$$1 + \frac{2r}{r + r^{2/n}} + \frac{4r^{2+1/n}}{(1 + \epsilon)(r + r^{2/n})(r - 1)} \leq R_f < 1 + \frac{2r}{r + 1} + \frac{4r^{2+1/n}}{(r + 1)(r - 1)}.$$

We make note of the following properties of the turning-points $d_{i,j}$ which hold for any agent $i$, turning point $j$, and integer $k$

$$d_{i+k,j} = r^{2k/n}d_{i,j}, \quad d_{i,j+k} = (-r)^kd_{i,j}.$$  \hfill (2.3)

Particularly useful is the fact that

$$d_{i+kn,j} = r^{2k/n}d_{i,j} = d_{i,j+2k} \hfill (2.4)$$

which allows us to refer to a turning-point $j$ of an agent with “label” $i \geq n$ or $i < 0$ with the understanding that we are actually referring to a later/earlier turning-point of agent $i \mod n$. We will make use of these last three properties often and without reference.

**Lemma 2.2.2.** Define $t_{i,j}$ to be the first time at which agent $i$ reaches location $d_{i,j}$. Then we have

$$t_{i,j} = \frac{r + 1}{r - 1}|d_{i,j}|.$$

**Proof.** Agent $i$ will reach $d_{i,j-1}$ at time $t_{i,j-1}$ and will then travel distance $|d_{i,j} - d_{i,j-1}|$ at unit speed to reach location $d_{i,j}$. We thus have the recursion

$$t_{i,j} = t_{i,j-1} + |d_{i,j} - d_{i,j-1}| = t_{i,j-1} + (r + 1)|d_{i,j-1}|.$$

Unrolling this recursion leads to

$$t_{i,j} = (r + 1) \sum_{k=-\infty}^{i-1} |d_{i,k}| = (r + 1)\frac{r^{2i/n}}{r - 1} \sum_{k=-\infty}^{j-1} r^k = \frac{r + 1}{r - 1}r^{2i/n+j} = \frac{r + 1}{r - 1}|d_{i,j}|.$$

We observed in the caption of Figure 2.1 that the turning points of the agents all lie along a common cone. This is evidenced by the ratio $t_{i,j} / |d_{i,j}| = \frac{r + 1}{r - 1}$ being independent of $i$ and $j$. We will make use of this property shortly.

We define the interval $I_{i,j}$ as follows.

**Definition 2.2.3.** The interval $I_{i,j}$ is defined as the semi-open interval

$$I_{i,j} := (d_{i,j}, d_{i+1,j}] = (1, r^{2/n}] \cdot d_{i,j}.$$
The sequence of intervals \([I_{i,j}]_{-\infty}^{\infty}\) with \(j\) even (resp. \(j\) odd) covers the entire positive (resp. negative) half-line without overlap. Thus, for any fixed \(j\) and position \(x\) there exists a unique integer \(i\) for which the position \(x \in I_{i,j}\). Using the property (2.4), we can equivalently say that there exists a unique \(j\) and \(i \in \{0, \ldots, n-1\}\) for which \(x \in I_{i,j}\). Note also that the symmetry inherent to the trajectories implies that for a fixed \(z \in (1, r^{2/n}]\) the competitive ratio when the target is placed at \(x = zd_{i,j} \in I_{i,j}\) will not depend on \(i\) or \(j\).

The next lemma will allow us to compute the search time of the algorithm.

**Lemma 2.2.4.** Define the time \(T_{i,j,k}(z)\) as the time of the first visit by agent \(i + k\), \(k = 1, \ldots, n\), to the location \(zd_{i,j} \in I_{i,j}\). Then

\[
T_{i,j,k}(z) = \left( z + \frac{2r^{2k/n}}{r-1} \right) |d_{i,j}|. \tag{2.5}
\]

**Proof.** Since \(d_{i,j} \not\in I_{i,j}\), agent \(i\) will not reach \(x\) until after its turning point \(d_{i,j} + 1\). On the other hand, agent \(i + 1\) will visit \(x\) while traveling towards \(d_{i+1,j}\) and it is clear that this will be the first visit of agent \(i + 1\) to \(x\). More generally, agent \(i + k\), \(k > 0\), will reach \(x\) for the first time while traveling towards \(d_{i+k,j}\). We can thus conclude that

\[
T_{i,j,k}(z) = t_{i+k,j} - |d_{i+k,j} - zd_{i,j}| = t_{i+k,j} - |d_{i+k,j}| + z|d_{i,j}|
\]

\[
= \frac{r + 1}{r - 1} |d_{i+k,j}| - |d_{i+k,j}| + z|d_{i,j}| = \frac{2}{r - 1} |d_{i+k,j}| + z|d_{i,j}|
\]

\[
= \left( z + \frac{2r^{2k/n}}{r-1} \right) |d_{i,j}|.
\]

\[
\square
\]

We now turn our attention to the distance \(\Delta x_{f}\). We will only bound this distance and to do this we inspect the space-time points at which the trajectories of the agents intersect.

**Lemma 2.2.5.** Respectively define \(\rho_{i,j,k}\) and \(\tau_{i,j,k}\), \(k = 0, 1, \ldots, f\), as the position and time at which the trajectory of agent \(i\) intersects the trajectory of agent \(i + k\) while agent \(i\) is traveling away from its turning point \(d_{i,j}\) and agent \(i + k\) is traveling towards its turning point \(d_{i+k,j}\). Then

\[
\rho_{i,j,k} = \frac{r - r^{2k/n}}{r-1} d_{i,j}, \quad \tau_{i,j,k} = \frac{r + r^{2k/n}}{r-1} |d_{i,j}|.
\]

**Proof.** One can observe that while travelling away from its turning point \(j\) agent \(i\) will be moving in the direction \(-(-1)^j\) along a line with equation

\[
t = t_{i,j} - (-1)^j (x - d_{i,j})
\]
On the other hand, agent $i + k$ will be traveling in the direction $(-1)^j$ along the line

$$t = t_{i+k,j} + (-1)^j(x - d_{i+k,j})$$

while moving towards its turning point $d_{i+k,j}$. The position $\rho_{i,j,k}$ at which agent $i$ and $i + k$ meet can thus be determined by subtracting one of these lines from the other and solving for $x$. We find that

$$\rho_{i,j,k} = \frac{(-1)^j}{2} \left[ (-1)^j(d_{i,j} + d_{i+k,j}) + (t_{i,j} - t_{i+k,j}) \right]$$

$$= \frac{(-1)^j}{2} \left[ (|d_{i,j}| + |d_{i+k,j}|) + \frac{r + 1}{r - 1}(|d_{i,j}| - |d_{i+k,j}|) \right]$$

$$= \frac{(-1)^j |d_{i,j}|}{2} \left[ (1 + r^{2k/n}) + \frac{r + 1}{r - 1}(1 - r^{2k/n}) \right]$$

$$= \frac{d_{i,j}}{2(r - 1)} \left[ (r - 1)(1 + r^{2k/n}) + (r + 1)(1 - r^{2k/n}) \right]$$

$$= \frac{d_{i,j}}{2(r - 1)} \left[ 2r - 2r^{2k/n} \right] = \frac{r - r^{2k/n}}{r - 1} d_{i,j}.$$

The time $\tau_{i,j}$ is then

$$\tau_{i,j} = -(1)^j(\rho_{i,j,k} - d_{i,j}) + t_{i,j} = -|d_{i,j}| \left( \frac{r - r^{2k/n}}{r - 1} - 1 \right) + \frac{r + 1}{r - 1} |d_{i,j}|$$

$$= \frac{r + 1 + r - 1 - r + r^{2k/n}}{r - 1} |d_{i,j}| = \frac{r + r^{2k/n}}{r - 1} |d_{i,j}|.$$

Note that when $k = 0$ we have $\rho_{i,j,0} = d_{i,j}$ and $\tau_{i,j,0} = t_{i,j}$ and thus this “self intersection” point is just a turning point.

The space-time points $(\rho_{i,j,k}, \tau_{i,j,k})$ for various $k$ are depicted in Figure 2.2. One can observe that for a fixed $k$ the points $(\rho_{i,j,k}, \tau_{i,j,k})$ all lie along a common cone. Let $C_k$ represent the cone corresponding to points $(\rho_{i,j,k}, \tau_{i,j,k})$. Then the slope of $C_k$ is $\beta_k := \frac{\tau_{i,j,k}/|\rho_{i,j,k}|}{r + r^{2k/n} - r^{2k/n}}$. By referring to Figure 2.2 one can observe that at all times $t > 0$ and for each $k = 0, \ldots, f - 1$, there exists an agent on either side of the origin in the annular region bounded by cones $C_k$ and $C_{k+1}$ (this fact also follows easily from the definition of $(\rho_{i,j,k}, \tau_{i,j,k})$). Of particular interest is the fact that there always exists an agent located between the cones $C_0$ and $C_1$ since this agent is the most distant from the origin (on its respective side).

**Observation 2.2.6.** At all times $t > 0$ there exists an agent located within each of the intervals $\pm \left[ \pm \frac{t}{\beta_1}, \frac{t}{\beta_0} \right]$. 44
Figure 2.2: Illustrating the points \((\rho_{i,j,k}, \tau_{i,j,k})\) and their respective cones. Only \((\rho_{i,j,1}, \tau_{i,j,1})\) are indicated.

This observation then easily leads to the following bound on \(\Delta_f^x\).

**Lemma 2.2.7.**

\[
|\mathbf{x}| + \frac{S_f^x}{\beta_1} \leq \Delta_f^x \leq |\mathbf{x}| + \frac{S_f^x}{\beta_0}.
\]

We can now prove Theorem 2.2.1.

**Proof.** In the worst case the target is just beyond a turning point and so we assume that the target is at location \(\mathbf{x} = (1 + \epsilon)d_{i,j}\), with \(\epsilon > 0\) arbitrarily small. Also suppose that it is agent \(k \in \{1, \ldots, f+1\}\) that is the first reliable agent to reach the target. Then by Lemma 2.2.4, Lemma 2.2.7, and equation (2.1) we have

\[
|x| + T_{i,j,k}(1 + \epsilon) \left(1 + \frac{1}{\beta_1}\right) \leq E_f^x \leq |x| + T_{i,j,k}(1 + \epsilon) \left(1 + \frac{1}{\beta_0}\right)
\]

Dividing by \(|x| = (1 + \epsilon)|d_{i,j}|\), substituting in the expressions for \(T_{i,j,k}(1), \beta_0,\) and \(\beta_1\) then yields

\[
1 + \frac{1}{1 + \epsilon} \left(1 + \epsilon + \frac{2r^{2k/n}}{r-1}\right) \left(1 + \frac{r - r^{2/n}}{r + r^{2/n}}\right) \leq E_f
\]

\[
\leq 1 + \frac{1}{1 + \epsilon} \left(1 + \epsilon + \frac{2r^{2k/n}}{r-1}\right) \left(1 + \frac{r - 1}{r + 1}\right)
\]

\[
1 + \frac{2r}{r + r^{2/n}} + \frac{4r^{1+2k/n}}{(1 + \epsilon)(r + r^{2/n})(r - 1)} \leq R_f \leq 1 + \frac{2r}{r + 1} + \frac{4r^{1+2k/n}}{(1 + \epsilon)(r + 1)(r - 1)}
\]
both sides of this inequality increase with $k$ and so taking $k = f + 1$ then yields

$$1 + \frac{2r}{r + r^{2/n}} + \frac{4r^{2+1/n}}{(1 + \epsilon)(r + r^{2/n})(r - 1)} \leq R_f \leq 1 + \frac{2r}{r + 1} + \frac{4r^{2+1/n}}{(1 + \epsilon)(r + 1)(r - 1)}.$$ 

The theorem then follows by taking $\epsilon \to 0$ on the right hand side.

To prove Theorem 2.2.1 we used the fact that the agent most distant from the target at time $S_x^f$ will be somewhere between the cones $C_0$ and $C_1$. Since $\lim_{n \to \infty} r^{2/n} = 1$ it is clear that $\lim_{n \to \infty} \beta_1 = \beta_0$, i.e. the cones $C_0$ and $C_1$ approach each other as $n$ gets large. This implies that the bounds of Theorem 2.2.1 will also approach each other for large $n$. This immediately leads to the following conclusion.

**Theorem 2.2.8.** The asymptotic competitive ratio of the proportional schedule algorithm is

$$\hat{R} = \lim_{f \to \infty} R_f = 7 - \frac{2(r - 3)}{r^2 - 1}.$$

In particular, if we take $r = 3 + 2\sqrt{2}$ then $\hat{R} = 4 + 2\sqrt{2}$.

**Proof.** Following from the discussion preceding this theorem we have

$$\hat{R} = \lim_{f \to \infty} \left[ 1 + \frac{2r}{r + 1} + \frac{4r^{2+1/n}}{(r + 1)(r - 1)} \right] = 1 + \frac{2r}{r + 1} + \frac{4r^2}{(r + 1)(r - 1)} = 1 + \frac{6r^2 - 2r}{r^2 - 1}$$

$$= 7 - \frac{2(r - 3)}{r^2 - 1}.$$ 

The second part of the lemma follows by optimizing the asymptotic competitive ratio with respect to $r$. Observe that

$$\frac{d\hat{R}}{dr} = \frac{4r(r - 3)}{(r^2 - 1)^2} - \frac{2}{r^2 - 1}.$$ 

Setting this equal to zero and rearranging yields the quadratic equation

$$r^2 - 6r + 1 = 0$$

which can be solved to find $r = 3 \pm 2\sqrt{2}$. Taking the positive root (since $r > 1$) and substituting this into the expression for $\hat{R}$ yields

$$\hat{R} = 7 - \frac{4\sqrt{2}}{(3 + 2\sqrt{2})^2 - 1} = 7 - \frac{4\sqrt{2}}{16 + 12\sqrt{2}} = 7 - \frac{1}{2\sqrt{2} + 3}$$

$$= 7 - \frac{3 - 2\sqrt{2}}{(2\sqrt{2} + 3)(3 - 2\sqrt{2})} = 7 - (3 - 2\sqrt{2}) = 4 + 2\sqrt{2}.$$ 

$\square$
To compute an exact expression for the competitive ratio as a function of \( r \) we would need to determine the identity \( i_f^x \) of the agent that is most distant from \( x \) at the time \( S_f^x \). Although this is not very difficult to do, it does not add to the results of the paper since we will be improving upon this algorithm in the next section. It is useful, however, to know the optimal competitive ratio for this algorithm for the sake of comparison and discussion. Figure 2.3 shows a plot of the optimized competitive ratios as a function of \( f \) along with the bounds from Theorem 2.2.1 evaluated with the optimized parameter \( r \). Also shown is the competitive ratio when \( r \) is chosen to optimize the search time only. One can observe that in all cases except \( f = 1 \), the actual competitive ratio is essentially identical to the lower bound implying that the optimal choice of \( r \) places the agent \( i_f^x \) on or near the interior cone \( C_1 \) at the time \( S_f^x \). This observation leads one to question whether or not the single degree of freedom provided by the parameter \( r \) is sufficient to facilitate an efficient trade-off between the search time and the distance \( \Delta_f^x \). In the next section we validate this concern and show that a generalized form of the proportional schedule leads to an improvement in the competitive ratio.

### 2.3 Generalized (proportional) schedules

In this section we consider a generalized form of the proportional schedule. Put simply, we will add an extra two turning points between each pair of turning points \( d_{i,j} \) and \( d_{i,j+1} \) of the normal proportional schedule. We will refer to these “sub-turning points” using the notation \( d_{i,j}^{0, \ell} \), \( \ell = 0, 1, 2 \). An intuitive parameterization of the turning points uses parameters
\( s \in [0, r + 1] \) and \( a \in [s - 1, r] \) as follows

\[
d^{(\ell)}_{i,j} = d_{i,j} \cdot \begin{cases} 
1, & \ell = 0 \\
-a, & \ell = 1 \\
s - a, & \ell = 2
\end{cases}
\]  

(2.6)

The parameter \( s \) controls the distance between \( d^{(1)}_{i,j} \) and \( d^{(2)}_{i,j} \), and \( a \) controls the location of \( d^{(1)}_{i,j} \) (relative to \( d_{i,j} \)). With the bounds given on \( s \) and \( a \) we will have \( d^{(1)}_{i,j} \in [d_{i,j}, d_{i,j+1}] \) and \( d^{(2)}_{i,j} \in [d_{i,j}, d_{i,j+1}] \). One can also observe that when \( s = 0 \) these trajectories are identical to those of the proportional schedule and so this can be rightly called a generalization.

Although the parameterization using \((r, s, a)\) is intuitive, it will be much more convenient to replace \( s \) with the parameter \( q := \frac{r + s}{r - 1} \). We will use both of these parameterizations, however, we will favor the one with \( q \). For the parameterization with \( q \) we have

\[
d^{(\ell)}_{i,j} = d_{i,j} \cdot \begin{cases} 
1, & \ell = 0 \\
-a, & \ell = 1 \\
q(r - 1) - r - a, & \ell = 2
\end{cases}
\]  

(2.7)

We make note of the following identities concerning the parameters \( q \) and \( s \).

\[
q = \frac{r + s}{r - 1}, \quad q - 1 = \frac{1 + s}{r - 1}, \quad q + s = r(q - 1).
\]  

(2.8)

We will use these identities repeatedly and without reference.

One approach to analyzing these trajectories would be to compute and optimize the competitive ratio as a function of the parameters \((r, q, a)\). This would involve a nightmarish case analysis that would scare away even the most interested readers. Alas, this is not the approach we take. Instead we will describe sets of objectively good choices for the parameters \( q \) and \( a \) and express the competitive ratio of the resulting trajectories as a function of the parameter \( r \). We will use the results from the previous section to guide us as much as possible. The analysis of the cases \( f = 1 \) and \( f > 1 \) is sufficiently different to warrant considering each separately. We will begin with the case that \( f > 1 \) since this case closely mirrors that of the vanilla proportional schedules.

### 2.3.1 Many faults

Recall that the worst case scenario for the proportional schedules occurs when the target is placed just beyond a turning point \( d_{i,j} \) and the first \( f \) agents that visit the target are faulty. We will refer to this scenario as Scenario A in order to refer to it quickly. Also recall that, in all cases that \( f > 1 \), it was optimal to choose \( r \) so that the agent \( i_f \) was located on or near
the inner bounding cone $C$ at the time $S^x_f$ in order to minimize the distance $\Delta^x_f$ in the event that Scenario A occurs. Scenario A will still be a potential worst case for the generalized schedule, however, we can now use our extra degrees of freedom to ensure that agent $i^x_f$ is located on the cone $C$ at the time $S^x_f$ while leaving the parameter $r$ to facilitate a more efficient tradeoff between $\Delta^x_f$ and $S^x_f$. Our goal is to prove the following theorem.

**Theorem 2.3.1.** Fix the number of faults $f > 1$ with $n = 2f + 1$. Define the functions

$$\hat{q}(r, u) := \frac{r^{2u/n-1} + 1}{(1 + r^{2/n})^{2(u-1)/n-1} - 2r^{1/n}} \quad (2.9)$$

$$\hat{a}(r, q) := \begin{cases} q(r^{1-2/n} - 1), & q \leq \frac{r}{r - r^{2/n}} \\ q(r^{1-4/n} - 1), & otherwise \end{cases} \quad (2.10)$$

and the set

$$P := \{(r, u) \mid r > 1, u \in \{f + 3, \ldots, n\}, \frac{r}{r - 1} \leq \hat{q}(r, u) \leq \min\left\{ \frac{r}{r - r^{1-2/n}}, \frac{r}{r - r^{4/n}} \right\}\} \quad (2.11)$$

Then, for pairs $(r, u) \in P$, the competitive ratio of the generalized proportional schedule with parameters $q = \hat{q}(r, u)$, and $a = \hat{a}(r, \hat{q}(r, u))$ is

$$R_f = \begin{cases} R^A_f, & q \leq \frac{r}{r - r^{2/n}} \\ \max\{R^A_f, R^B_f\}, & otherwise \end{cases} \quad (2.12)$$

where

$$R^A_f = 1 + \frac{2q(1 + qr^{1/n})}{q + (q - 1)r^{2/n}}, \quad and \quad R^B_f \leq 3 + \frac{2(q - 1)}{q(r^{1-4/n} - 1)} \left[ 2r - \frac{r^{3/n}(1 + 2q^{1/n})}{q + (q - 1)r^{2/n}} \right].$$

The rough idea behind this theorem is as follows. Taking $q = \hat{q}(r, u)$ ensures that the agent $i + u$ will be located on the cone $C$ (properly modified for the generalized schedules) in the event that Scenario A occurs (see Lemma 2.3.7). Choosing $a = \hat{a}(r, q)$ ensures that agent $i + u$ will be the most distant agent in the event of Scenario A (see Lemma 2.3.6). The quantity $R^A_f$ gives the competitive ratio of Scenario A and $R^B_f$ gives the competitive ratio of an additional potential worst case. One can refer to Figure 2.5 at the end of this subsection for an illustration of the optimized trajectories resulting from these parameter choices for the cases $f = 2, 3, 4$.

To proceed we need to define analogues of the quantities $t_{i,j}$, $T_{i,j,k}(z)$, $\tau_{i,j,k}$, and $\rho_{i,j,k}$ in the context of the generalized schedules.
Lemma 2.3.2. The time $t_{i,j}^{(\ell)}$ at which agent $i$ reaches its sub-turning point $d_{i,j}^{(\ell)}$ is

$$t_{i,j}^{(\ell)} = |d_{i,j}| \cdot \begin{cases} 2q - 1, & \ell = 0 \\ 2q + a, & \ell = 1 \\ q(r + 1) - r + a, & \ell = 2 \end{cases} \quad (2.13)$$

Proof. While traveling from $d_{i,j-1}$ to $d_{i,j}$ agent $i$ will travel distance $|d_{i,j} - d_{i,j-1}|$ plus twice the distance between $d_{i,j-1}^{(1)}$ and $d_{i,j-1}^{(2)}$. We thus have

$$t_{i,j} = t_{i,j-1} + |d_{i,j} - d_{i,j-1}| + 2|d_{i,j-1}^{(2)} - d_{i,j-1}^{(1)}| = t_{i,j-1} + (r + 1)|d_{i,j-1}| + 2s|d_{i,j-1}|
$$

$$= t_{i,j-1} + (r + 1 + 2s)|d_{i,j-1}| = (r + 1 + 2s)\sum_{k=1}^{j-1} |d_{i,k}|
$$

$$= \frac{r + 1 + 2s}{r - 1} |d_{i,j}| = (2q - 1)|d_{i,j}|.$$

After reaching $d_{i,j}$ at time $t_{i,j}$ agent $i$ must travel distance $|d_{i,j}^{(1)} - d_{i,j}| = (1 + a)|d_{i,j}|$ to reach $d_{i,j}$ and thus $t_{i,j}^{(1)} = t_{i,j} + (1 + a)|d_{i,j}| = (2q + a)|d_{i,j}|$. Similarly, after reaching $d_{i,j}$ at time $t_{i,j}^{(2)}$ agent $i$ must travel distance $|d_{i,j}^{(2)} - d_{i,j}^{(1)}| = s|d_{i,j}|$ to reach $d_{i,j}^{(2)}$. Thus

$$t_{i,j}^{(2)} = t_{i,j}^{(1)} + s|d_{i,j}| = t_{i,j}^{(1)} + (1 + s + a)|d_{i,j}| = (2q + s + a)|d_{i,j}| = [q(r + 1) - r + a]|d_{i,j}|.$$

Lemma 2.3.3. The time $T_{i,j,k}(z)$ at which agent $i + k$ first reaches location $z = zd_{i,j} \in I_{i,j}$ is

$$T_{i,j,k}(z) = \begin{cases} T_{i,j,k}^o(z) := \left[ z + 2qr^{2k/n-1} \right]|d_{i,j}|, & a \geq \frac{z^{r^{2k/n}}}{r^{2k/n}} \\ T_{i,j,k}^+(z) := \left[ z + 2(q - 1)r^{2k/n} \right]|d_{i,j}|, & a < \frac{z^{r^{2k/n}}}{r^{2k/n}} \end{cases} \quad (2.14)$$

Proof. As was the case in Lemma 2.2.4 agent $i + k, k = 1, \ldots, n$, will arrive to location $x$ while traveling between its turning points $d_{i+k,j-1}$ and $d_{i+k,j}$. However, for the generalized algorithm, the expression for $T_{i,j,k}(z)$ will depend on whether or not agent $i + k$ reaches $x$ before or after its sub-turning point $d_{i+k,j-1}^{(1)}$, i.e. whether or not we have $x \in [d_{i+k,j-1}, d_{i+k,j-1}^{(1)}]$ or $x \in (d_{i+k,j-1}^{(1)}, d_{i+k,j-1})$. We consider first the case that $x \in [d_{i+k,j-1}, d_{i+k,j-1}^{(1)}]$. We have

$$T_{i,j,k}(z) = t_{i+k,j-1} + |zd_{i,j} - d_{i+k,j-1}| = t_{i+k,j-1} + z|d_{i,j}| + |d_{i+k,j-1}|
$$

$$= 2q|d_{i+k,j-1}| + z|d_{i,j}| = \left[ z + 2qr^{2k/n-1} \right]|d_{i,j}|$$

If $x \in (d_{i+k,j-1}^{(1)}, d_{i+k,j-1})$ then agent $i + k$ will travel an extra distance equal to $2|d_{i+k,j-1}^{(2)} - d_{i+k,j-1}^{(1)}| = 2s|d_{i+k,j-1}|$ to reach $x$ as compared to the previous case. We thus have

$$T_{i,j,k}(z) = \left[ z + 2qr^{2k/n-1} \right]|d_{i,j}| + 2s|d_{i+k,j-1}| = \left[ z + (2q + s)r^{2k/n-1} \right]|d_{i,j}|
$$

$$= \left[ z + 2(q - 1)r^{2k/n} \right]|d_{i,j}|.$$
We now establish under what conditions we find ourselves in each of these two cases. To do this we need to compare \( x \) to \( d_{i+k,j-1}^{(1)} \). To have \( x \in [d_{i+k,j-1}, d_{i+k,j-1}^{(1)}] \) when \( j \) is even we need \( x \leq d_{i+k,j-1} \) and when \( j \) is odd we need \( x \geq d_{i+k,j-1} \). We thus need to consider the inequality

\[
(-1)^j x \leq (-1)^j d_{i+k,j-1}^{(1)} \quad \Rightarrow \quad z|d_{i,j}| \leq ar^{2k/n-1}|d_{i,j}| \quad \Rightarrow \quad a \geq \frac{z r}{r^{2k/n}}.
\]

The proportional schedules enjoyed the property that the \((f+1)\text{st}\) agent to reach any location \( x = zd_{i,j} \in I_{i,j} \) was the agent \( i + f + 1 \). The next lemma outlines the conditions for this to also be the case with the generalized schedules.

**Lemma 2.3.4.** Suppose that \( a \geq r^{1/n} \). Then agent \( i + k, k = f + 1, \ldots, n, \) will reach any position \( x = zd_{i,j} \in I_{i,j} \) at the time \( T_{i,j,k}(z) = T_{i,j,k}^{(0)}(z) \). Furthermore,

1. if \( q \leq \frac{r}{r^{1/n}} \) then agents \( i + 1, \ldots, i + f \) will reach \( x \) before agent \( i + f + 1 \)
2. if \( \frac{r}{r^{1/n}} < q \leq \frac{r}{r^{2/n}} \) then agents \( i + 1, \ldots, i + f - 1 \) will reach \( x \) before agent \( i + f + 1 \) and agent \( i + f \) will reach \( x \) before agent \( i + f + 1 \) provided that \( a \geq r^{2k/n} \).

**Proof.** By Lemma 2.3.3 we will have \( T_{i,j,f+1}(z) = T_{i,j,f+1}^{(0)}(z) \) when \( a \geq \frac{r}{r^{1/n}} = \frac{z r}{r^{1/n}} \).

For \( x \in I_{i,j} \) we have \( z \in [1, r^{2/n}] \) and thus agent \( i \) will reach any position \( x \in I_{i,j} \) provided that \( a \geq \frac{r^{2/n}}{r^{1/n}} = r^{1/n} \).

Now suppose that \( a \geq r^{1/n} \). Since the condition \( a \geq \frac{r}{r^{1/n}} \) gets easier to satisfy for larger \( k \) we can conclude that agents \( i + k, k = f + 2, \ldots, n, \) will reach \( x \) at times \( T_{i,j,k}(z) = T_{i,j,k}^{(0)}(z) \). Let \( \dagger = 0, + \) and observe that for fixed \( i, j, \) and \( z \) we have

\[
T_{i,j,1}^{(0)}(z) < T_{i,j,2}^{(0)}(z) < \ldots < T_{i,j,f+1}^{(0)}(z).
\]

In particular, agents \( i + f + 2, \ldots, i + n \) will arrive to \( x \) after agent \( i + f + 1 \). Thus, in order to be the \((f + 1)\text{st}\) agent to reach \( x \), agents \( i + 1, i + 2, \ldots, i + f \) must reach \( x \) before agent \( i + f + 1 \).

Define \( k_\ast \) as the largest integer such that \( a < \frac{r}{r^{2k_\ast/n}} \). Then for each \( k \leq k_\ast \) we have \( T_{i,j,k}(z) = T_{i,j,k}^{(0)}(z) \) and agent \( i + k_\ast \) will arrive to \( x \) after agents \( i + 1, \ldots, i + k_\ast - 1 \). Moreover, agents \( i + k_\ast + 1, \ldots, i + f \) will arrive to \( x \) before agent \( i + f + 1 \). Thus, to make sure agent \( i + f + 1 \) is the \((f + 1)\text{st}\) agent to reach \( x \) we must make sure that \( T_{i,j,f+1}^{(0)}(z) \geq T_{i,j,k_\ast}(z) \). We have

\[
T_{i,j,f+1}^{(0)}(z) \geq T_{i,j,k_\ast}^{(0)}(z) \quad \Rightarrow \quad [z + 2qr^{2(f+1)/n-1}]|d_{i,j}| \geq [z + 2(q-1)r^{2k_\ast/n}]|d_{i,j}| \quad \Rightarrow \quad qr^{2(f+1)/n-1} \geq (q-1)r^{2k_\ast/n} \quad \Rightarrow \quad \frac{q}{q-1}r^{1/n} \geq r^{2k_\ast/n}.
\]
We can further rearrange this to get
\[ q \leq \frac{r}{r - r^{2(f+1-k_f)}/n}. \]

Taking \( k_f \leq f \) demonstrates that agents \( i + 1, \ldots, i + f \) will all reach \( x \) before agent \( i + f + 1 \) when \( q \leq \frac{r}{r - r^{2(f+1-k_f)}/n} \). Similarly, taking \( k_f \leq f - 1 \) demonstrates that agents \( i + 1, \ldots, i + f - 1 \) will all reach \( x \) before agent \( i + f + 1 \) when \( q \leq \frac{r}{r - r^{2(f+1-k_f)}/n} \). Now suppose that \( \frac{r}{r - r^{2(f+1-k_f)}/n} < q \leq \frac{r}{r - r^{2(f+1-k_f)}/n} \). Then we claim that agent \( i + f \) reaches its turning point \( d_{i+f,j-1}^{(1)} \) somewhere in \( I_{i,j} \) and agent \( i + f + 1 \) will be the \((f + 1)^{st}\) agent to reach only the points \((d_{i,j}, d_{i+f,j-1}^{(1)})\). Indeed, we can observe that

\[ d_{i+f,j-1}^{(1)} = ar^{2f/n-1}d_{i,j} = \frac{a}{r^{1/n}}d_{i,j} \]

and since we are assuming that \( a \geq r^{1/n} \) we have \( d_{i+f,j-1}^{(1)} \in I_{i,j} \). We can also conclude from this that agent \( i + f \) will arrive before agent \( i + f + 1 \) to exactly those positions \( x = zd_{i,j} \) for which \( z \leq \frac{a}{r^{1/n}} \), i.e. \( a \geq zr^{1/n} \). \( \square \)

**Lemma 2.3.5.** Suppose that \( a \geq r^{1/n} \). Then the point \((\rho_{i,j,k}, \tau_{i,j,k}), k = 0, \ldots, f\), at which the trajectory of agent \( i \) intersects that of agent \( i + k \) while agent \( i \) is moving away from \( d_{i,j} \) and agent \( i + k \) is moving towards \( d_{i+k,j} \) is

\[ (\rho_{i,j,k}, \tau_{i,j,k}) = \begin{cases} (\rho_{i,j,k}^o, \tau_{i,j,k}^o), & a \geq q(r^{1-2k/n} - 1) \\ (\rho_{i,j,k}^+, \tau_{i,j,k}^+), & \text{otherwise} \end{cases} \] \hspace{1cm} (2.15)

where

\[ \rho_{i,j,k}^o := q(1 - r^{2k/n-1})d_{i,j}, \quad \tau_{i,j,k}^o := q(1 + r^{2k/n-1})|d_{i,j}| \]

and

\[ \rho_{i,j,k}^+ := [q - (q - 1)r^{2k/n}]d_{i,j}, \quad \tau_{i,j,k}^+ := [q + (q - 1)r^{2k/n}]|d_{i,j}|. \]

**Proof.** To determine a general expression for \( \rho_{i,j,k} \) and \( \tau_{i,j,k} \) one must consider multiple cases depending on where the intersection of the sub-turning points. Agent \( i \) will either be located in the interval \([d_{i,j}, d_{i,j}^{(1)}] \) or the interval \([d_{i,j}^{(1)}, d_{i,j+1}] \) and agent \( i + k \) will either be in \([d_{i+k,j-1}, d_{i+k,j-1}^{(1)}] \) or \([d_{i+k,j-1}^{(1)}, d_{i+k,j}] \). We will only be interested in the two cases that agent \( i \) is in \([d_{i,j}, d_{i,j}^{(1)}] \) and agent \( i + 1 \) is in \([d_{i+k,j-1}, d_{i+k,j-1}^{(1)}] \) or \([d_{i+k,j-1}^{(1)}, d_{i+k,j}] \). One can refer to Figure 2.4 for an illustration of each of these two cases.

Let \( \rho_{i,j,k}^o \) and \( \tau_{i,j,k}^o \) correspond to the intersection points when agent \( i \) is moving between \([d_{i,j}, d_{i,j}^{(1)}] \) and agent \( i + 1 \) is moving between \([d_{i+k,j-1}, d_{i+k,j-1}^{(1)}] \) (this situation is depicted
on the left side of Figure 2.4. Let $\rho_{i,j,k}^+\circ$ and $\tau_{i,j,k}^+\circ$ correspond to the intersection points when agent $i$ is moving between $[d_{i,j}, d_{i,j}^{(1)}]$ and agent $i + 1$ is moving between $(d_{i+k,j-1}^{(1)}, d_{i+k,j}]$ (this situation is depicted on the right side of Figure 2.4).

If agent $i$ is moving between $[d_{i,j}, d_{i,j}^{(1)}]$ then it will be traveling in the direction $-(-1)^j$ along the line with equation

$$t = t_{i,j} - (-1)^j(x - d_{i,j}) = t_{i,j} + |d_{i,j}| - (-1)^j x. \quad (2.16)$$

Consider first $\rho_{i,j,k}^\circ\circ$ and $\tau_{i,j,k}^\circ\circ$. In this case agent $i + k$ will be traveling in the direction $(-1)^j$ along the line with equation

$$t = t_{i+k,j-1} + (-1)^j(x - d_{i+k,j-1}) = t_{i+k,j-1} + |d_{i+k,j-1}| + (-1)^j x$$

$$= r^{2k/n-1}(t_{i,j} + |d_{i,j}|) + (-1)^j x.$$ 

Subtracting this equation from (2.16) and solving for $x$ yields

$$\rho_{i,j,k}^\circ\circ = \frac{(-1)^j(1 - r^{2k/n-1})}{2} (t_{i,j} + |d_{i,j}|) = \frac{(-1)^j(1 - r^{2k/n-1})}{2} ((2q - 1)|d_{i,j}| + |d_{i,j}|)$$

$$= q(1 - r^{2k/n-1})d_{i,j}.$$ 

Substituting this into (2.16) then yields

$$\tau_{i,j,k}^\circ\circ = t_{i,j} + |d_{i,j}| - (-1)^j\rho_{i,j,k}^\circ\circ = 2q|d_{i,j}| - q(1 - r^{2k/n-1})|d_{i,j}|$$

$$= q(1 + r^{2k/n-1})d_{i,j}|$$
This case will occur provided that \( \tau_{i,j,k}^0 \leq i_{i,j}^{(1)} \) and \( \tau_{i,j,k}^0 \leq t_{i+k,j-1}^{(1)} \). We get from the first inequality

\[
\tau_{i,j,k}^0 \leq i_{i,j}^{(1)} \quad \rightarrow \quad q(1 + r^{2k/n - 1})|d_{i,j}| \leq t_{i,j} + (1 + a)|d_{i,j}| = (2q + a)|d_{i,j}|
\]

and finally

\[
a \geq q(r^{2k/n - 1} - 1)
\]

The right hand side of this inequality is negative for all \( k \leq f \) and the inequality is therefore satisfied for any \( a \geq r^{1/n} \). From the inequality \( \tau_{i,j,k}^0 \leq t_{i+k,j-1}^{(1)} \) we get

\[
q(1 + r^{2k/n - 1})|d_{i,j}| \leq (2q + a)|d_{i+k,j-1}| = (2q + a)r^{2k/n - 1}|d_{i,j}|
\]

which can be manipulated to yield

\[
a \geq q(r^{1-2k/n} - 1)
\]

which is of course the condition of (2.15).

Now consider \( \rho_{i,j,k}^+ \) and \( \tau_{i,j,k}^+ \). If we refer to Figure 2.4 one can observe that the point \((\rho_{i,j,k}^+, \tau_{i,j,k}^+)\) is shifted to the left (i.e. in the direction \(-(-1)^j\)) and up by an amount \( |d_{i+k,j-1}^{(2)} - d_{i+k,j-1}^{(1)}| \) as compared to the point \((\rho_{i,j,k}^0, \tau_{i,j,k}^0)\). Since \( |d_{i+k,j-1}^{(2)} - d_{i+k,j-1}^{(1)}| = s|d_{i+k,j-1}^{(1)} - d_{i+k,j-1}^{(2)}| = sr^{2k/n - 1}|d_{i,j}| \) we immediately find that

\[
\rho_{i,j,k}^+ = \rho_{i,j,k}^0 - (-1)^j|d_{i+k,j-1}^{(1)} - d_{i+k,j-1}^{(2)}| = q(1 - r^{2k/n - 1})d_{i,j} - sr^{2k/n - 1}d_{i,j}
\]

\[
= [q - (q + s)r^{2k/n - 1}]d_{i,j} = [q - (q - 1)r^{2k/n}]d_{i,j}.
\]

and, similarly,

\[
\tau_{i,j,k}^+ = [q + (q - 1)r^{2k/n}]|d_{i,j}|.
\]

As a last step we note that the definition of \( \rho_{i,j,0} \) and \( \tau_{i,j,0} \) being self intersection points is again a good interpretation. Indeed, we have

\[
\rho_{i,j,0} = \rho_{i,j,0}^+ = [q - (q - 1)]d_{i,j} = d_{i,j}
\]

\[
\tau_{i,j,0} = \tau_{i,j,0}^+ = [q + (q - 1)r^{2k/n}]|d_{i,j}| = (2q - 1)|d_{i,j}| = t_{i,j}.
\]

Let \( \tau \) be \( \tau_{i,j,k}^+ / |\rho_{i,j,k}^+| \) are independent of \( i \) and \( j \), for fixed \( k \) and \( \tau \) the points \((\rho_{i,j,k}^+, \tau_{i,j,k}^+)\) all lie along a common cone. Let \( \rho_{i,j,k}^0 \) refer to the cone with slope \( \beta_k := \tau_{i,j,k}^+ / |\rho_{i,j,k}^+| \) corresponding to the points \((\rho_{i,j,k}^+, \tau_{i,j,k}^+)\). The proportional schedules had the property that at all times \( t > 0 \) there was an agent between the cones \( C_0 \) and \( C_1 \) and
this agent was the most distant from the origin on its respective side. For the generalized schedules we will want the same property to hold for the cones $C^+_0$ and $C^+_1$. We observe that

$$
\beta_k^o := \frac{\tau^o_{i,j,k}}{|\rho^o_{i,j,k}|} = \frac{r + r^{2k/n}}{r - r^{2k/n}}, \quad \text{and} \quad \beta_k^+ := \frac{\tau^+_{i,j,k}}{|\rho^+_{i,j,k}|} = \frac{q + (q - 1)r^{2k/n}}{|q - (q - 1)r^{2k/n}|}.
$$

(2.17)

We include the absolute value in the denominator of $\beta_k^+$ since it is possible that $q - (q - 1)r^{2k/n} < 0$. We will later show that when $k = 1$ we will indeed have $q - (q - 1)r^{2k/n} \geq 0$ as a result of the condition $q \leq \frac{r}{r - r^{2/n}}$ in the definition of $P$.

**Lemma 2.3.6.** During the time interval $[\tau_{i-1,j,1}, \tau_{i,j,1}]$ agent $i$ has the most negative/positive position when $j$ is odd/even provided that

$$a = q(r^{1-4/n} - 1)$$

or

$$q(r^{1-4/n} - 1) < a \leq q(r^{1-2/n} - 1) \quad \text{and} \quad q \leq \frac{r}{r - r^{2/n}}.$$

**Proof.** We will only be considering values of $a$ satisfying $q(r^{1-4/n} - 1) \leq a \leq q(r^{1-2/n} - 1)$. Referring to Lemma 2.3.3 we can see that this implies that the trajectories of agents $i$ and $i + 1$ will intersect at the point $(\rho_{i,j,1}, \tau_{i,j,1}) = (\rho^0_{i,j,1}, \tau^0_{i,j,1})$. On the other hand we will have $(\rho_{i,j,k}, \tau_{i,j,k}) = (\rho^0_{i,j,k}, \tau^0_{i,j,k})$ for all $k = 2, 3, \ldots, f$. We observe that $\beta_k^o > \beta_k^+ \ldots > \beta_j^+$ implying that, for odd/even $j$, agent $i$ is further to the left/right than all agents $i + 2, i + 3, \ldots, i + f$ during the time interval $[\tau_{i-1,j,2}, \tau_{i,j,2}] = [\tau^o_{i-1,j,2}, \tau^0_{i,j,2}]$. Of course, we want to show that agent $i$ is further to the left/right (for odd/even $j$) than all other agents during the time interval $[\tau_{i-1,j,1}, \tau_{i,j,1}] = [\tau^0_{i-1,j,1}, \tau^0_{i,j,1}]$. Since agent $i + 2$ will be further to the left/right of agent $i$ after the time $\tau_{i,j,2}^0$, we need to either ensure that $\tau_{i,j,2}^o \leq \tau_{i,j,1}$, or, in the case that $\tau_{i,j,2}^o < \tau_{i,j,1}$, agent $i + 2$ must turn around at exactly the time $\tau_{i,j,2}^o$. In other words, the intersection point must coincide with a turning point. This latter condition will occur precisely when $\tau_{i,j,2}$ changes from $\tau_{i,j,2}^o$ to $\tau_{i,j,2}^+$, i.e. when $a = q(r^{1-4/n} - 1)$.

For $a > q(r^{1-4/n} - 1)$ we will need $\tau_{i,j,2}^o \geq \tau_{i,j,1}^+$. We have

$$\tau_{i,j,2}^o \geq \tau_{i,j,1}^+ \Rightarrow q(1 + r^{4/n-1}) \geq q + (q - 1)r^{2/n} \Rightarrow q \leq \frac{r}{r - r^{2/n}}.$$

This completes the proof. \hfill \Box

In the next lemma we describe how we should choose $q$ if we want the agent furthest from $d_{i,j}$ to be located on the cone $C^+_1$ at the time the $(f + 1)^{st}$ agent reaches $d_{i,j}$.

**Lemma 2.3.7.** If we take $q = \tilde{q}(r, k)$ then agents $i + k - 1$ and $i + k$, $k = f + 3, \ldots, n$, will both be located on the cone $C^+_1$ on the opposite side of the origin from $d_{i,j}$ at the time $T^0_{i,j,f+1}(1)$. 

55
\textbf{Proof.} Agents $i + k - 1$ and $i + k$ will both be located on the cone $C_i^+$ on the opposite side of the origin from $d_{i,j}$ at the time $\tau_{i+k-1,j-1,1}^+$. We thus need to solve the equation $\tau_{i+k-1,j-1,1}^+ = T^\circ_{i,j,f+1}(1)$ for $q$. We have

$$\tau_{i+k-1,j-1,1}^+ = T^\circ_{i,j,f+1}(1)$$

$$\rightarrow [q + (q-1)\frac{r^2}{n}]|d_{i+k-1,j-1}| = \left[1 + 2qr^{2(f+1)/n-1}\right]|d_{i,j}|$$

$$\rightarrow [q(1 + r^{2/n}) - r^{-2/n}]r^{2(k-1)/n-1} = 1 + 2qr^{1/n}$$

$$\rightarrow q[(1 + r^{2/n})r^{2(k-1)/n-1} - 2r^{1/n}] = r^{2k/n-1} + 1$$

and finally

$$q = \frac{r^{2k/n-1} + 1}{(1 + r^{2/n})r^{2(k-1)/n-1} - 2r^{1/n}} = \frac{r^{2k/n} + r}{(1 + r^{-2/n})r^{2k/n} - 2r^{1+1/n}} = \hat{q}(r,k).$$

We need one last lemma before proving Theorem 2.3.1.

\textbf{Lemma 2.3.8.} We have $\hat{a}(r,q) > r^{1/n}$ when $r > 1$ and $q \geq \frac{r}{r-1}$.

\textbf{Proof.} We need to consider the two cases corresponding to the definition of $\hat{a}(r,q)$. First consider the case that $q \leq \frac{r}{r-2/n}$. In this case we need to show that $q(r^{1-2/n} - 1) > r^{1/n}$. Since $q \geq \frac{r}{r-1}$ we will demonstrate the stronger condition that

$$\frac{r(r^{1-2/n} - 1)}{r - 1} > r^{1/n} \quad \rightarrow \quad r^2 > (r - 1)r^{3/n} + r^{1+2/n}$$

which is clearly satisfied for $r > 1$ and $n \geq 5$. Now consider the case that $q > \frac{r}{r-2/n}$. Then we need to show that $q(r^{1-4/n} - 1) > r^{1/n}$. Since $q > \frac{r}{r-2/n}$ we get the stronger condition

$$\frac{r(r^{1-4/n} - 1)}{r - r^{2/n}} > 1 \quad \rightarrow \quad r^2 > r + r^{1-2/n}(r^{6/n} - 1) \quad \rightarrow \quad r > 1 + r^{4/n} - \frac{1}{r^{2/n}}.$$ 

For $n \geq 5$ both sides of this inequality grow with $r > 1$, however, the left hand side grows faster. Moreover, when $r = 1$ the two sides are equal. We can thus conclude that $\hat{a}(r,q) > r^{1/n}$.

We are now ready to prove Theorem 2.3.1.
Proof. (Theorem [2.3.1]) Suppose without loss of generality that the target is at location \( x = zd_{i,j} \in I_{i,j} \). In the worst case the first \( f \) agents that reach the target are faulty and \( x \) is just beyond a turning point. For the generalized schedules we will need to consider two separate turning points.

Consider pairs \((r, u) \in \mathcal{P}\) and suppose that \( q = \hat{q}(r, u) \) and \( a = \hat{a}(r, q) \). By definition of \( \mathcal{P} \) we have \( r > 1 \) and \( \frac{r}{r-1} \leq q \leq \min\{\frac{r}{r-1-\frac{z}{2n}}, \frac{r}{r-1+\frac{z}{2n}}\} \). Then, by Lemma [2.3.8] we also have \( a > r^{1/n} \). By Lemma [2.3.4] we know that agent \( i + f + 1 \) will thus reach location \( x \) at the time

\[
T_{i,j,f+1}(z) = T_{i,j,f+1}^o(z) = (z + qr^{2(f+1)/n-1})|d_{i,j}| = (z + qr^{1/n})|d_{i,j}|,
\]

agents \( i+1, \ldots, i+f-1 \) will all reach \( x \) before agent \( i+f+1 \), and agents \( i+f+2, \ldots, i+n \) will reach \( x \) after agent \( i+f+1 \). Agent \( i+f \) will reach \( x \) before agent \( i+f+1 \) provided that \( q \leq \frac{r}{r-2n} \) or \( q > \frac{r}{r-2n} \) and \( a \geq zr^{1/n} \). We first consider the case that agent \( i+f \) reaches \( x \) before agent \( i+f+1 \).

With \( q \leq \frac{r}{r-2n} \) or \( q > \frac{r}{r-2n} \) and \( a \geq zr^{1/n} \) we know that agent \( i+f+1 \) will be the \((f+1)^{st}\) agent to reach the target. In the worst case the target is at position \( x = (1+\epsilon)d_{i,j} \), i.e. \( z = 1+\epsilon \). The search time is therefore

\[
S_f^x = T_{i,j,f+1}^o(1+\epsilon) = (1+\epsilon + 2qr^{1/n})|d_{i,j}|.
\]

Since \( q = \hat{q}(r, u) \), we know by Lemma [2.3.7] that agent \( i+u \) will be (one of) the agents most distant from \( x \) at time \( S_f^x \). Moreover, this agent will be located on the cone \( C_i^+ \) at time \( T_{i,j,f+1}^o(1) \). We can thus conclude that

\[
\Delta_f^x = |x| + \frac{T_{i,j,f+1}^o(1)}{\beta_i^+} + \epsilon|d_{i,j}|,
\]

and the evacuation time is therefore

\[
E_f^x = S_f^x + \Delta_f^x = T_{i,j,f+1}^o(1+\epsilon) + |x| + \frac{T_{i,j,f+1}^o(1)}{\beta_i^+} + \epsilon|d_{i,j}|
\]

\[
= (1 + 3\epsilon)|d_{i,j}| + \left(1 + \frac{1}{\beta_i^+}\right) T_{i,j,f+1}^o(1)
\]

\[
= (1 + 3\epsilon)|d_{i,j}| + \left(1 + \frac{1}{\beta_i^+}\right)(1 + 2qr^{1/n})|d_{i,j}|.
\]

Dividing by \( |x| = (1+\epsilon)|d_{i,j}| \) and taking the limit \( \epsilon \to 0 \) yields the competitive ratio to be

\[
R_f^A = 1 + \left(1 + \frac{1}{\beta_i^+}\right)(1 + 2qr^{1/n})
\]

57
where we have included a superscript \( A \) to indicate that this is the competitive ratio of Scenario A. We have from equation (2.17) that
\[
\beta_i^+ = \frac{q + (q - 1)r^{2/n}}{q - (q - 1)r^{2/n}}.
\]
We claim that \( q - (q - 1)r^{2/n} \geq 0 \) as a result of the requirement that \( q \leq \frac{r}{r_2} \). Indeed,
\[
q - (q - 1)r^{2/n} = q(1 - r^{2/n}) + r^{2/n} \geq 0 \quad \Rightarrow \quad q \geq \frac{r^{2/n}}{r^{2/n} - 1} = \frac{r}{r - r^{1-2/n}}.
\]
Now observe that
\[
1 + \frac{1}{\beta_i^+} = 1 + \frac{q - (q - 1)r^{2/n}}{q + (q - 1)r^{2/n}} = \frac{2q}{q + (q - 1)r^{2/n}}
\]
and we can conclude that the competitive ratio of Scenario A is
\[
R_i^A = 1 + \frac{2q(1 + 2qr^{1/n})}{q + (q - 1)r^{2/n}}
\]
as required.

Now consider the case that \( q > \frac{x}{r - r^{2/n}} \) and \( a < zr^{1/n} \). Recall that the condition \( a < zr^{1/n} \) derived from the requirement that agent \( i + f \) reaches its turning point \( d_{i+j-1}^{(1)} \) before reaching \( x \). The time at which agent \( i + f \) reaches \( x \) is \( T_i^{+}(z) \) and at this time agents \( i + 1, \ldots, i + f - 1 \) and \( i + f + 1 \) have already visited \( x \). The worst case for this scenario places the target just beyond the turning point \( d_{i+j-1}^{(1)} \) and so we take \( x = (1 + \epsilon) d_{i+j-1}^{(1)} = (1 + \epsilon)\frac{a}{r_i^{1/n}} [d_{i,j}] \). Thus, with \( z = (1 + \epsilon)\frac{a}{r_i^{1/n}} \), the search time for this case is at most \( T_i^{+}(z) \), i.e.
\[
S_f^i \leq T_i^{+}(z) = (z + 2(q - 1)r^{2f/n})[d_{i,j}] = (z + 2(q - 1)r^{1-1/n})[d_{i,j}].
\]
At the time \( T_{i,j,f+1}^{(0)}(1) \) the agent most distant from \( x \) was at distance \( |x| + T_{i,j,f+1}^{(0)}(1)/\beta_i^+ \). Thus, at time \( T_{i,j,f}^{+}(z) \) this agent will be further away from \( x \) by at most the distance \( T_{i,j,f}^{+}(z) - T_{i,j,f+1}^{(0)}(1) \). We therefore have that
\[
\Delta_f^x \leq |x| + \frac{T_{i,j,f+1}^{(0)}(1)}{\beta_i^+} + T_{i,j,f}^{+}(z) - T_{i,j,f+1}^{(0)}(1).
\]
For the evacuation time we find
\[
E^x_f = S^x_f + \Delta^x_f \leq T_{i,j,f}^+(z) + |x| + \frac{T_{i,j,f+1}^0(1)}{\beta_1^+} + T_{i,j,f}^+(z) - T_{i,j,f+1}^0(1)
\]
\[
= |x| - \left(1 - \frac{1}{\beta_1^+}\right)T_{i,j,f+1}^0(1) + 2T_{i,j,f}^+(z)
\]
\[
= z|d_{i,j}| - \left(1 - \frac{1}{\beta_1^+}\right) (1 + 2qr^{1/n})|d_{i,j}| + 2(z + 2(q - 1)r^{1-1/n})|d_{i,j}|
\]
\[
= \left[3z + 4(q - 1)r^{1-1/n} - \left(1 - \frac{1}{\beta_1^+}\right) (1 + 2qr^{1/n})\right]|d_{i,j}|.
\]

We have
\[
1 - \frac{1}{\beta_1^+} = 1 - \frac{q - (q - 1)r^{2/n}}{q + (q - 1)r^{2/n}} = \frac{2(q - 1)r^{2/n}}{q + (q - 1)r^{2/n}}.
\]
and thus
\[
E^x_f \leq \left[3z + 4(q - 1)r^{1-1/n} - \frac{2(q - 1)r^{2/n}(1 + 2qr^{1/n})}{q + (q - 1)r^{2/n}}\right]|d_{i,j}|
\]
\[
= \left[3z + \frac{2(q - 1)}{r^{1/n}} \left(2r - \frac{r^{3/n}(1 + 2qr^{1/n})}{q + (q - 1)r^{2/n}}\right)\right]|d_{i,j}|.
\]

Dividing through by \(z|d_{i,j}|\) with \(z = (1 + \epsilon)^{\frac{a}{r^{1/n}}}\) and taking the limit \(\epsilon \to 0\) yields the competitive ratio
\[
R_f^B \leq 3 + \frac{2(q - 1)}{a} \left(2r - \frac{r^{3/n}(1 + 2qr^{1/n})}{q + (q - 1)r^{2/n}}\right).
\]

Since we are assuming that \(q > \frac{r}{r - r^{2/n}}\), we have \(a = \hat{a}(r, q) = q(r^{1-4/n} - 1)\) and we can finally conclude that
\[
R_f^B \leq 3 + \frac{2(q - 1)}{q(r^{1-4/n} - 1)} \left(2r - \frac{r^{3/n}(1 + 2qr^{1/n})}{q + (q - 1)r^{2/n}}\right)
\]
as required.

Figure 2.5 illustrates the optimized trajectories of the agents for the cases \(f = 2, 3, 4\). Table 2.2 lists the (numerically) optimized competitive ratios of the generalized schedule along with the corresponding optimal pair \((r, u)\) and the resulting parameters \(a\) and \(q\). Figure 2.6 plots these competitive ratios as a function of \(f\) along with those of the optimized proportional schedule. One can observe that, in many cases, the two algorithms have identical
competitive ratios. In all cases for \( f > 1 \) the optimal competitive ratio is defined by Scenario A. It is interesting to note that in every case except \( f = 3 \) we have \( \hat{a}(r,q) = q(r^{1-2/n} - 1) \). In fact, the case \((n,f) = (7,3)\) seems to be unique in a number of ways. It has an identical competitive ratio as the \( f = 2 \) case. Moreover, the optimal parameter values for \( q \) and \( a \) are identical for \( f = 3 \) and \( f = 2 \), and, although the parameter \( r \) is not the same, one can confirm that the quantity \( r^{1/n} \) is identical in both cases.

![Figure 2.5: The optimized trajectories of the agents.](image)

### 2.3.2 Three agents, one fault

Our goal is to prove the following theorem.

**Theorem 2.3.9.** Suppose that \( r > 2\sqrt{2} \) and take

\[
a = \frac{r + 1 + r^{2/3} - \sqrt{(r + r^{2/3} + 1)^2 + 4r^{2/3}(r + 1)(r - 2r^{1/3})}}{2(r - 2r^{1/3})},
\]

(2.18)
Table 2.2: Optimal parameter choices for the generalized schedules and the corresponding competitive ratio. The competitive ratio of the optimized proportional schedules is included for comparison.

<table>
<thead>
<tr>
<th>n</th>
<th>f</th>
<th>Rprop. sched.</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>3.58545</td>
<td>5</td>
<td>1.45340</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.17225</td>
<td>1.67348</td>
<td>7.25377</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>5.97532</td>
<td>6</td>
<td>1.45340</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.25582</td>
<td>1.67348</td>
<td>7.25377</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>4.21585</td>
<td>8</td>
<td>1.38190</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.22813</td>
<td>2.84964</td>
<td>7.14702</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>3.22306</td>
<td>10</td>
<td>1.44983</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.32740</td>
<td>7.10648</td>
<td>7.10648</td>
</tr>
</tbody>
</table>

Figure 2.6: The optimized competitive ratio of the generalized schedules as a function of the number of faults.

Then the competitive ratio of the generalized schedule for \((n, f) = (3, 1)\) is

\[
R_1 = 2 + \frac{r^{1/3}(2 - r^{1/3}) + \sqrt{(r + 1 + r^{2/3})^2 + 4r^{2/3}(r + 1)(r - 2r^{1/3})}}{r + 1 - 2r^{1/3}}.
\]  

In particular, if \(r = 6.833921, a = 1.699557, \) and \(q = 1.518949\) then \(R_1 = 7.437011\).

Recall the potential worst case Scenario A for the proportional schedule – the target is placed just beyond a turning point \(d_{i,j}\) and agent \(i + 1\) (who would be the first to reach the target) is faulty. Thus, agent \(i + 2\) will be the first reliable agent to reach the target and at this time agent \(i = i + 3 \mod 3\) will still need to evacuate. The problem with the normal proportional schedule is that agent \(i\) might be relatively far from location \(d_{i,j}\) when it hears the announcement. We will thus use the extra turning points of the generalized algorithm in order to position agent \(i\) relatively close to \(d_{i,j}\) at the time agent \(i + 2\) reaches \(d_{i,j}\). In particular, if we choose \(q\) according to (2.19), then agent \(i\) will be located at its sub-turning point \(d_{i,j}^{(2)}\) at exactly the same time agent \(i + 2\) reaches location \(d_{i,j}\) (see Lemma 2.3.10). However, in fixing the value of \(q\) in this way we will introduce a new potentially worst-case
location of the target and, choosing \( a \) according to (2.18), will ensure this new potential worst case is no worse than Scenario A (see Lemma 2.3.13).

![Figure 2.7: Illustrating the two worst case scenarios for the generalized proportional schedule for \( n = 3 \) and \( f = 1 \) when the parameters \( a \) and \( q \) are chosen according to (2.18) and (2.19). Scenario A on the left occurs when the target is just beyond the turning point \( d_{i,j} \) and agent \( i + 1 \) is faulty. Scenario B on the right occurs when the target is just beyond the turning point \( d_{i+1,j-1}^{(1)} \) and agent \( i \) is faulty.]

Figure 2.7 illustrates the trajectories of the three agents when \( a \) and \( q \) are chosen according to (2.18) and (2.19). The left side of the figure displays Scenario A. One can observe that at the instant agent \( i + 2 \) reaches location \( d_{i,j} \) agent \( i \) will be at its turning point \( d_{i,j}^{(2)} \). The right side of Figure 2.7, which will be referred to as Scenario B, represents the new potentially worst case situation. In this scenario the target is placed at location \( d_{i+1,j-1}^{(1)} \) and it is agent \( i \) that is faulty. Agent \( i + 1 \) will be the first reliable agent to reach the target and agent \( i + 2 \) will be the final agent to evacuate. Observe that for this scenario the announcement by agent \( i + 1 \) will not affect the trajectory of agent \( i + 2 \) since this agent was already on its way to \( d_{i+1,j-1}^{(1)} \) at the time of the announcement. We will choose \( a \) in order to ensure that the competitive ratios resulting from Scenarios A and B will be equal.

**Scenario A:** This is the case depicted on the left of Figure 2.7. The target is just beyond the turning point \( d_{i,j} \) and agent \( i + 2 \) is the first reliable agent to reach the target. The next lemma demonstrates that if we choose \( s \) according to (2.19) then agent \( i \) will be at its turning point \( d_{i,j}^{(2)} \) at the moment agent \( i + 2 \) reaches \( d_{i,j} \).
Lemma 2.3.10. If $q$ is given by (2.19) and $a \geq \frac{1}{r^{1/3}}$ then agent $i$ will reach its turning point $d_{i,j+2}$ at the same time agent $i + 2$ reaches location $d_{i,j}$.

Proof. Agent $i$ will reach $d_{i,j}^{(2)}$ at time $t_{i,j}^{(2)}$ and agent $i + 2$ will reach location $d_{i,j}$ at the time $t_{i,j}^{(2)}(1)$. We will thus need to solve the equation $t_{i,j}^{(2)} = T_{i,j,2}(1)$ for the parameter $q$. Since $a \geq \frac{1}{r^{1/3}}$ Lemma 2.3.3 states that agent $i + 2$ will reach $d_{i,j}$ at the time $T_{i,j,2}(1) = T_{i,j,2}(1)$. We therefore need to solve

$$t_{i,j}^{(2)} = T_{i,j,2}(1) \rightarrow q(r+1) - r + a = 1 + 2qr^{2(2)/n-1} \rightarrow q(r+1 - 2r^{1/3}) = r + 1 - a$$

and finally

$$q = \frac{r + 1 - a}{r + 1 - 2r^{1/3}}$$

as required. □

We now let $R_1^A$ represent the competitive ratio for Scenario A. In the next lemma we derive an expression for this competitive ratio.

Lemma 2.3.11. Suppose that $a \geq \frac{1}{r^{1/3}}$. Then the competitive ratio of Scenario A is

$$R_1^A = 1 + \frac{t_{i,j}^{(1)} + |d_{i,j}^{(1)}|}{|d_{i,j}|} = 1 + 2(q + a).$$

Proof. With $q$ given by (2.19) we know that agent $i + 2$ will reach $d_{i,j}$ at the time $T_{i,j,2}(1) = t_{i,j}^{(2)}$ and at this time agent $i$ will be located at $d_{i,j}^{(2)}$. In the worst case the target is just beyond $d_{i,j}$ and the competitive ratio is therefore

$$R_1^A = 1 + \frac{t_{i,j}^{(2)} - |d_{i,j}^{(2)}|}{|d_{i,j}|} = 1 + \frac{t_{i,j}^{(1)} + |d_{i,j}^{(1)}|}{|d_{i,j}|}.$$ 

With $t_{i,j}^{(1)} = (2q + a)|d_{i,j}|$ and $|d_{i,j}^{(1)}| = a|d_{i,j}|$ we get

$$R_1^A = 1 + 2(q + a).$$

Scenario B: This is the case depicted on the right of Figure 2.7. The target is placed just beyond the turning point $d_{i+1,j-1}^{(1)}$, agent $i + 1$ is the first reliable agent to reach the target, and agent $i + 2$ will evacuate last. We note that for this case to occur it must be that $|d_{i+1,j-1}^{(1)}| \leq |d_{i,j}|$ which implies that $a \leq r^{1/3}$. Let $R_1^B$ represent the competitive ratio for this scenario.

63
Lemma 2.3.12. Suppose that \( a \leq r^{1/3} \). Then the competitive ratio of Scenario B is

\[
R_B^1 = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}|}{d_{i+1,j-1}^{(1)}} = 1 + \frac{2qr^{2/3}}{a}.
\]

Proof. In the worst case the target is at location \( x \) just beyond \( d_{i+1,j-1}^{(1)} \). Referring to Figure 2.7 one can observe that the announcement in this case will not change the trajectory of agent \( i+2 \) and the evacuation will complete at the time agent \( i+2 \) would normally reach \( x \). The last turning point agent \( i+2 \) visits before visiting \( x \) is the turning point \( d_{i+2,j-1}^{(1)} \) and thus the competitive ratio is

\[
R_B^1 = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}|}{d_{i+1,j-1}^{(1)}}.
\]

With \( t_{i+2,j-1} = (2q - 1)r^{1/3}|d_{i,j}|, \ |d_{i+2,j-1}| = r^{1/3}|d_{i,j}|, \) and \( |d_{i+1,j-1}^{(1)}| = ar^{-1/3}|d_{i,j}| \) we get

\[
R_B^1 = 1 + \frac{(2q - 1)r^{1/3} + r^{1/3}}{ar^{-1/3}} = 1 + \frac{2qr^{2/3}}{a}.
\]

\( \square \)

Lemma 2.3.13. Suppose that \( \frac{1}{r^{1/3}} \leq a \leq r^{1/3} \), and \( r > 2\sqrt{2} \). Then the competitive ratio of Scenarios A and B will be equal when \( q \) is given by (2.19) and \( a \) is given by (2.18).

Proof. We need to solve the equation \( R_A^1 = R_B^1 \) for \( a \). We have

\[
1 + 2(q + a) = 1 + \frac{2qr^{2/3}}{a} \quad \rightarrow \quad q(a - r^{2/3}) + a^2 = 0.
\]

Substituting in the expression (2.19) gives

\[
\left( \frac{r + 1 - a}{r + 1 - 2r^{1/3}} \right)(a - r^{2/3}) + a^2 = 0
\]

\[\rightarrow (r + 1 - a)(a - r^{2/3}) + a^2(r + 1 - 2r^{1/3}) = 0\]

and after a little more manipulation we arrive at

\[
a^2(r - 2r^{1/3}) + a(r + 1 + r^{2/3}) - r^{2/3}(r + 1) = 0.
\]

Let \( P(a) \) represent the polynomial in \( a \) on the left of this equation. There are two possible solutions to \( P(a) = 0 \):

\[
a = \frac{-(r + 1 + r^{2/3}) \pm \sqrt{(r + 1 + r^{2/3})^2 + 4r^{2/3}(r + 1)(r - 2r^{1/3})}}{2(r - 2r^{1/3})}.
\]
Note that with our assumption that \( r > 2\sqrt{2} \) the denominator of \( a \) will be positive. Clearly, then, only the positive root can result in \( a > 0 \). To show that this value of \( a \) indeed lies within the range \([\frac{1}{r^{1/3}}, r^{1/3}]\) we make use of the polynomial \( P(a) \). When \( r > 2\sqrt{2} \) the coefficient of \( a^2 \) is positive and so the parabola curves upward. Moreover, when \( a = r^{1/3} \) we have

\[
P(r^{1/3}) = r^{2/3}(r - 2r^{1/3}) + r^{1/3}(r + r^{2/3} + 1) - r^{2/3}(r + 1) \\
= r^{5/3} - 2r + r^{4/3} + r + r^{1/3} - r^{5/3} - r^{2/3} \\
= -r + r^{4/3} + r^{1/3} - r^{2/3} \\
= r(r^{1/3} - 1) - r^{1/3}(r^{1/3} - 1) \\
= (r - r^{1/3})(r^{1/3} - 1)
\]

which is clearly positive. On the other hand, when \( a = 1/r^{1/3} \) we have

\[
P(r^{-1/3}) = r^{-2/3}(r - 2r^{1/3}) + r^{-1/3}(r + r^{2/3} + 1) - r^{-2/3}(r + 1) \\
= r^{1/3} - 2r^{-1/3} + r^{2/3} + r^{1/3} + r^{-1/3} - r^{5/3} - r^{2/3} \\
= 2r^{1/3} - r^{-1/3} - r^{5/3}
\]

and it is simple to confirm that this is negative for \( r > 2\sqrt{2} \). □

To derive the expression for \( R_1 \) in Theorem 2.3.9 we substitute (2.18) and (2.19) into the expression for \( R_1^A \) in Lemma 2.3.11. We find that

\[
R_f = 1 + 2(q + a) = 1 + 2a + \frac{2(r + 1 - a)}{r + 1 - 2r^{1/3}} = 1 + 2\left(\frac{r + 1 + (r - 2r^{1/3})a}{r + 1 - 2r^{1/3}}\right) \\
= 3 + \frac{4r^{1/3} + 2(r - 2r^{1/3})a}{r + 1 - 2r^{1/3}} \\
= 3 + \frac{4r^{1/3} + 2(r - 2r^{1/3})\left(-\frac{(r+1+r^{2/3})\sqrt{(r+1+r^{2/3})^2+4r^{2/3}(r+1)(r-2r^{1/3})}}{2(r-2r^{1/3})}\right)}{r + 1 - 2r^{1/3}} \\
= 3 + \frac{4r^{1/3} - (r + 1 + r^{2/3}) + \sqrt{(r + 1 + r^{2/3})^2 + 4r^{2/3}(r + 1)(r - 2r^{1/3})}}{r + 1 - 2r^{1/3}} \\
= 3 + \frac{r^{1/3}(2 - r^{1/3}) + \sqrt{(r + 1 + r^{2/3})^2 + 4r^{2/3}(r + 1)(r - 2r^{1/3})}}{r + 1 - 2r^{1/3}}.
\]

Numerically optimizing (2.20) with respect to \( r > 2\sqrt{2} \) yields the specific results quoted in Theorem 2.3.9.
2.4 Byzantine search with one fault

In this section we use our evacuation algorithm for \((n, f) = (3, 1)\) to improve upon the best known upper bound for search by three agents, with at most one byzantine fault. For this problem we only require the agents to reliably identify the target and, in particular, they do not need to evacuate. A reliable identification of the target occurs at the first time a provably reliable agent reaches the target. We will prove the following result.

**Theorem 2.4.1.** Byzantine search for three agents, at most one of which is faulty, can be completed with a competitive ratio no more than \(7.43701137\).

We first give a quick discussion of the model. A byzantine faulty agent is similar to a crash faulty agent except that byzantine agents can also lie about finding the target, in addition to failing to announce the target. Therefore, in this case, an announcement that the target has been found cannot necessarily be trusted. In order to confirm whether or not the target is at an announced location \(x\) at least one provably reliable agent must confirm that the target is at \(x\). With \(n = 2f + 1\), it can be required that all agents reach \(x\) before this confirmation occurs (which is, of course, just evacuation in disguise). However, in this case it is possible that the agents do not find the target at \(x\) and must continue their search. Nevertheless, in this process the reliable agents will be able to identify at least one of the faulty agents (the one who lied) and thus, if the search continues, it will be from a better standpoint in terms of the ratio of faulty agents to reliable agents. In the specific case of \((n, f) = (3, 1)\), the moment the single faulty agent is identified, both of the remaining agents will know each other is reliable.

**Proof.** (Theorem 2.4.1) We base our search algorithm off of the generalized schedule of Theorem 2.3.9 in particular, we will only consider the generalized trajectory corresponding to the optimized choices of the parameters \((r, q, a)\), i.e. \(r = 6.833921\), \(a = 1.699557\), and \(q = 1.518949\). We will represent by \(\alpha\) the competitive ratio for evacuation corresponding to these parameter choices, i.e. \(\alpha = 7.43701137\).

At the beginning of our search algorithm we let the agents follow their generalized schedule trajectories until the moment an announcement is made. Suppose that this announcement claims that the target is at location \(x \in I_{i,j}\). Then the behavior of the agents will depend on whether or not \(x \in (d_{i,j}, d_{i+2,j-1}^{(1)})\) or \(x \in (d_{i+2,j-1}^{(1)}, d_{i+1,j})\). We consider first the case that \(x \in (d_{i,j}, d_{i+2,j-1}^{(1)})\).

**Case 1, \(x \in (d_{i,j}, d_{i+2,j-1}^{(1)})\):** Let us assume without loss of generality that \(x > 0\). An illustration of this case is provided in Figure 2.8 for reference. When the first announcement claims that the target is at location \(x \in (d_{i,j}, d_{i+2,j-1}^{(1)})\) all agents that have not visited \(x\) at the time of the announcement will immediately move to \(x\) to check the claim. One can observe that the agents will arrive to \(x\) in the order \(i+1, i+2, i\). If agent \(i\) is the one who made the
Figure 2.8: Setup for the case that the first announcement claims that the target is at location \( x \in (d_{i,j}, d_{i+2,j-1}) \) and the actual target is on the left side at location \( x' \). Agent \( i \) is in red, agent \( i + 1 \) in blue, and agent \( i + 2 \) in green. On the left side it is agent \( i + 1 \) that falsely announces the target and on the right side it is agent \( i + 2 \) that falsely announces the target. In either case the dashed lines show where the agents would be if the announcement were not made and the solid lines show the actual trajectories of the agents.

announcement then agents \( i + 1 \) and \( i + 2 \), having already visited \( x \), will immediately know that agent \( i \) is faulty. They will thus immediately proceed to move in opposite directions at full speed until the target is found. This will clearly not be a worst case and so we will suppose that it is one of agents \( i + 1 \) or \( i + 2 \) that made the announcement.

In this case the first reliable agent (either \( i + 1 \) or \( i + 2 \)) that reaches \( x \) will immediately move to the left after reaching \( x \) and continue in this direction until the real target is found. Agent \( i \), who will be moving to the right when it reaches \( x \), will continue moving to the right after \( x \) until either it, or the other reliable agent, finds the real target. It is clear from the results on evacuation that agent \( i \) will be able to find a target at any position \( x' \) to the right of \( x \) in time less than \( \alpha x' \). Thus, we can assume that the target is on the left side of the origin.

Assume first that it is agent \( i + 1 \) that made the erroneous announcement. This situation is depicted on the left side of Figure 2.8. Since agent \( i + 2 \) will reach \( x \) after its turning point \( d_{i+2,j-1} \) and since \( x \leq d_{i+2,j-1}^{(1)} \) the exact time agent \( i + 2 \) will reach \( x \) is \( t_{i+2,j-1} + |d_{i+2,j-1}| + x \). Moreover, the real target must be somewhere to the left of \( d_{i+2,j-1} \), since, otherwise, agent \( i + 2 \) would have announced it already. Thus, the competitive ratio will be

\[
R = 1 + \sup \frac{t_{i+2,j-1} + |d_{i+2,j-1}| + 2x}{|x'|}
\]
where the supremum is over \( x \in (d_{i,j}, d_{i+2,j-1}) \) and \( x' < d_{i+2,j-1} \). Since the ratio increases with \( x \) and decreases with \( x' \) we have

\[
R = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}|}{d_{i+2,j-1}} = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}|}{d_{i+2,j-1}} 
\]

This ratio is independent of \( i \) and \( j \) and so we also have

\[
R = 1 + \frac{t_{i,j} + |d_{i,j}|}{d_{i,j}}. 
\]

This, however, is exactly the expression for \( R^A \) in Lemma 2.3.11 which equals \( \alpha \).

Now consider the situation depicted on the right side of Figure 2.8 where agent \( i+2 \) makes the announcement. Then agent \( i+1 \) will have already visited \( x \) when the announcement is made and will immediately proceed to move left until it finds the target at \( x' \).

Referring to Figure 2.8 one can observe that at the time of the announcement the leftmost point visited by a reliable agent will be the point \( d_{i,j} \) and we must therefore have \( x' < d_{i,j} \).

One can also observe that agent \( i+1 \) will reach \( x' \) at the time \( t_{i+1,j} + |d_{i+1,j}| + x' \) and so the competitive ratio in this situation will be

\[
R = 1 + \frac{t_{i+1,j} + |d_{i+1,j}|}{d_{i+1,j}} = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}|}{d_{i+2,j-1}} 
\]

and this is identical to the expression for \( R^B \) in Lemma 2.3.12. We can thus conclude that the competitive ratio of the search is \( \alpha \) when \( x \in (d_{i,j}, d_{i+2,j-1}) \).

**Case 2, \( x \in (d_{i+2,j-1}, d_{i+1,j}] \):** This situation is depicted in Figure 2.9. As before, we assume that \( x > 0 \) and, since agent \( i \) will be the last agent to reach \( x \), we may assume that it is not agent \( i \) that announces the target. Then, we will first send agent \( i \) to its turning point \( d_{i,j+1} \) before having it move to check the claim at \( x \). The remaining reliable agent will move to the left after it visits \( x \). If agent \( i \) finds the target before reaching \( x \) it will announce this and the search will end once the remaining reliable agent visits \( x \). This will not be a worst case and so we assume that agent \( i \) does not find the target before reaching \( x \).

Suppose that it is agent \( i+1 \) that announces the target and refer to the left side of Figure 2.9. If the target is at location \( x' \) to the right of \( x \) then agent \( i \) will reach the target after it visits its turning point \( d_{i,j+1} \) and so the time at which agent \( i \) reaches \( x' \) is \( t_{i,j+1} + |d_{i,j+1}| + x' \) and the competitive ratio will be

\[
R = 1 + \sup_{x'} \frac{t_{i,j+1} + |d_{i,j+1}|}{x'} = 1 + \frac{t_{i,j+1} + |d_{i,j+1}|}{d_{i+2,j-1}} = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}|}{d_{i+1,j-1}} 
\]
Figure 2.9: Setup for the case that the first announcement claims that the
target is at location $x \in (d_{i+2,j-1}^{(1)}, d_{i+1,j}]$ and the actual target is on the left
side at location $x'$. Agent $i$ is in red, agent $i + 1$ in blue, and agent $i + 2$ in
green. On the left side it is agent $i + 1$ that falsely announces the target and
on the right side it is agent $i + 2$ that falsely announces the target. In either
case the dashed lines show where the agents would be if the announcement
were not made and the solid lines show the actual trajectories of the agents.

where we have used our assumption that $x' > x > d_{i+2,j-1}^{(1)}$ and the fact that the ratio is
independent of $i$ and $j$. This expression is, of course, the same as $R_i^B$ from Lemma 2.3.12
and so in this case the competitive ratio will be at most $\alpha$. Thus, we consider the case
that the target is at location $x'$ to the left of the origin. In this case, the target must be
at location $x' < d_{i,j+1}$ since otherwise agent $i$ would have already found it. Agent $i + 2$
will receive the announcement as it is moving away from $d_{i+2,j-1}$ and so it will reach $x$ at
the time $t_{i+2,j-1} + |d_{i+2,j-1}| + x$. The earliest it could reach the target at $x'$ will then be
$t_{i+2,j-1} + |d_{i+2,j-1}| + 2x + |x'|$. The competitive ratio is then

$$R = 1 + \sup \frac{t_{i+2,j-1} + |d_{i+2,j-1}| + 2x}{|x'|}$$
where the supremum is over $x \in (d_{i+2,j-1}^{(1)}, d_{i+1,j}]$ and $x' < d_{i,j+1}$. We thus find that

$$R = 1 + \frac{t_{i+2,j-1} + |d_{i+2,j-1}| + 2|d_{i+1,j}|}{|d_{i,j+1}|}$$

$$= 1 + \frac{r^{1/3}(2q - 1)|d_{i,j}| + r^{1/3}|d_{i,j}| + 2r^{2/3}|d_{i,j}|}{r|d_{i,j}|}$$

$$= 1 + \frac{2(q + r^{1/3})}{r^{2/3}}.$$

Substituting in the values $r = 6.833921$, and $q = 1.518949$ then yields $R = 2.897498 < \alpha$. Thus, when agent $i + 1$ announces the target the competitive ratio is at most $\alpha$.

Now suppose that it is agent $i + 2$ that announces the target. Then we have the situation depicted on the right of Figure 2.9. One can easily observe from this figure that this situation cannot be worse than the case when agent $i + 1$ announces the target. Indeed, agent $i$ will reach a target $x' > x$ at the same time for either case, and agent $i + 1$ will reach a target $x' < d_{i,j+1}$ earlier than agent $i + 2$ would in the situation that $i + 1$ was the announcing agent. Thus, the competitive ratio will be at most $\alpha$ when $x \in (d_{i+2,j-1}^{(1)}, d_{i+1,j}]$. This completes the proof.

2.5 Conclusions

We have studied the problem of evacuation by $n = 2f + 1$ agents when at most $f$ of the agents are crash faulty. We introduced a novel type of search algorithm which gives an improvement for the evacuation as compared to the trajectories used for optimal crash search. These trajectories also gave an improvement on the best known upper bound for the problem of search by three agents at most one of which is byzantine faulty.

Since we did not carry out a pure optimization of the parameters for our new algorithm it is possible that there are better choices of these parameters than the ones described here. Furthermore, it is possible to further generalize the trajectories of the agents with the addition of more sub-turning points, however, this seems rather unlikely to improve the results. It is also an open problem to improve lower bounds for the evacuation problem since the state of the art derives from the optimality of the search problem, i.e. evacuation has at least the same competitive ratio as search since the agents must first find the target in order to evacuate. As a result, there is a rather large gap between the upper bounds presented here and the best known lower bounds. The complexity of the trajectories described here hints that achieving a tight lower bound for this problem may be a difficult task indeed.
References


Chapter 3

Time-Energy Tradeoffs for Evacuation by Two Robots in the Wireless Model

This chapter is the first of two concerning evacuation on the line when it is desired to minimize the energy consumed during the search, in addition to the time. Included is the paper “Time-Energy Tradeoffs for Evacuation by Two Robots in the Wireless Model” which appears in Theoretical Computer Science [62]). This is the journal version of a conference paper first appearing in the proceedings of SIROCCO2019. This chapter specifically concerns the case when the agents can communicate wirelessly.

The novel idea here is to consider strategies which minimize the energy consumed by the searchers during the evacuation procedure. Here the energy consumed by a searcher travelling distance $x$ at speed $s$ is defined to be $xs^2$. The dependence of the energy consumed on the speed-squared is well motivated by the phenomenon of viscous drag in fluid dynamics. The study of the problem in general has clear motivations arising from the physical implementation of mobile robots with an internal energy supply.
Time-Energy Tradeoffs for Evacuation by Two Robots in the Wireless Model

Jurek Czyzowicz\textsuperscript{1}, Konstantinos Georgiou\textsuperscript{2}, Ryan Killick\textsuperscript{3}, Evangelos Kranakis\textsuperscript{3}, Danny Krizanc\textsuperscript{4}, Manuel Lafond\textsuperscript{5}, Lata Narayanan\textsuperscript{6}, Jaroslav Opatrny\textsuperscript{6}, and Sunil Shende\textsuperscript{7}

\textsuperscript{1}Université du Quebéc en Outaouais, Gatineau, Québec, Canada
\textsuperscript{2}Ryerson University, Dept. of Mathematics, Toronto, Ontario, Canada
\textsuperscript{3}School of Computer Science, Carleton University, Ottawa, Ontario, Canada
\textsuperscript{4}Department of Mathematics \& Comp. Sci., Wesleyan University, Middletown, CT, USA
\textsuperscript{5}Department of Computer Science, Université de Sherbrooke, Sherbrooke, Quebec, Canada
\textsuperscript{6}Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada
\textsuperscript{7}Department of Computer Science, Rutgers University, Camden, NJ, USA

Abstract

Two robots stand at the origin of the infinite line and are tasked with searching collaboratively for an exit at an unknown location on the line. They can travel at maximum speed $b$ and can change speed or direction at any time. The two robots can communicate with each other at any distance and at any time. The task is completed when the last robot arrives at the exit and evacuates. We study time-energy tradeoffs for the above evacuation problem. The evacuation time is the time it takes the last robot to reach the exit. The energy it takes for a robot to travel a distance $x$ at speed $s$ is measured as $xs^2$. The total and makespan evacuation energies are respectively the sum and maximum of the energy consumption of the two robots while executing the evacuation algorithm.

Assuming that the maximum speed is $b$, and the evacuation time is at most $cd$, where $d$ is the distance of the exit from the origin and $c$ is some positive real number, we study the problem of minimizing the total energy consumption of the robots. We prove that the problem is solvable only for $bc \geq 3$. For the case $bc = 3$, we give an optimal algorithm, and give upper bounds on the energy for the case $bc > 3$.

We also consider the problem of minimizing the evacuation time when the available energy is bounded by $\Delta$. Surprisingly, when $\Delta$ is a constant, independent of the distance $d$ of the exit from the origin, we prove that evacuation is possible in time $O(d^{3/2} \log d)$, and this is optimal up to a logarithmic factor. When $\Delta$ is linear in $d$, we give upper bounds on the evacuation time.
3.1 Introduction

Linear search is an online problem in which a robot is tasked with finding an exit placed at an unknown location on an infinite line. It has long been known that the classic doubling strategy, which guarantees a search time of $9d$ for an exit at distance $d$ from the initial location is optimal for a robot travelling at speed at most 1 (see any of the books [2, 8, 155] for additional variants, details and information). If even one more robot is allotted to the search then clearly an exit at distance $d$ can always be found in time $d$ by one of the robots. Therefore the problem of group search by multiple robots on the line is concerned with minimizing the time the last robot arrives at the exit; the problem is also called evacuation. It was first introduced as part of a study on cycle-search [57] and further elaborated on an infinite line for multiple communicating robots with crash [77] and Byzantine faults [70].

The time taken for group search on the line clearly depends on the communication capabilities of the robots. In the wireless communication model, the robots can communicate at any time and over any distance. In the face-to-face communication model, the robots can only communicate when they are in the same place at the same time. A straightforward algorithm achieves evacuation time $3d$ in the wireless model, and can be seen to be optimal, while it has been shown that in the face-to-face model, two robots cannot achieve better evacuation time than one robot [40].

In this paper, we consider the energy required for group search on the line. We use the energy model in which the energy consumption of a robot travelling a distance $x$ at speed $s$ is proportional to $xs^2$. This model, also used in [61], is motivated by the concept of viscous drag in fluid dynamics; see Section 3.1.1 for more details. Let $c$ be a chosen positive constant. Then, the authors of [61], studied the question of the minimum energy required for group search on the line by two robots performing under the face-to-face communication model and travelling at speed at most $b$ while guaranteeing that both robots reach the exit within time $cd$, where $d$ is the distance of the exit from the starting position of the robots. For the special case $b = 1, c = 9$, they proved the surprising result that two robots can evacuate with less energy than one robot, while taking the same evacuation time. Note that in the face-to-face model, where no information can be exchanged from distance, robots are bound to follow zig-zag trajectories effectively arranging for a sequence of possible meeting points in order to facilitate the efficient evacuation of their peers. In contrast, robots’ trajectories we consider in this work, where robots perform in the wireless model, need only make one turn before they terminate. As a result, our model of energy that does not take into consideration any turning costs can be thought as more realistic in the wireless communication model than in the face-to-face communication model.

Our main approach throughout the paper is to investigate time-energy tradeoffs for group search by two robots in the wireless communication model. Assuming that the maximum speed is $b$, and the evacuation time is at most $cd$, where $d$ is the distance of the exit from the origin, we study the problem of minimizing the total energy consumption of the robots. We
also consider the problem of minimizing the evacuation time when the available energy is bounded by $\Delta$.

### 3.1.1 Model and problem definitions

Two robots are placed at the origin of an infinite line. An exit is located at unknown distance $d$ from the origin and can be found if and only if a robot walks over it. A robot can change its direction or speed at any time, e.g., as a function of its distance from the origin, or the distance walked so far. Robots operate under the wireless model of communication in which messages can be transmitted between robots instantaneously at any distance. Feasible solutions are robots’ trajectories in which, eventually, both robots evacuate, i.e., they both reach the exit. Given a location of the exit, the time by which the second robot reaches the exit is referred to as the *evacuation time*. We distinguish between constant-memory robots that can only travel at a constant number of hard-wired speeds, and unbounded-memory robots that can dynamically compute speeds and distances, and travel at any possible speed.

The energy model being used throughout the paper is motivated from the concept of viscous drag in fluid dynamics [20]. In particular, an object moving with constant speed $s$ will experience a drag force $F_D$ proportional to $s^2$. In order to maintain the speed $s$ over a distance $x$ the object must do work equal to the product of $F_D$ and $x$ resulting in a continuous energy loss proportional to the product of the object’s squared speed and travel distance. For simplicity we take the proportionality constant to be one, and define the energy consumption moving at constant speed $s$ over a segment of length $x$ to be $xs^2$. We extend the definition of energy for a robot moving in the same direction from point $a$ to point $b$ on the line, using speed $s(x) \in \mathbb{R}, x \in [a,b]$, as $\int_{a}^{b} s^2(x)dx$. The total energy of a specific robot traversing more intervals, possibly in different directions, is defined as the sum of the energies used in each interval.

Given a team of two robots, the *total evacuation energy* is defined as the sum of the robots’ energies used till both robots evacuate. Similarly, we define the *makespan evacuation energy* as the maximum energy used by any of the two robots.

For each $d > 0$ there are two possible locations for the exit to be at distance $d$ from the origin: we will refer to either of these as input instances $d$ for the group search problem. More specifically, we are interested in the following three optimization problems:

**Definition 3.1.1.** Problem EE$_d$($c$): Minimize the total evacuation energy, given that the evacuation time is no more than $cd$ (for all instances $d$) and using speeds no more than $b$.

**Definition 3.1.2.** Problem TE$_d$($\Delta$): Minimize the evacuation time, given that the total evacuation energy is no more than $\Delta$ (for all instances $d$), and using speeds at most $b$.

---

1 The constant of proportionality has (SI) units $kg/m$ and depends, among other things, on the shape of the object and the density of the fluid through which it moves.
Definition 3.1.3. Problem $\text{ME}_d^b(\Delta)$: Minimize the evacuation time, given that the make-span evacuation energy is no more than $\Delta$ (for all instances $d$), and using speeds at most $b$.

For the last two problems, we consider two cases when the evacuation energy $\Delta$ is a constant and when it is linear in $d$.

3.1.2 Our results

Consider the following intuitive and simple algorithm for wireless evacuation, which is a parametrized version of a well-known algorithm for the case of unit speed robots that achieve evacuation time $3d$.

Definition 3.1.4 (Algorithm Simple Wireless Search $\mathcal{N}_{s,r}$). Robots move at opposite directions with (possibly non-constant) speed $s$ until the exit is found. The finder announces “exit found” and halts. The other robot changes direction and moves at (possibly non-constant) speed $r$ until the exit is reached.

We analyze the behaviour of this algorithm for all three proposed problems, and determine the speeds that achieve the minimum evacuation energy (or time) among all algorithms of this class, while respecting the given bound on evacuation time (resp. energy). In some cases, the algorithms derived are shown to be optimal. In particular, our main results are the following:

1. We show that the problem $\text{EE}_d^b(c)$ admits a solution if and only if $cb \geq 3$. Furthermore, for every $c, b > 0$ with $cb = 3$, we show that the optimal total evacuation energy is $4b^2d$, and this is achieved by $\mathcal{N}_{s,r}$ with $s = r = b$ (Theorem 3.2.3).

2. For every $c, b > 0$ with $cb \geq 3$, we derive the optimal values of $s$ and $r$ for the algorithm $\mathcal{N}_{s,r}$ that minimize the total evacuation energy (Theorem 3.2.4).

3. We observe that if total or makespan energy $\Delta$ is a constant, problems $\text{TE}_d^b(\Delta)$ and $\text{ME}_d^b(\Delta)$ cannot be solved by robots that can only use a finite number of speeds. We prove that if $\Delta$ is bounded by a constant, the optimal evacuation time is $\Omega(d^{3/2})$ (see Theorem 3.3.2). Somewhat surprisingly, we give an algorithm with evacuation time $O(d^{3/2} \log d)$ (see Theorem 3.3.3); thus the algorithm is optimal up to a logarithmic factor. Our algorithm requires the robots to continuously change their speed at every distance $x$ from the origin. This is the only part that requires robots to have unbounded memory.

4. For the problems $\text{TE}_d^b(\Delta)$ and $\text{ME}_d^b(\Delta)$ with total or makespan energy $\Delta = O(d)$ and $b = 1$, we give upper bounds on the evacuation time (see Theorems 3.3.3 and 3.5.1 respectively).
3.1.3 Related Work

In group search, a set of communicating robots interact and co-operate by exchanging information in order to complete the task which usually involves finding an exit placed at an unknown location within a given search domain. Some of the pioneering results related to our work are concerned with search on an infinite domain, like a straight line [13, 22, 25, 128], while others with search on the perimeter of a closed domain like unit disk [57] or equilateral triangle or square [79]. The communication model being used may be either wireless [57] or face-to-face [36, 72, 79]. Search and evacuation problems with a combinatorial flavour have been recently considered in [63, 64] and search-and-fetch problems in [112, 111], while [43] studied average-case/worst-case trade-offs for a specific evacuation problem on the disk. The interested reader may also wish to consult a recent survey paper [68] on selected search and evacuation topics.

Traditional approaches to evaluating the performance of search have been mostly concerned with time. This is apparent in the book [8] and the research described in the seminal works on deterministic [13], stochastic [22, 25] and randomized [128] search and continued up to the most recent research papers on linear search, for robots with terrain dependent speeds [78], robots with faults [113] in particular Byzantine [70] or crash-fault behaviour [77], robots with probabilistic faults [30], and in group search problems with distinguished searchers, such as in [114] for linear search, or [64, 65] for searching the unit disk (see also the survey paper [68]). Aside from the research by [84], in which the authors are looking at the turn cost when robots change direction during the search, little or no research has been conducted on other measures of performance.

[61] was the first paper on search and evacuation that changed the focus from optimizing the time to (a) minimizing the energy consumption required to find the exit and (b) time/energy tradeoffs. The authors determine optimal (and in some cases nearly optimal) linear search algorithms inducing the lowest possible energy consumption and also propose a linear search algorithm that simultaneously achieves search time $9d$ and consumes energy $8.42588d$, for an exit located at distance $d$ unknown to the robots. However, this differs from our present work in that the authors focus exclusively on the face-to-face communication model while here we focus on the wireless model. In the present paper, we extend the results of [61] to the realm of the wireless communication model and study time/energy trade-offs for evacuating two robots on the infinite line. Despite their apparent similarities, the face-to-face and wireless communication models require completely different approaches to the design of efficient linear search algorithms.
3.2 Minimizing Energy Given Bounds on Evacuation Time and Speed

This section is devoted to the problem EE\textsubscript{\textit{b}}\textsubscript{\textit{d}}\textsubscript{\textit{c}} of minimizing the total evacuation energy, given that the robots can travel at speed at most \textit{b} and are required to complete the evacuation within time \textit{cd} for every instance \textit{d} where \textit{d} is the distance of the exit from the origin. We start with establishing a necessary condition on the product \textit{bc}.

**Lemma 3.2.1.** No online (wireless) algorithm can solve EE\textsubscript{\textit{b}}\textsubscript{\textit{d}}\textsubscript{\textit{c}} if \textit{bc} < 3.

**Proof.** (Lemma 3.2.1) Consider an arbitrary algorithm for the problem, and for some \textit{d} > 0, we let it run till at least one point among \pm \textit{d} is reached by a robot, say \textit{R}, in time at least \textit{d}/\textit{b}. Suppose that location +\textit{d} is reached first by \textit{R}. We place the exit at point −\textit{d} − \epsilon, so that \textit{R} needs additional \((2\textit{d} + \epsilon)/\textit{b}\) time to reach it, resulting in evacuation time at least \((3\textit{d} + \epsilon)/\textit{b}\). Since the argument holds for every \epsilon > 0, the claim follows.

Next we show that algorithm \textit{N}_{\textit{b},\textit{b}} is an optimal solution to the problem EE\textsubscript{\textit{b}}\textsubscript{\textit{d}}\textsubscript{\textit{c}} when \textit{bc} = 3. We start with the following lemma:

**Lemma 3.2.2.** Let \textit{b}, \textit{c} > 0 with \textit{bc} = 3 and consider an evacuation algorithm such that robots use maximum speed \textit{b} and evacuate by time \textit{cd} for an exit at distance \textit{d} from the origin. Then for every \textit{d} > 0, the points \pm \textit{d} must be visited at time \textit{d}/\textit{b}.

**Proof.** (Lemma 3.2.2) Suppose not. Notice that the points \pm \textit{d} cannot be visited before time \textit{d}/\textit{b} using speed at most \textit{b}. We look at two cases.

**Case 1:** There exists \textit{d} > 0 such that neither \textit{d} nor −\textit{d} is visited at time \textit{d}/\textit{b}. Consider the first time \textit{t} > \textit{d}/\textit{b} when either of them is visited, wlog let the point +\textit{d} be visited at time \textit{t} > \textit{d}/\textit{b} by robot \textit{R}\textsubscript{1}. We put the exit at −\textit{d}. Then \textit{R}\textsubscript{1} has to travel an additional distance of 2\textit{d}, and can use speed at most \textit{b}, so needs time at least 2\textit{d}/\textit{b} to get to the exit. The total time taken by \textit{R}\textsubscript{1} to evacuate is at least \textit{t} + 2\textit{d}/\textit{b} > 3\textit{d}/\textit{b} = \textit{cd}.

**Case 2:** There exists \textit{d} > 0 such that \textit{d} is visited at time \textit{d}/\textit{b} but −\textit{d} is not visited at this time (or vice versa). Wlog suppose \textit{R}\textsubscript{1} is at point \textit{d} at time \textit{d}/\textit{b}. Let −\textit{d} + 2\epsilon be the closest point to −\textit{d} that \textit{has} been visited at time \textit{d}/\textit{b} where \epsilon > 0 since by assumption −\textit{d} is not visited at this time. We put the exit at −\textit{d} + \epsilon. The time limit to evacuate is \textit{c}(\textit{d} − \epsilon). At time \textit{d}/\textit{b}, \textit{R}\textsubscript{1} is at distance 2\textit{d} − \epsilon from the exit, so the total time for \textit{R}\textsubscript{1} to reach the exit is at least

\[
\textit{d}/\textit{b} + (2\textit{d} − \epsilon)/\textit{b} = 3\textit{d}/\textit{b} − \epsilon/\textit{b} = \textit{cd} − \frac{\textit{c}\epsilon}{3} > \textit{cd} − \textit{c}\epsilon
\]
In both cases, we showed that the robots cannot evacuate in the required time bound. This completes the proof by contradiction.

**Theorem 3.2.3.** For every $b, c > 0$ with $bc = 3$, the algorithm $N_{b,b}$ is the only feasible solution to $EE_d^b(c)$, and is therefore optimal, and has total energy consumption $4b^2d$.

**Proof.** (Theorem 3.2.3) Lemma 3.2.2 implies that in order to achieve an evacuation time $cd$, both robots must use the maximum speed $b$ and explore in different directions. If the exit is found at distance $d$ by one of the robots, the time is $d/b$, and therefore, the other robot must travel at the maximum speed $b$ in order to arrive at the exit in time $cd$. Thus, the only algorithm that can evacuate within time $cd$ while using speed at most $b$ is $N_{b,b}$. A total distance of $4d$ is travelled by the two robots, all at speed $b$, therefore the total energy consumed is $4b^2d$.

Next we consider the case of $c, b > 3$ and determine the optimal choices of speeds $s, r$ for $N_{s,r}$, as well as the induced total evacuation energy and competitive ratio for problem $EE_d^b(c)$.

**Theorem 3.2.4.** Let $\delta = 2 + \sqrt{2} \approx 3.25992$. For every $c, b > 0$, problem $EE_d^b(c)$ admits a solution by algorithm $N_{s,r}$ if and only if $cb \geq 3$. For the spectrum of $c, b$ for which a solution exists, the following choices of speeds $s, r$ are feasible and optimal for $N_{s,r}$

<table>
<thead>
<tr>
<th>$s$</th>
<th>$3 \leq cb \leq \delta$</th>
<th>$cb &gt; \delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>$\frac{b}{bc-2}$</td>
<td>$\frac{\delta}{c}$</td>
</tr>
<tr>
<td>$r$</td>
<td>$b$</td>
<td>$\frac{\delta}{\sqrt{2c}}$</td>
</tr>
</tbody>
</table>

The induced total evacuation energy is $f(cb)\frac{2d}{c^2}$, where

$$f(x) := \begin{cases} \frac{x^2}{(x-2)^2} + x^2 & , 3 \leq x \leq \delta \\ \frac{1}{2} \left(2 + \sqrt{2}\right)^3 & , x > \delta \end{cases}$$

It was observed in [61] that the optimal offline solution, given that $d$ is known, equals $\frac{2d}{c^2}$. The competitive ratio is given by $\sup_d \frac{e(c,b,d)}{2d} f(cb)$ for algorithms inducing total evacuation energy $e(c,b,d)$. The competitive ratio of $N_{s,r}$ for the choices of Theorem 3.2.4 is summarized in Figure 3.1. Note that in particular, Theorem 3.2.4 claims that the competitive ratio only depends on the product $cb$, and when $cb = 3$, the competitive ratio is 18 and is decreasing in $cb$ (strictly only when $cb < \delta$). The optimal speed choices for the unbounded problem $EE_d^c(\infty)$ are exactly those that appear under case $cb > \delta$. The remaining of the section is devoted to proving Theorem 3.2.4.

First we derive closed formulas for the performance of $N_{s,r}$. From the definition of energy used, and given that the robots move at speed 1, we deduce what the evacuation time
and energy are when the exit is placed at distance $d$ from the origin. The following two functions will be invoked throughout our argument below.

$$T(s, r) := \frac{1}{s} + \frac{2}{r} \tag{3.1}$$

$$E(s, r) := s^2 + r^2 \tag{3.2}$$

**Lemma 3.2.5.** Let $b, c$ be such that there exist $s, r$ for which $N_{s,r}$ is feasible. Then, for instance $d$ of $EE^b_d(c)$, the induced evacuation time of $N_{s,r}$ is $d \cdot T(s, r)$ and the induced total evacuation energy is $2d \cdot E(s, r)$.

Next we show the spectrum of $c, b$ for which $N_{s,r}$ is applicable.

**Lemma 3.2.6.** Algorithm $N_{s,r}$ gives rise to a feasible solution to problem $EE^b_d(c)$ if and only if $bc \geq 3$. For every such $b, c > 0$, the optimal choices of $N^T_{s,r}$ can be obtained by solving the convex program:

$$\min_{s, r \in \mathbb{R}} E(s, r) \quad \text{(NLP)}$$

s.t. $T(s, r) \leq c$

$0 \leq s, r \leq b$.

Moreover, if $E(s_0, r_0)$ optimizes NLP$_c^b$, then the competitive ratio of $N_{s_0,r_0}$ equals $c^2 \cdot E(s_0, r_0)$.

**Proof.** (Lemma 3.2.6) For fixed parameter $b$, consider the non-linear program

$$\min_{s, r \in \mathbb{R}} \frac{1}{s} + \frac{2}{r} \quad \text{(3.3)}$$

s.t. $0 \leq s, r \leq b$
\( T(s, r) = \frac{1}{s} + \frac{2}{r} \) is clearly strictly decreasing in either of \( s, r > 0 \). Hence when \( b \) is fixed, both constraints \( s, r \leq b \) must be tight at optimality. But then, \( \min_{s, r \in \mathbb{R}} T(s, r) = T(b, b) = \frac{3}{b} \). It follows that NLP\(_b^c\) has a feasible solution if and only if \( 3/b \leq c \).

By Lemma 3.2.5, it is immediate that NLP\(_b^c\) exactly models the problem of choosing optimal speeds for \( \mathcal{N}_{s, r} \), for problem EE\(_b^c\). Also note that \( T(s, r) \) and \( E(s, r) \) are strictly convex functions when \( s, r > 0 \), and hence NLP\(_b^c\) is a convex program. Moreover, an optimizer always exists, since the function is bounded from below, and is defined over a compact feasible region. Finally, the claim pertaining to the competitive ratio follows from Lemma 3.2.5.

A corollary of Lemma 3.2.6 is that any feasible solution for NLP\(_b^c\) satisfying first-order necessary optimality conditions is also a globally optimal solution. As a result, the proof of Theorem 3.2.4 follows by showing the proposed solution is feasible and satisfies first-order necessary optimality conditions. This is done in Lemmas 3.2.7 and 3.2.8.

Towards proving that first-order optimality conditions are satisfied, we argue first that for all \( c, b > 0 \) with \( cb \geq 3 \), the optimizers of NLP\(_b^c\) satisfy the time constraint tightly. Indeed, if not, then one could reduce any of the values among \( s, r \) to make the constraint tight, improving the induced energy. Hence, in the optimal solutions to NLP\(_b^c\), any of \( s, r \leq b \) could be additionally tight or not. In what follows, \( \delta \) represents \( 2 + \sqrt{2} \), as in the statement of Theorem 3.2.4.

**Lemma 3.2.7.** For each \( c, b > 0 \) for which \( 3 \leq cb \leq \delta \), the optimal solution to NLP\(_b^c\) is given by \( s = \frac{b}{bc - 2} \), \( r = b \).

**Proof.** (Lemma 3.2.7) First-order optimality (KKT) conditions for \( s, r \), assuming that time constraint and \( r \leq b \) are tight, are

\[
-\nabla E(s, r) = \lambda_1 \nabla T(s, r) + \lambda_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}
\]

\[
T(s, r) = c
\]

\[
0 \leq s \leq b
\]

\[
r = b
\]

\[
\lambda_1, \lambda_2 \geq 0,
\]

where functions \( E(\cdot, \cdot), T(\cdot, \cdot) \) are as in (3.1), (3.2). Utilizing only the equality constraints above, we easily derive that

\[
s = \frac{b}{bc - 2}, \quad \lambda_1 = \frac{2b^3}{(bc - 2)^3}, \quad \lambda_2 = -\frac{2(b^4c^3 - 6b^3c^2 + 12b^2c - 10b)}{(bc - 2)^3}
\]

Note that \( s \leq b \) for all \( cb \geq 3 \). \( \lambda_1 \) is clearly positive. It is enough to verify that \( \lambda_2 \geq 0 \).
Indeed, define $g(x) = 10 - 12x + 6x^2 - x^3$, and note that $\lambda_2 = b\frac{g(cb)}{(cb-2)^3}$. Since $cb \geq 3$ and $b > 0$, we conclude that $\lambda_2 \geq 0$ as long as $g(cb) \geq 0$. For that, we calculate the 3 roots of $g$

$$2 + \sqrt[3]{2}, \ 2 - \frac{1 \pm i\sqrt{3}}{2^{2/3}}.$$ 

Since the leading coefficient of $g$ is negative, and since $g$ has a unique real root, we conclude that $g(x) \geq 0$ as long as $x \leq 2 + \sqrt[3]{2}$ as wanted.

Lemma 3.2.8. For each $c, b > 0$ for which $cb > \delta$, the optimal solution to NLP$^b_c$ is given by $s = \frac{\delta}{\sqrt{2}c}, r = \frac{\delta}{c}$.

Proof. (Lemma 3.2.8) first-order optimality (KKT) conditions for $s, r$, assuming that only time constraint is tight, are

$$-\nabla E(s, r) = \lambda \nabla T(s, r)$$

$$T(s, r) = c$$

$$0 \leq s, r \leq b$$

$$\lambda \geq 0$$

Using only the equality constraints, we derive

$$s = \frac{1 + 2^{2/3}}{c}, \ r = \frac{1}{2}s(cs - 1)^{2} = \frac{2 + \sqrt[3]{2}}{c}, \ \lambda = 2s^3.$$ 

Observe that the proposed values of $s, r$ satisfy the speed bound only if $cb \geq \delta$. But then, we also see that $\lambda > 0$ for all such $c, b$, and hence $s, r$ do indeed satisfy the first-order optimality conditions.

Proof. (Theorem 3.2.4) By Lemma 3.2.6 and Lemma 3.2.7 the optimal induced energy when $3 \leq cb \leq \delta$ is

$$2dE\left(\frac{b}{bc - 2}, b\right) = 2d\left(\frac{b^2}{(bc - 2)^2} + b^2\right)$$

and the induced competitive ratio is

$$(cb)^2 \left(1 + \frac{1}{(cb - 2)^2}\right).$$

Finally, by Lemma 3.2.6 and Lemma 3.2.8 the optimal induced energy when $cb > \delta$ is

$$2dE\left(\frac{1 + 2^{2/3}}{c}, \frac{2 + \sqrt[3]{2}}{c}\right) = d\left(\frac{2 + \sqrt[3]{2}}{c^2}\right)^3.$$ 

Hence the competitive ratio is constant and equals

$$\frac{1}{2} \left(2 + \sqrt[3]{2}\right)^3 \approx 17.3217,$$

completing the proof of Theorem 3.2.4.
3.3 Minimizing Evacuation Time, Given Constant Evacuation Energy

In this section we consider the problem of minimizing evacuation time, given constant total (or makespan) evacuation energy for the case that $b = 1$. First we observe that if the robots can use only a finite number of speeds, there is no feasible solution to the problems $\text{ME}^1_d(\Delta)$ or $\text{TE}^1_d(\Delta)$.

**Theorem 3.3.1.** If $\Delta$ is a constant, and the robots have access to only a finite number of speeds, there is no feasible solution to the problems $\text{ME}^1_d(\Delta)$ or $\text{TE}^1_d(\Delta)$

*Proof. (Theorem 3.3.1)* Suppose the robots can only use speeds in a finite set. Wlog let $s$ be the minimum speed in the set. Define $d' = \Delta/s^2$, and place the exit at $d' + \epsilon$ for any $\epsilon > 0$. Travelling at any speed at or above $s$, it is impossible for even one of the robots to reach the exit with energy $\leq \Delta$.

Next we prove a lower bound on the evacuation time in this setting.

**Theorem 3.3.2.** For every constant $e \in \mathbb{R}_+$, the optimal evacuation time for problem $\text{ME}^1_d(e)$ is $\Omega(d^{3/2})$, asymptotically in $d$.

*Proof. (Theorem 3.3.2)* For any arbitrarily large value of $d$, we place the exit at distance $d$ from the origin. For any robot to reach the exit before running out of battery, a robot can travel at speed at most $\sqrt{e/d}$. Therefore the time for even the first robot to reach the exit is at least $\frac{d}{\sqrt{e/d}} = \frac{d^{3/2}}{\sqrt{e}}$.

Note that the above lower bound also holds for problem $\text{TE}^1_d(e)$ (if the total evacuation energy is no more than $e$, then also the makespan evacuation energy is no more than $e$). Next we prove that this naive lower bound is nearly tight (up to a $\log d$ factor). First we consider the case that $e \leq 1$. Then, we show how to modify our solution to also solve the problem when $e > 1$.

The key idea is to allow functional speed $s = s(x)$ to depend on the distance $x$ of the robot from the origin. We will make sure that the choice of $s$ is such that, for every large enough $d$, once the exit is located at distance $d$, there is “enough” leftover energy for the other robot to evacuate too. For that, we will choose the maximum possible speed $r$ (which can now depend on $d$, and which will be constant) so as to evacuate without exceeding the maximum energy bounds. Notably, even though our algorithmic solution is described as a solution to $\text{TE}^1_d(e)$, it will be transparent in the proof that it is also a feasible solution to $\text{ME}^1_d(e)$.
Theorem 3.3.3. For every constant \( e \leq 1 \), problem \( \text{TE}_d^1(e) \) admits a solution by \( N_{s,r} \), where (functional) speed \( s \) is chosen as

\[
s(x) = \frac{1}{\sqrt{2 + 2x} \left(1/e + \log(1 + x)\right)}.
\]

When the exit is found (hence its distance \( d \) from the origin becomes known), speed \( r \) is chosen as

\[
r = \sqrt{\frac{e}{2d(e \log(d + 1) + 1)}},
\]

inducing evacuation time \( O\left(d^{3/2} \log d\right) \), where in particular the constant in the asymptotic (in \( d \)) is independent of \( e \).

Proof. (Theorem 3.3.3) First we observe that since \( e \leq 1 \), \( s(x) \leq 1 \) for all \( x \geq 0 \). Given that \( d \) is at least, say, 1, it is also immediate that \( r \leq 1 \), hence the speed choices comply with the speed bound.

The exit placed at distance \( d \) from the origin is located by the finder in time

\[
\int_0^d \frac{1}{s(x)} \, dx = \frac{2\sqrt{2} \left( (d + 1)^{3/2} (3e \log(d + 1) - 2e + 3) + 2e - 3 \right)}{9e} \leq d^{3/2} \log d,
\]

where the inequality holds for every \( e \leq 1 \), and for big enough \( d \).

When the exit is located by a robot, the other robot is at distance \( 2d \) from the exit. Moreover, each of the robots have used energy

\[
\int_0^d s^2(x) \, dx = \frac{e}{2} - \frac{e}{2e \log(d + 1) + 2},
\]

hence the leftover energy for the non-finder (i.e., the robot that did not find the exit) to evacuate is at least

\[
e - 2 \left( \frac{e}{2} - \frac{e}{2e \log(d + 1) + 2} \right) = \frac{e}{e \log(d + 1) + 1}.
\]

The non-finder is informed of \( d \), and hence can choose constant speed \( r \) so as to use exactly all of the leftover energy, i.e. by choosing \( r \) satisfying

\[
\int_0^{2d} r^2 \, dx = \frac{e}{e \log(d + 1) + 1}.
\]

Note that \( N_{s,r} \) is also a feasible solution to problem \( \text{ME}_d^b(e) \). Solving for \( r \) gives the value declared at the statement of the theorem. Finally, choosing this specific value of \( r \), the non-finder needs additional \( 2d/r \) time to evacuate, which is at most

\[
(2d)^{3/2} \sqrt{\frac{e \log(d + 1) + 1}{e}} \leq (2d)^{3/2} \sqrt{\frac{\log(d + 1)}{e}} \leq d^{3/2} \log d,
\]

85
where the last inequality holds for big enough $d$, since $e$ is constant. So the overall evacuation time is no more than $2d^{3/2} \log d$, for big enough $d$, as promised.

It remains to address the case $e > 1$. For this, we recall that we solve $\text{TE}_d^1(e)$ for large enough values of $d$, and we modify our solution so as to choose functional speed

$$\bar{s}(x) := \min\{s(x), 1\},$$

effectively using even less energy than before. The distance that is traversed at speed 1 depends only on constant $e$, and hence the additional evacuation time is $O(1)$ with respect to $d$.

### 3.4 Minimizing Evacuation Time with Bounded Linear Total Evacuation Energy

In this section we study the problem $\text{TE}_d^1(\Delta)$ of minimizing the evacuation time, where $\Delta = ed$ for some constant $e$. We show how to choose optimal speed values $s, r$ for algorithm $\mathcal{N}_{s,r}$. Note that even though $d$ is unknown to the algorithm, speeds $s, r$ may depend on the known constant $e$, and the maximum speed $b = 1$.

In this section we prove the following theorem:

**Theorem 3.4.1.** Let $\delta = 2 + \sqrt{2} \approx 3.25992$. For every constant $e \in \mathbb{R}^+$, problem $\text{TE}_d^1(ed)$ admits a solution by $\mathcal{N}_{s,r}$, where speeds $s, r$ are chosen as follows

<table>
<thead>
<tr>
<th>$e$</th>
<th>$s$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e &lt; \delta$</td>
<td>$\sqrt{\frac{e}{2(1+2^{1/3})}}$</td>
<td>$\sqrt{\frac{e-2}{2}}$</td>
</tr>
<tr>
<td>$\delta \leq e &lt; 4$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$e \geq 4$</td>
<td>$\sqrt{\frac{e}{(2+2^{1/3})}}$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

The induced evacuation time is given by $g(e)d$ where $g(e)$ is given by:

$$g(e) := \begin{cases} \sqrt{(2+2^{1/3})^{3}}, & e < \delta \\ 2 + \sqrt{\frac{e-2}{e-2}}, & \delta \leq e < 4 \\ 3, & e \geq 4 \end{cases}$$

First we observe that, given the values of $s = s(e), r = r(e)$, it is a matter of straightforward calculations to verify, assuming they are feasible and optimal, that the induced evacuation time is indeed equal to $g(e)d$ as promised. Given Lemma 3.2.5 we know that the
optimal speed choices for algorithm $N_{s,r}$, for problem $TE^1_d (ed)$ are obtained as the solution to the following NLP.

$$\min_{s,r \in \mathbb{R}} \frac{1}{s} + \frac{2}{r} \quad \text{(NLP')}
$$

$$\text{s.t. } 2(s^2 + r^2) \leq e \leq s, r \leq 1$$

The optimal solutions to NLP$_e'$ can be obtained by solving complicated algebraic systems and by invoking KKT conditions, for the various values of $e$, as we also did for NLP$_b^*$. However, the advantage is that one can map the optimal solutions to NLP$_1^c$, see Theorem 3.2.4 and use $b = 1$, to feasible solutions to NLP$_e'$. Then, we just need to verify first-order optimality conditions for the candidate optimizers. Since the NLP is convex, these should also be unique global optimizers.

Indeed, one of the critical structural properties pertaining to the optimizers of NLP$_1^c$ is that the time constraint $\frac{1}{s} + \frac{2}{r} \leq cd$ is satisfied tightly. At the same time, the optimal speed values, as described in Theorem 3.2.4, as a function of $c$, achieve evacuation energy equal to $f(c)d^2$. Attempting to find the correspondence between parameters $c, e$ (and problems NLP$_1^c$, NLP$_e'$), we consider the transformation $f(c)\frac{2}{d^2} = e$. For the various cases of the piece-wise function $f$, the transformation gives rise to the piece-wise function $g$ and optimal speeds $s, r$ (as a function of $e$) of Theorem 3.4.1.

Overall, the previous approach provides just a mapping between the provable optimizers $s(c), r(c)$ to NLP$_1^c$, and candidate solutions $s(e), r(e)$ to NLP$_e'$, and more importantly, it saves us from solving complicated algebraic systems induced by KKT conditions. What we verify next (which is much easier), is that feasibility and KKT conditions are indeed satisfied for the obtained candidate solutions $s(e), r(e)$. Since the NLP is convex, that also shows that $s(e), r(e)$, as stated in Theorem 3.4.1 are actually global optimizers to NLP$_e'$.

**Lemma 3.4.2.** For every $e \in \mathbb{R}_+$, speeds $s(e), r(e)$, as they are defined in Theorem 3.4.1 are feasible to NLP$_e'$.

**Proof.** (Lemma 3.4.2) Speeds $s = s(e)$ and $r = r(e)$ are clearly non negative. Next we verify that they never attain value more than 1. We examine two cases. When $e < 2 + \sqrt{2}$, it is easy to see that $r/s = \sqrt{2}$. Hence, it is enough to check that $r \leq 1$, which is immediate from the formula of $r = r(e)$. In the other case, we assume $e \in [\delta, 4)$. Speed $r$ is clearly at most 1, as well as $s = \sqrt{(e - 2)/2} \leq \sqrt{(4 - 2)/2} = 1$.

Next we verify that the given speeds comply with the evacuation energy bounds. When $e < 2 + \sqrt{2}$ we have

$$2(s^2 + r^2) = 2e \left( \frac{1}{2(1 + 2^{2/3})} + \frac{1}{(2 + 2^{1/3})} \right) = e.$$
When \( e \in [\delta, 4) \) we have
\[
2(s^2 + r^2) = 2e \left( \frac{e - 2}{2} + 1 \right) = e.
\]
Lastly, when \( e \geq 4 \) both speeds are 1, and clearly, \( 2(s^2 + r^2) = 4 \leq e \) as wanted. \( \square \)

**Lemma 3.4.3.** For every \( e \in \mathbb{R}_+ \), speeds \( s(e), r(e) \), as stated in Theorem 3.4.1 are the optimal solutions to NLP\( e' \).

**Proof.** (Lemma 3.4.3) For every \( e \in \mathbb{R}_+ \), we verify that speeds \( s(e), r(e) \) satisfy first-order optimality conditions. Since NLP\( e' \) is convex, that would imply that \( s(e), r(e) \) are the unique optimizers.

First we observe that the energy inequality constraint is always tight (verified within the proof of Lemma 3.4.2). Apart from that constraint, let \( I(e) \) (possibly empty) denote the set of constraints, among \( s, r \leq 1 \), which are tight for the specific candidate optimizer \( s(e), r(e) \), and for a specific value of \( e \). For \( i \in I \) we denote the corresponding constraint by \( g_i(s, r) \leq 1 \).

When \( e < 2 + \frac{3}{\sqrt{2}} \), the bound constraint is the only constraint which is tight. Therefore KKT conditions are satisfied as long as there exists \( \lambda \geq 0 \) such that
\[
-\nabla \left( \frac{1}{s} + \frac{2}{r} \right) = \lambda \nabla 2(s^2 + r^2) \iff \left( \frac{1/s}{2/r} \right) = \lambda \left( \frac{4s}{4r} \right)
\]
A solution exists as long as \( 2s^3 = r^3 \), which is indeed, the case, which also implies that \( \lambda = 1/(4s^3) \geq 0 \).

When \( e \in [2 + \frac{3}{\sqrt{2}}, 4) \), the bound constraint and constraint \( r \leq 1 \) are tight. Therefore KKT conditions are satisfied as long as there exist \( \lambda_1, \lambda_2 \geq 0 \) such that
\[
-\nabla \left( \frac{1}{s} + \frac{2}{r} \right) = \lambda_1 \nabla 2(s^2 + r^2) + \lambda_2 \left( \frac{0}{1} \right) \iff \left( \frac{1/s}{2/r} \right) = \lambda_1 \left( \frac{4s}{4r} \right) + \lambda_2 \left( \frac{0}{1} \right)
\]
Solving for \( \lambda_1, \lambda_2 \), and using the provided values for \( s = s(e) \) and \( r = r(e) \) we obtain
\[
\lambda_1 = \sqrt{2} \left( \frac{1 + 2^{2/3}}{e} \right)^{3/2}, \quad \lambda_2 = 2 - 2 \left( \frac{2 + \sqrt{2}}{e} \right)^{3/2},
\]
and clearly both values are nonnegative when \( e \geq 2 + \sqrt{2} \).

Lastly, for the first-order optimality conditions, when \( e \geq 1 \), all (but the non-negativity constraints) are tight. Therefore KKT conditions are satisfied as long as there exist \( \lambda_1, \lambda_2, \lambda_3 \geq 0 \) such that
\[
-\nabla \left( \frac{1}{s} + \frac{2}{r} \right) = \lambda_1 \nabla 2(s^2 + r^2) + \lambda_2 \left( \frac{1}{0} \right) + \lambda_3 \left( \frac{0}{1} \right) \iff \left( \frac{1/s}{2/r} \right) = \lambda_1 \left( \frac{4s}{4r} \right) + \lambda_2 \left( \frac{1}{0} \right) + \lambda_3 \left( \frac{0}{1} \right)
\]
Since \( r = s = 1 \), the above system simplifies to

\[
\begin{align*}
1 &= 4\lambda_1 + \lambda_2 \\
2 &= 4\lambda_1 + \lambda_3
\end{align*}
\]

which admits the solution \( \lambda_1 = 1/4 \geq 0, \lambda_2 = 0, \lambda_3 = 1 \geq 0. \)

### 3.5 Minimizing Evacuation Time with Bounded Linear Makespan Evacuation Energy

In this section we study the problem \( \text{ME}_d^1(\Delta) \) of minimizing the evacuation time, given that the makespan evacuation energy \( \Delta = ed \) for some constant \( e \). We show how to choose optimal speed values \( s, r \) for algorithm \( N_{s,r} \). Note that even though \( d \) is unknown to the algorithm, speeds \( s, r \) may depend on the known value \( e \), and the maximum speed \( b = 1 \).

**Theorem 3.5.1.** For every constant \( e \in \mathbb{R}_+ \), problem \( \text{ME}_d^1(ed) \) admits a solution by \( N_{s,r} \), where speeds \( s, r \) are chosen as follows

<table>
<thead>
<tr>
<th>( e )</th>
<th>( s )</th>
<th>( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e &lt; 3 )</td>
<td>( \sqrt{\frac{e}{3}} )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( e \geq 3 )</td>
<td>( \sqrt{\frac{e}{3}} )</td>
<td>( 1 )</td>
</tr>
</tbody>
</table>

The induced evacuation time is given by \( g(e)d \) where

\[
g(e) := \begin{cases} 
3\sqrt{\frac{e}{3}}, & e < 3 \\
3, & e \geq 3
\end{cases}
\]

**Proof.** (Theorem 3.5.1) What distinguishes the performance, and feasibility, of \( N_{s,r} \) between \( \text{TE}_d^1(ed) \) and \( \text{ME}_d^1(ed) \), is that in the former, the total evacuation energy (equal to \( d(2s^2 + 2r^2) \)) is bounded by \( e \), while in the latter the makespan evacuation energy (equal to \( d(s^2 + 2r^2) \)) is bounded by \( e \). Hence, similar to the analysis for \( \text{TE}_d^1(ed) \), the optimal speed choices for \( N_{s,r} \) to \( \text{ME}_d^1(ed) \) are the optimal solutions to the following NLP.

\[
\begin{align*}
\min_{s,r \in \mathbb{R}_+} & \quad \frac{1}{s} + \frac{2}{r} \\
\text{s.t.} & \quad s^2 + 2r^2 \leq e \\
& \quad 0 \leq s, r \leq 1
\end{align*}
\]

\( \text{(NLP)'e} \)

Note that \( \text{NLP}'_e \) is convex, hence any choice of feasible speeds satisfying first-order optimality (KKT) conditions is also the unique global minimizer. Moreover, the choices of \( s, r \)
of the statement of the theorem are clearly feasible to NLP′′. Hence, it suffices to show that the choices of \( s, r \) do indeed satisfy KKT conditions.

When \( e < 3 \) we note that the energy constraint is tight, while both speed constraints are not tight. Hence, \( s, r \) are the unique optimizers if there exists \( \lambda \geq 0 \) satisfying

\[
-\nabla \left( \frac{1}{s} + \frac{2}{r} \right) = \lambda \nabla (s^2 + 2r^2) \iff \left( \frac{1/s^2}{2/r^2} \right) = \lambda \left( \frac{2s}{4r} \right)
\]

from which we conclude that \( \lambda = 1/(2s^3) = 1/(2r^3) > 0 \) as wanted (for \( s = r = \sqrt{e/3} \)).

When \( e \geq 3 \) we note that the speed constraints are both tight, while the energy constraint is tight only when \( e = 3 \). In that case, it suffices to show that there exist nonnegative \( \lambda_1, \lambda_2 \) satisfying

\[
-\nabla \left( \frac{1}{s} + \frac{2}{r} \right) = \lambda_1 \left( \frac{1}{0} \right) + \lambda_2 \left( 0 \frac{1}{1} \right)
\]

Clearly, \( \lambda_1 = 1/s^2 = 1 > 0 \) and \( \lambda_2 = 2/r^2 = 2 > 0 \), which concludes the proof.

\[\square\]

### 3.6 Conclusion

We considered a variant of the well-studied linear search problem in the wireless communication model where two robots try to reach a hidden exit on the line. The novelty of our work is the study of the total evacuation energy, a new measure of solution quality that was first introduced in [61]. The new measure, together with the traditional objective of minimizing the evacuation time, gives rise to a challenging multi-objective optimization problem. We analyze the problem by minimizing one of the objectives while converting the other objective to a bounded constraint on the search problem, e.g., minimizing evacuation time subject to bounded evacuation energy. Somewhat surprisingly, even though the search domain is unbounded, we show that there is indeed a feasible solution to the problem even when the evacuation energy is restricted to be constant.

While our definition of the evacuation energy is inspired by fundamental principles of physics, the definition does not consider any additional energy expended to change the direction of motion or to arbitrarily adjust the speed. Alternative formulations of energy consumption that incorporate these considerations may give rise to new multi-objective optimization problems that will be studied in future work.

Our search problem was considered under the wireless communication model, while the face-to-face model was studied by the same authors in [61]. It would be worthwhile to explore other communication models, such as the pebble model or the blackboard model, for other geometric search domains like a unit disk, the plane, or a graph, and for a given number of searchers. Likewise, one may inject uncertainty into the problem by either allowing faulty
robots and/or performing average case instead of worst case analysis. Finally, improving upon the trade-offs we established in this work remains an interesting open problem.

Acknowledgments

This research is supported by NSERC discovery grants, NSERC graduate scholarship, and NSF.

References


92


Chapter 4

Energy Consumption of Group Search on a Line

This chapter presents the paper “Energy Consumption of Group Search on a Line” [61] which appears in the proceedings of ICALP2019. The paper treats the same problem as Chapter 3 except in the F2F model of communication.
Energy Consumption of Group Search on a Line

Jurek Czyzowicz\textsuperscript{1}, Konstantinos Georgiou\textsuperscript{2}, Ryan Killick\textsuperscript{3}, Evangelos Kranakis\textsuperscript{3}, Danny Krizanc\textsuperscript{4}, Manuel Lafond\textsuperscript{5}, Lata Narayanan\textsuperscript{6}, Jaroslav Opatrny\textsuperscript{6}, Sunil Shende\textsuperscript{7}

\textsuperscript{1}Université du Québec en Outaouais, Gatineau, Québec, Canada
\textsuperscript{2}Department of Mathematics, Ryerson University, Toronto, Ontario, Canada
\textsuperscript{3}School of Computer Science, Carleton University, Ottawa, Ontario, Canada
\textsuperscript{4}Department of Mathematics & Comp. Sci., Wesleyan University, Middletown, CT, USA
\textsuperscript{5}Department of Computer Science, Université de Sherbrooke, Sherbrooke, Québec, Canada
\textsuperscript{6}Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada
\textsuperscript{7}Department of Computer Science, Rutgers University, Camden, NJ, USA

Abstract

Consider two robots that start at the origin of the infinite line in search of an exit at an unknown location on the line. The robots can collaborate in the search, but can only communicate if they arrive at the same location at exactly the same time, i.e. they use the so-called face-to-face communication model. The group search time is defined as the worst-case time as a function of \(d\), the distance of the exit from the origin, when both robots can reach the exit. It has long been known that for a single robot traveling at unit speed, the search time is at least \(9d - o(d)\); a simple doubling strategy achieves this time bound. It was shown recently in [40] that \(k \geq 2\) robots traveling at unit speed also require at least \(9d\) group search time.

We investigate energy-time trade-offs in group search by two robots, where the energy loss experienced by a robot traveling a distance \(x\) at constant speed \(s\) is given by \(s^2x\), as motivated by energy consumption models in physics and engineering. Specifically, we consider the problem of minimizing the total energy used by the robots, under the constraints that the search time is at most a multiple \(c\) of the distance \(d\) and the speed of the robots is bounded by \(b\). Motivation for this study is that for the case when robots must complete the search in \(9d\) time with maximum speed one \((b = 1; c = 9)\), a single robot requires at least \(9d\) energy, while for two robots, all previously proposed algorithms consume at least \(28d/3\) energy.

When the robots have bounded memory and can use only a constant number of fixed speeds, we generalize an algorithm described in [15][40] to obtain a family of algorithms parametrized by pairs of \(b, c\) values that can solve the problem for the entire spectrum of these pairs for which the problem is solvable. In particular, for each such
pair, we determine optimal (and in some cases nearly optimal) algorithms inducing the lowest possible energy consumption.

We also propose a novel search algorithm that simultaneously achieves search time $9d$ and consumes energy $8.42588d$. Our result shows that two robots can search on the line in optimal time $9d$ while consuming less total energy than a single robot within the same search time. Our algorithm uses robots that have unbounded memory, and a finite number of dynamically computed speeds. It can be generalized for any $c, b$ with $cb = 9$, and consumes energy $8.42588b^2d$.

4.1 Introduction

The problem of searching for a treasure at an unknown location in a specified continuous domain was initiated over fifty years ago \cite{22, 25}. Search domains that have been considered include the infinite line \cite{13, 22, 25, 128}, a set of rays \cite{32, 35}, the unit circle \cite{36, 71, 144}, and polygons \cite{79, 118, 132}. Consider a robot (sometimes called a mobile agent) starting at some known location in the domain and looking for an exit that is located at an unknown distance $d$ away from the start. What algorithm should the robot use to find the exit as soon as possible? The most common cost measure used for the search algorithm is the worst-case search time, as a function of the distance $d$ of the exit from the starting position. For a fixed-speed robot, the search time is proportional to the length of the trajectory of the robot. Other measures such as turn cost \cite{84} and different costs for revisiting \cite{31} have also been studied.

We consider for the first time the energy consumed by the robots while executing the search algorithm. The energy used by a robot to travel a distance $x$ at speed $s$ is computed as $s^2x$ and is motivated from the concept of viscous drag in fluid dynamics; see Section 4.2 for details on the energy model. For a single robot searching on the line, the classic Spiral Search algorithm (also known as the doubling strategy) has search time $9d$ and is known to be optimal when the robot moves with unit speed. Since in the worst case, the robot travels distance $9d$ at unit speed, the energy consumption is $9d$ as well. Clearly, as the speed of the robot increases, the time to find the exit decreases but the energy used increases. Likewise, as the speed of the robot decreases, the time to find the exit increases, while the energy consumption decreases. Thus there is a natural trade-off between the time taken by the robot to search for the exit and the energy consumed by the robot. To investigate this trade-off, we consider the problem of minimizing the total energy used by the robots to perform the search when the speed of the robot is bounded by $b$, and the time for the search is at most a multiple $c$ of the distance $d$ from the starting point to the exit.

Group search by a set of $k \geq 2$ collaborating robots has recently gained a lot of attention. In this case, the search time is the time when all $k$ robots reach the exit. The problem has also been called evacuation, in view of the application when it is desired that all robots reach and evacuate from the exit. Two models of communication between the robots have been
considered. In the wireless communication model, the robots can instantly communicate with each other at any time and over any distance. In the face-to-face communication model (F2F), two robots can communicate only when in the same place at the same time. In many search domains, and for both communication models, group search by $k \geq 2$ agents has been shown to take less time than search by a single agent; see for example [71, 79].

In this paper, we focus on group search on the line, by two robots using the F2F model. Chrobak et al [40] showed that group search in this setting cannot be performed in time less than $9d - o(d)$, regardless of the number of robots, assuming all robots use at most unit speed. They also describe several strategies that achieve search time $9d$. In the first strategy, the two robots independently perform the Spiral Search algorithm, using unit speed during the entire search. Next, they consider a strategy first described in [15], that we call the Two-Turn strategy, whereby two robots head off independently in opposite directions at speed $1/3$; when one of them finds the exit, it moves at unit speed to chase and catch the other robot, after which they both return at unit speed to the exit. Finally, they present a new strategy, called the Fast-Slow algorithm in which one robot moves at unit speed, while the other robot moves at speed $1/3$, both performing a spiral search. The doubling strategy is very energy-inefficient, it uses energy $18d$ if the two robots always travel together, or $14d$ if the robots start by moving in opposite directions. The other two algorithms both use energy $28d/3 > 9d$. Interestingly, the two strategies that achieve an energy consumption of $28d/3$ with search time $9d$, both use two different and pre-computed speeds, but are quite different in terms of the robot capacities needed. In the Two-Turn strategy, the robots are extremely simple and use constant memory; they use only three states. In Fast-Slow and Spiral Search, the robots need unbounded memory, and perform computations to determine how far to go before turning and moving in the opposite direction.

Memory capability, time- and speed-bounded search, and energy consumption by a two-robot group search algorithm on the line: these considerations motivate the following questions that we address in our paper:

1. Is there a search strategy for constant-memory robots that has energy consumption $< 9d$?
2. Is there any search strategy that uses time $9d$ and energy $< 9d$?

4.1.1 Our results

We generalize the Two-Turn strategy for any values of $c, b$. We analyze the entire spectrum of values of $c, b$ for which the problem admits a solution, and for each of them we provide optimal or nearly-optimal speed choices for our robots (Theorem 4.3.4). In particular, and somewhat surprisingly, our proof makes explicit how for any fixed $c$ the optimal speed choices do not simply "scale" with $b$; rather more delicate speed choices are necessary to comply with the speed and search time bounds. For the special case of $c \cdot b = 9$, our results
match with the specific Two-Turn strategy described in [40]. Our results further show that no Two-Turn strategy can achieve energy consumption less than $9d$ while keeping the search time at $9d$. In fact, we conjecture that this trade-off is impossible for any group search strategy that uses only constant memory robots.

In the unbounded-memory model, for the special case of $c = 9$ and $b = 1$, we give a novel search algorithm that achieves energy consumption of $8.42588d$, thus answering the second question above in the affirmative. This result shows that though two robots cannot search faster than one robot on the line [40], somewhat surprisingly, two robots can search using less total energy than one robot, in the same optimal time. Our algorithm uses robots that have unbounded memory, and a finite number of dynamically computed speeds. Note that our algorithm can be generalized for any $c, b$ with $cb = 9$, and utilizes energy $8.42588b^2d$ (Theorem 4.4.7).

4.1.2 Related Work

Several authors have investigated various aspects of mobile robot (agent) search, resulting in an extensive literature on the subject in theoretical computer science and mathematics (e.g., see [8, 110] for reviews). Search by constant-memory robots has been done mainly for finite-state automata (FSA) operating in discrete environments like infinite grids, their finite-size subsets (labyrinths) and other graphs. The main concern of this research was the feasibility of search, rather than time or energy efficiency. For example, [38] showed that no FSA can explore all labyrinths, while [29] proved that one FSA using two pebbles or two FSAs, communicating according to the F2F model can explore all labyrinths. However, no collection of FSAs may explore all finite graphs communicating in the F2F model [151] or wireless model [51]. On the other hand, all graphs of size $n$ may be explored using a robot having $O(\log n)$ memory [150].

Exploration of infinite grids is known as the ANTS problem [93], where it was shown that four collaborating FSAs in the semi-synchronous execution model and communicating according to the F2F scenario can explore an infinite grid. Recently, [37] showed that four FSAs are really needed to explore the grid (while three FSAs can explore an infinite band of the 2-dimensional grid).

Continuous environment cases have been investigated in several papers when the efficiency of the search is often represented by the time of reaching the target (e.g., see [13, 22, 25, 128]). Even in the case of continuous environment as simple as the infinite line, after the seminal papers [22, 25], various scenarios have been studied where the turn cost has been considered [84], the environment was composed of portions permitting different search speeds [78], some knowledge about the target distance was available [32] or where some other parameters are involved in the computation of the cost function [31] (e.g. when the target is moving).

The group search, sometimes interpreted as the evacuation problem has been studied first
for the disc environment under the F2F [36, 57, 71, 79, 137] and wireless [57] communication
cenarios and then also for other geometric environments (e.g., see [79]). Other variants of
search/evacuation problems with a combinatorial flavour have been recently considered in
[43, 63, 64, 112, 111]. Some papers investigated the line search problem in the presence of

crash faulty [77] and Byzantine faulty agents [70]. The interested reader may also consult
the recent survey [68] on selected search and evacuation topics.

The energy used by a mobile robot is usually considered as being spent solely for

 travelling. As a consequence, in the case of a single, constant speed robot the search time is

 proportional to the distance travelled and the energy used by a robot. Therefore the problems

 of minimization of time, distance or energy are usually equivalent for most robots’ tasks. For

 teams of collaborating robots, the searchers often need to synchronize their walks in

 order to wait for information communicated by other searchers (e.g., see [36, 57, 137]),

hence the time of the task and the distance travelled are different. However, the distance

 travelled by a robot and its energy used are still commensurable quantities.

To the best of our knowledge, energy consumption as a function of mobile robot speed

 which is based on laws of engineering physics (related to the drag force) has never been

 studied in the search literature before. Our present work is motivated by [40], which proves

 that the competitive ratio 9 is tight for group search time with two mobile agents in the

 F2F model when both agents have unit maximal speeds. More exactly, it follows from [40]

 that having more unit-speed robots cannot improve the group search time obtained by a

 single robot. Nevertheless, our paper shows that using more robots can improve the energy

 spending, while keeping the group-search time still the best possible.

Chrobak et al [40] present interesting examples of group search algorithms for two

 distinct speed robots communicating according to the F2F scenario. An interested reader

 may consult [17], where optimal group search algorithms for a pair of distinct maximal

 speed robots were proposed for both communication scenarios (F2F and wireless) and for

 any pair of robots’ maximal speeds. It is interesting to note that, according to [17], for any

 distinct-speed robots with F2F communication, the optimal group search time is obtained if

 one of the robots perform the search step not using its full speed.

Paper Organization: In Section 4.2 we formally define the evacuation problem EE$^b$, and

 proper notions of efficiency. Our algorithms and their analysis for constant-memory

 robots is presented in Section 4.3 while in Section 4.4 we introduce and analyze algorithms

 for unbounded-memory robots. Whenever we omit proofs, due to space limitations, we

 provide an outline of our arguments. The interested reader may consult the full version of

 our paper [67] for the missing details.
4.2 Preliminaries

Two robots start walking from the origin of an infinite (bidirectional) line in search of a hidden exit at an unknown absolute distance \( d \) from the origin. The exit is considered found only when one of the robots walks over it. An algorithm for group search by two robots specifies trajectories for both robots and terminates when both robots reach the exit. The time by which the second robot reaches the exit is referred to as the search time or the evacuation time.

**Robot models:** The two robots operate under the F2F communication model in which two robots can communicate only when they are in the same place at the same time. Each robot can change its speed at any time. We distinguish between constant-memory robots that can only travel at a constant number of hard-wired speeds, and unbounded-memory robots that can dynamically compute speeds and distances, and travel at any possible speed.

**Energy model:** A robot moving at constant speed \( s \) traversing an interval of length \( x \) is defined to use energy \( s^2 \cdot x \). This model is well motivated from first principles in physics and engineering and corresponds to the energy loss experienced by an object moving through a viscous fluid [20]. In particular, an object moving with constant speed \( s \) will experience a drag force \( F_D \) proportional\(^1\) to \( s^2 \). In order to maintain the speed \( s \) over a distance \( x \) the object must do work equal to the product of \( F_D \) and \( x \) resulting in a continuous energy loss proportional to the product of the object’s squared speed and travel distance. For simplicity we have taken the proportionality constant to be one.

The total energy that a robot uses traveling at speeds \( s_1, s_2, \ldots, s_t \), traversing intervals of length \( x_1, x_2, \ldots, x_t \), respectively, is defined as \( \sum_{i=1}^{t} s_i^2 \cdot x_i \). For group search with two robots, the energy consumption is defined as the sum total of the two robots’ energies used until the search algorithm terminates.

For each \( d > 0 \) there are two possible locations for the exit to be at distance \( d \) from the origin: we will refer to either of these as input instances \( d \) for the group search problem. Our goal is to solve the following optimized search problem parametrized by two values, \( b \) and \( c \):

**Problem EE\(b^c\):** Design a group search algorithm for two robots in the F2F model that minimizes the energy consumption for \( d \)-instances under the constraints that the search time is no more than \( c \cdot d \) and the robots use speeds that are at most \( b \). When there are no speed limits on the robots (i.e. \( b = \infty \)), we abbreviate EE\(c^\infty\) by EE\(c\). Note that \( b, c \) are inputs to the algorithm, but \( d \) and the exact location of the exit are not known.

As it is standard in the literature on related problems, we assume that the exit is at least a known constant distance away from the origin. In this work, we pick the constant equal to 2, although our arguments can be adjusted to any other constant. It is not difficult to show that EE\(b^c\) is well defined for each \( b, c > 0 \) with \( bc \geq 1 \), and the optimal offline solution,

---

\(^1\)The constant of proportionality has (SI) units \( kg/m \) and depends, among other things, on the shape of the object and the density of the fluid through which it moves.
for instance $d$, is for both robots to move at speed $\frac{1}{c}$ to the exit. This offline algorithm has energy consumption $\frac{2d}{c^2}$. Consider an online algorithm for $EE^b_c$, which on any instance $d$ has energy consumption at most $e(c, b, d)$. The competitive ratio of the algorithm is defined as $\sup_{d>0} \frac{d^2}{2d^2} e(c, b, d)$.

Due to [40], and when $b = 1$, no online algorithm (for two robots) can have evacuation time less than $9d - \epsilon$ (for any $\epsilon > 0$ and for large enough $d$). By scaling, using arbitrary speed limit $b$, we obtain the following fact.

**Observation 4.2.1.** No online F2F algorithm can solve $EE^b_c$ if $cb < 9$.

### 4.3 Solving $EE^b_c$ with Constant-Memory Robots

In this section we propose a family of algorithms for solving $EE^b_c$ (including $b = \infty$). The family uses an algorithm that is parametrized by three discrete speeds: $s$, $r$ and $k$. The robots use these speeds depending on finite state control as follows:

**Algorithm $N_{s,r,k}$:** Robots start moving in opposite directions with speed $s$ until the exit is found by one of them. The finder changes direction and moves at speed $r > s$ until it catches the other robot. Together the two robots return to the exit using speed $k$.

**Lemma 4.3.1.** Let $b, c$ be such that there exist $s, r, k$ for which $N_{s,r,k}$ is feasible. Then, for instance $d$ of $EE^b_c$, the induced evacuation time of $N_{s,r,k}$ is $d \cdot T(s, r, k)$ and the induced energy consumption is $2d \cdot E(s, r, k)$, where

$$T(s, r, k) := \frac{2(k + r)}{k(r - s)} + 1, \quad E(s, r, k) := \frac{r}{r - s} \left( s^2 + r^2 + 2k^2 \right)$$

We propose a systematic way in order to find optimal values for $s, r, k$ of algorithm $N_{s,r,k}$ for optimization problem $EE^b_c$ (including $b = \infty$), whenever such values exist.

**Theorem 4.3.2.** Algorithm $N_{s,r,k}$ gives rise to a feasible solution to problem $EE^b_c$ if and only if $bc \geq 9$. For every such $b, c > 0$, the optimal choices of $N_{s,r,k}$ can be obtained by solving Non Linear Program:

$$\min_{s, r, k \in \mathbb{R}} E(s, r, k) \quad (fNLP^b_c)$$

s.t. $T(s, r, k) \leq c$

$\quad r > s$

$\quad 0 \leq s, r, k \leq b$

where functions $E(\cdot, \cdot, \cdot), T(\cdot, \cdot, \cdot)$ are as in Lemma 4.3.1. Moreover, if $s_0, r_0, k_0$ are the optimizers to $fNLP^b_c$, then the competitive ratio of $N_{s_0,r_0,k_0}$ equals $c^2 \cdot E(s_0, r_0, k_0)$.
The following subsections are devoted to solving \( f\text{NLP}^b \), effectively proving Theorem 4.3.4. First in Section 4.3.1 we solve the case \( b = \infty \) and we use our findings to solve the case of bounded speeds \( b \) in the follow-up Section 4.3.2.

### 4.3.1 Optimal Choices of \( N_{s,r,k} \) for the Unbounded-Speed Problem

In this section we propose solutions to the unbounded-speed problem \( EE_c \). Since \( EE_c \) is the same as \( EE_c^\infty \) the problem is well-defined for every fixed \( c > 0 \). Moreover, by the proof of Theorem 4.3.2 algorithm \( N_{s,r,k} \) induces a feasible solution for every \( c > 0 \) as well, and the optimal speeds can be found by solving \( f\text{NLP}_c^\infty \). Indeed, in the remainder of the section we show how to choose optimal values for \( s, r, k \) for solving \( EE_c \) with \( N_{s,r,k} \).

Let
\[
\sigma \approx 2.65976, \quad \rho \approx 11.3414, \quad \kappa \approx 6.63709, \tag{4.1}
\]
whose exact values are the roots of an algebraic system and will be formally defined later. The main theorem of this section reads as follows.

**Theorem 4.3.3.** Let \( \sigma, \rho, \kappa \) as in (4.1). For every \( c > 0 \), the optimal speeds of \( N_{s,r,k} \) for problem \( EE_c \) are \( s = \frac{\sigma}{c}, \quad r = \frac{\rho}{c}, \quad k = \frac{\kappa}{c} \). Moreover, the competitive ratio of the corresponding solution is independent of \( c \) and equals
\[
\rho \left(\frac{(2\sigma^2 + \rho^2 + \kappa^2)}{\rho - \sigma}\right) \approx 292.369.
\]

A high level outline of the proof of Theorem 4.3.3 is as follows. First we show that any optimal choices of the speeds of \( N_{s,r,k} \) must satisfy the time constraint of \( f\text{NLP}_c^\infty \) tightly. Then, we show that finding optimal speeds \( s, r, k \) of \( N_{s,r,k} \) for the general problem \( EE_c \) reduces to problem \( EE_1 \). Finally, we obtain the optimal solution to \( f\text{NLP}_1^\infty \) by standard tools of nonlinear programming (KKT conditions).

### 4.3.2 (Sub)Optimal Choices of \( N_{s,r,k} \) for the Bounded-Speed Problem

In this section, we show how to choose optimal values for \( s, r, k \) for solving \( EE_b \) with \( N_{s,r,k} \), for the entire spectrum of \( c, b \) values for which the problem is solvable by online algorithms.

The main result of this section is the following:

**Theorem 4.3.4.** Let \( \gamma_1 \approx 9.06609, \quad \gamma_2 = \rho \approx 11.3414, \) and \( \sigma, \rho, \kappa \) as in (4.1). For every \( c, b > 0 \) with \( cb \geq 9 \), the following choices of speeds \( s, r, k \) are feasible for \( N_{s,r,k} \):

<table>
<thead>
<tr>
<th>( cb )</th>
<th>( s )</th>
<th>( r )</th>
<th>( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 9 \leq cb \leq \gamma_1 )</td>
<td>( -\sqrt{(bc)^2 - 10bc + 9 + bc - 3} )</td>
<td>( b )</td>
<td>( b )</td>
</tr>
<tr>
<td>( \gamma_1 &lt; cb &lt; \gamma_2 )</td>
<td>( 0.532412b - 0.0262661b^2c )</td>
<td>( \frac{2b}{bcs - b - cs^2 - s} )</td>
<td>( \frac{\sigma}{c} )</td>
</tr>
<tr>
<td>( cb \geq \gamma_2 )</td>
<td>( \frac{b}{c} )</td>
<td>( \frac{b}{c} )</td>
<td>( \frac{\rho}{c} )</td>
</tr>
</tbody>
</table>
The induced competitive ratio is given by:

\[
f(x) := \begin{cases} 
\frac{1}{2}x \left( x - \sqrt{(x-9)(x-1)} \right) + \sqrt{(x-9)(x-1)+3}, & 9 \leq x \leq \gamma_1 \\
\frac{x^2 \left( (0.532412 -0.0262661)x^2 + \frac{11.95.6/90.2699 -1.x)^2}{(x(x-2.5698)-398.916)x+222.18x+1} \right)}{0.0262661x+0.467588}, & \gamma_1 < x < \gamma_2 \\
x \geq \gamma_2 
\end{cases}
\]

and the induced energy, for instances \(d\), is \(f(cb)^{2d/c^2}\). Moreover, the competitive ratio depends only on the product \(cb\).

In particular, the speeds’ choices are optimal when \(cb \leq \gamma_1\) and when \(cb \geq \gamma_2\). When \(\gamma_1 < cb < \gamma_2\), the derived competitive ratio is no more than 0.03 additively off from that induced by optimal choices of \(s, r, k\).

**Corollary 4.3.5.** For \(c = 9, b = 1\), the bounded-memory robot algorithm \(N_{s,r,k}\) has energy consumption \(28d/3\) and competitive ratio 378.

Theorem 4.3.4 is proven by solving \(f^{NLP}_b\) of Theorem 4.3.2. Speed values \(s, r, k\), are chosen optimally when \(cb\) is either at most \(\gamma_1\) or at least \(\gamma_2\) (i.e. optimizers to \(f^{NLP}_b\) admit analytic description). The optimal speed parameters when \(\gamma_1 < cb < \gamma_2\) cannot be determined analytically (they are roots of high degree polynomials). The values that appear in Theorem 4.3.4 are heuristically chosen, but interestingly induce nearly optimal competitive ratio.

The proof of Theorem 4.3.4 is given by Lemma 4.3.6 (the case \(cb \leq \gamma_1\)), Lemma 4.3.7 (the case \(cb \geq \gamma_2\)), and Lemma 4.3.8 (the case \(\gamma_1 < cb < \gamma_2\)). Next we state these Lemmata, and we sketch their proofs.

**Lemma 4.3.6.** For every \(c \in (9/b, \gamma_1/b]\), where \(\gamma_1 \approx 9.06609\), the optimizers to \(f^{NLP}_b\) are \(k = r = b\), and \(s_b = -\sqrt{\frac{bc}{2c} - 10bc+9+bc-3}{2c}\). The induced competitive ratio is \(f(cb)\), (see definition of \(f(x)\) for \(x \leq \gamma_1\) in statement of Theorem 4.3.4), and the energy consumption, for instances \(d\), is \(f(cb)^{2d/c^2}\).

For proving Lemma 4.3.6, first we recall the known optimizer for the special case \(cb = 9\), and we identify the tight constraints. Requiring that the exact same inequality constraints to \(f^{NLP}_b\) remain tight, we ask how large can the product \(cb\) be so as to have KKT condition hold true. From the corresponding algebraic system, we obtain the answer \(cb \leq \gamma_1 \approx 9.06609\).

Similarly, from Theorem 4.3.3 we know the optimizers to \(f^{NLP}_b\) for large enough values of \(cb\), and the corresponding tight constraints to the NLP. Again, using KKT conditions, we show that the same constraints remain tight for the optimizers as long as \(cb \geq \gamma_2 \approx 11.3414\). This way we obtain the following Lemma.
Lemma 4.3.7. For every $c > \rho/b \approx 11.3414/b$, the optimal speeds of $N_{s,r,k}$ for $EE^b_c$ are $s = \sigma/c$, $r = \rho/c$, $k = \kappa/c$, i.e. they are the same as for $EE^\infty_c$. If the target is placed at distance $d$ from the origin, then the induced energy equals $584.738 \frac{d}{c}$. Moreover, the induced competitive ratio is 292.369, and is independent of $b,c$.

The case $\gamma_1 < cb < \gamma_2$ can be solved optimally only numerically, since the best speed values are obtained by roots to a high degree polynomial. Nevertheless, the following lemma proposes a heuristic choice of speeds (that of Theorem 4.3.4) which is surprisingly close to the optimal.

Lemma 4.3.8. The choices of $s, r, k$ of Theorem 4.3.4 when $\gamma_1 < cb < \gamma_2$ are feasible. Moreover, the induced competitive ratio is at most 0.03 additively off from the competitive ratio induced by the optimal choices of speeds (evaluated numerically).

The trick in order to find “good enough” optimizers to $f_{NLP}^b$ is to guess the subset of inequality constraints that remain tight when $\gamma_1 < cb < \gamma_2$. First, we observe that constraint $r \leq b$ is tight for the provable optimizers for all $c,b$ when $cb \in [9,\gamma_1] \cup [\gamma_2,\infty)$. As the only other constraint that switches from being tight to non-tight in the same interval is $k \leq b$, we are motivated to maintain tightness for constraints $r \leq b$ and the time constraint. Still the algebraic system associated with the corresponding KKT conditions cannot be solved analytically. To bypass this difficulty, and assuming we know (optimal) speed $s$, we use the tight time constraint to find speed $k$ as a function of $c,b,s$. From numerical calculations, we see that optimal speed $s$ is nearly optimal in $c$, and so we heuristically set $s = \alpha c + \beta$. We choose $\alpha, \beta$ so as to have $s$ satisfy optimality conditions for the boundary values $cb = \gamma_1, \gamma_2$. After we identify all parameters to our solution, we compare the value of our solution to the optimal one (obtained numerically), and we verify (using numerical calculations) that our heuristic solution is only by at most 0.03 additively off. The advantage of our analysis is that we obtain closed formulas for the speed parameters for all values of $cb \geq 9$.

4.4 Solving $EE^b_c$ with Unbounded-Memory Robots

In this section we prove Theorem 4.4.7, that is we solve $EE^b_c$ by assuming that the two robots have unbounded memory, and in particular that they can perform time and state dependent calculations and tasks. Note that, by scaling, our results hold for all $b, c$ for which $cb = 9$. For simplicity our exposition is for the natural case $c = 9$ and $b = 1$. Also, as before, $d$ will denote the unknown distance to the exit from the origin. Moreover, the exit is still assumed, for the purposes of performance analysis, to be at least 2 away from the origin.

Throughout the execution of our evacuation algorithm, robots can be in 3 different states (similar to the case of constant-memory robots). First, both robots start with the **Exploration State** and they remain in this until the exit is located. While in the exploration state, robots execute an elaborate exploration that requires synchronous movements in which robots, at a
high level, stay in good proximity, still they expand the searched space relatively fast. Then, the exit finder enters the *Chasing State* in which the robot, depending on its distance from the origin, calculates a speed at which to move in order to catch and notify the other robot. Lastly, when the two robots meet, they both enter the *Exit State* in which both robots move toward the exit with the smallest possible speed while meeting the time constraint.

Our algorithm takes as input the values of $c = 9$, $b = 1$, and use a speed value $s \leq b$, that will be chosen later. When the exit finder switches its state from Exploration to Chasing, it remembers the distance $d$ of the exit to the origin, as well as the value $k$ of a counter that was used while in the Exploration State. When the exit finder catches the other robot, they both switch to the Exit State, and they remember their distance $p$ from the origin, as well as the value of time $t$ that their rendezvous was realized. The speed of their Exit State will be determined as a function of $p, d, t$ (and hence of $s, c, b$ as well).

### 4.4.1 A Critical Component: $l$-Phase Explorations

We adopt the language of [40] in order to discuss a structural property that any feasible evacuation algorithm for EE$_9^1$ satisfies. As a result, the purpose of this section is to provide high level intuition for our evacuation algorithm that is presented in subsequent sections.

We refer to the two robots (starting exploration from the origin) as $L$ and $R$, intended to explore to the left and to the right of the origin, respectively. The robot trajectories can be drawn on the Cartesian plane where point-location $(x, -t)$ will correspond to point $x$ on the line being visited by some robot at time $t$. The following Theorem (due to [40]) was originally phrased for the time-evacuation unit-speed robots’ problem. We adopt the language of our problem.

**Theorem 4.4.1** ([40]). For any feasible solution to EE$_9^1$, the point-location of any robot lies within the cone spanned by vectors $\left(\begin{array}{c} -1 \\ -3 \end{array}\right)$, $\left(\begin{array}{c} 1 \\ -3 \end{array}\right)$.

Next we present some preliminaries toward describing our $k$-phase exploration algorithms. A **phase** is a pair $(s, r)$ where $s \in [0, 1]$ is a speed and $r \in \mathbb{R}$ is a distance ratio, possibly negative. An **$l$-phase algorithm** is determined by a position $p_0$ on the line and a sequence $S = (s_1, r_1), \ldots, (s_k, r_l)$ of $l$ phases (movement instructions). Whenever $r_i x < 0$, movement will be to the left, whereas $r_i x > 0$ will correspond to movement to the right. We will make sure that each time the loop is executed, position $x$ and corresponding time induce point-locations of the robots that lie in the boundary of the cone of Theorem [4.4.1]. If a loop starts at location $x$, then it takes additional time $\sum_{i \in [l]} \frac{|r_i x|}{s_i}$ to complete one iteration. We will be referring to quantity $1 + \sum_{i \in [l]} \frac{|r_i x|}{3 s_i}$ as the expansion factor of Exploration $S$. 

105
4.4.2 Algorithm $\mathcal{A}(s)$: The Exploration, Chasing and Exit States

In this section we give a formal description of our evacuation algorithm. The most elaborate part of it is when robots are in Exploration States, in which they will perform 3-phase exploration. It can be shown that 3-phase exploration based evacuation algorithms that do not violate the constraints of problem EE$^1$ have expansion factor at most 4. Moreover, among those, the ones who minimize the induced energy consumption makes robots move at speed 1 in the first and third phase.\footnote{Robot’s speed in the second phase will be denoted by $s$.}

We now present a specific 3-phase exploration algorithm, that we denote by $\mathcal{A}(s)$, complying with the above conditions, with phases $(-1, 1), (4s/(1 - s), s)$ and $(4 - 4s/(1 - s), 1), \text{where } s \text{ is an exploration speed to be determined later.}$ Robot $L$ will execute the 3-phase exploration with starting position -1, while robot $R$ with starting position 2. When subroutine $\text{travel}(v, p)$ is invoked, the robot sets its speed to $v$ and, from its current position, goes toward position $p$ on the line until it reaches it. We depict the trajectories of the robots while in the Exploration State in Figure 4.1.

\begin{figure}[h]
\centering
\begin{tabular}{ll}
\hline
\textbf{Exploration State of L} & \textbf{Exploration State of R} \\
1: $\text{travel}(1/3, -1);$ & 1: $\text{travel}(1/3, 2)$ \\
2: $k \leftarrow 0;$ & 2: $k \leftarrow 0$ \\
3: repeat & 3: repeat \\
4: $\text{travel}(1, 0);$ & 4: $\text{travel}(1, 0)$ \\
5: $\text{travel}(s, -4^{k+1} \cdot \frac{s}{1-s});$ & 5: $\text{travel}(s, 2 \cdot 4^{k+1} \cdot \frac{s}{1-s})$ \\
6: $\text{travel}(1, -4^{k+1});$ & 6: $\text{travel}(1, 2 \cdot 4^{k+1})$ \\
7: $k \leftarrow k + 1;$ & 7: $k \leftarrow k + 1$ \\
\hline
\end{tabular}
\caption{Algorithm $\mathcal{A}(s)$}
\end{figure}

\footnote{The proof of these facts is lengthy and technical, and is not required for the correctness of our algorithm, rather it only justifies some parameter choices}
Figure 4.1: A representation of position (x-axis, vertical dashed line is 0) and time (y-axis), and the trajectory followed by the two robots (solid lines). The two diagonal dashed lines form the “1/3 cone” of Theorem 4.4.1.

Figure 4.2: The robots’ behavior when the exit is found by $L$ is indicated by the bold line. In the first case (left), the catch-up speed is slower than 1 (and the rendezvous is realized at the turning point of the non-finder), whereas it is 1 in the second case (right).
HASING

STATE
1: \( K \leftarrow 4^k \)
2: if I am \( R \) then
3: \( K \leftarrow 2 \cdot 4^k \)
4: \( s' \leftarrow \min \left\{ \frac{d}{4K - d/s}, 1 \right\} \)
5: Travel toward the other robot at speed \( s' \) until meeting it at distance \( p \) from the origin, and at time \( t \).

EXIT STATE
1: \( \bar{s} \leftarrow \frac{p + d}{9d - t} \)
2: Go toward the exit with speed \( \bar{s} \).

A complete execution of one repeat loop within the Exploration State will be referred to as a \textit{round}. Variable \( k \) counts the number of completed rounds.

Each robot stays in the Exploration State till the exit it found. When switching to the Chasing state (which happens only for the exit finder), robot remembers its current value of counter \( k \), as well as the distance \( d \) of the exit to the origin. Based on these values (as well as \( s \)) it calculates the most efficient trajectory in order to catch the other robot (predicting, when applicable, that the rendezvous can be realized while the other robot is approaching the exit finder). When the rendezvous is realized, robots store their current distance \( p \) to the origin, as well as the time \( t \) that has already passed. Then, robots need to travel distance \( p + d \) to reach the exit. Knowing they have time \( 9d - t \) remaining, they go to the exit together as slow as possible to reach the exit in time exactly \( 9d \). Figure 4.2 provides an illustration of the behavior of the robots after finding the exit.

4.4.3 Performance Analysis & an Optimal Choice for Parameter \( s \)

In this section we are ready to provide the details for proving Theorem 4.4.7. Evacuation algorithm \( \mathcal{A}(s) \) is not feasible to \( \text{EE}_c^b \) for all values of speed parameter \( s \) (of the Exploration States). We will show later that trajectories induce evacuation time at most \( 9d \) only if \( s \in [1/3, 1/2] \). In what follows, and even though we have not fixed the value of \( s \) yet, we will assume that \( s \) has some value between \( 1/3 \) and \( 1/2 \). The purpose of this section is to fix a value for parameter \( s \), show that \( \mathcal{A}(s) \) is feasible to \( \text{EE}_o^b \), and subsequently compute the induced energy consumption and competitive ratio. As a reminder, each iteration of the repeat loop of the Exploration States is called a round, and \( k \) is a counter for these rounds.

\textbf{Proposition 4.4.2.} For every \( k \geq 0 \), and at the start of its \( k \)-th round,

\begin{itemize}
  \item robot \( L \) is at position \( -4^k \) at time \( 3 \cdot 4^k \), and
  \item robot \( R \), is at position \( 2 \cdot 4^k \) at time \( 6 \cdot 4^k \).
\end{itemize}

Let \( X \in \{L, R\} \) be one of the robots. We define \( K(X, k) = 4^k \) if \( X = L \), and \( K(X, k) = 2 \cdot 4^k \) if \( X = R \), i.e. the position of \( X \) at the start of round \( k \). We will often
We define

\[ \text{We may simply write } D_s s_4(4) \]

The speed \( s_t \) toward the exit after meeting for the cases \( d \) toward the exit after having found the exit, and set \( K := K(X, k) \). Then the following hold:

1. If \( d \in D_1, \) then \( p = 0 \) and \( t = 8K \).
2. If \( d \in D_2, \) then \( |p| = \frac{d + ds - 4Ks}{1-s} \) and \( t = 8K + \frac{d + ds - 4K}{1-s} \).
3. If \( d \in D_3, \) then \( |p| = 2ds/(1 - s) \) and \( t = 8K + 2d + 2ds/(1 - s) \).

Using the lemma above, we can now prove that \( A(s) \) meets the speed bound and the evacuation time bound.

**Lemma 4.4.3.** Suppose that robot \( X \in \{L, R\} \) finds the exit at distance \( d \) when its round counter has value \( k \). Let \( p \) and \( t \) be, respectively, the position and time at which \( X \) first meets with the other robot after having found the exit, and set \( K := K(X, k) \). Then the following hold:

1. If \( d \in D_1, \) then \( p = 0 \) and \( t = 8K \).
2. If \( d \in D_2, \) then \( |p| = \frac{d + ds - 4Ks}{1-s} \) and \( t = 8K + \frac{d + ds - 4K}{1-s} \).
3. If \( d \in D_3, \) then \( |p| = 2ds/(1 - s) \) and \( t = 8K + 2d + 2ds/(1 - s) \).

We may simply write \( D_1, D_2 \) and \( D_3 \) if \( K \) is clear from the context. Note that during the second phase of round \( K \), robot \( L \) explores \( D_1 \) and \( D_2 \), whereas \( D_3 \) is explored during the third phase. The same statement holds for \( R \). The following lemma will be useful in analyzing the worst case evacuation time and energy consumption of our algorithm.

**Lemma 4.4.4.** For any \( s \in [1/3, 1/2] \), evacuation algorithm \( A(s) \) is feasible to \( EE_1 \).

Lemma 4.4.3 allows us to derive the speed \( s_{b1}, s_{b2} \) and \( s_{b3} \) at which both robots go toward the exit after meeting for the cases \( d \in D_1, d \in D_2 \) and \( d \in D_3 \), respectively. We also know the speed \( s_{c1} \) at which the exit-finder catches up to the other robot when \( d \in D_1 \). We define

\[ s_{b1} := \frac{d}{9d - 8K}, \quad s_{c1} = \frac{d}{4K - d/s}, \quad s_{b2} := \frac{2d - 4Ks}{d(8 - 9s - 1/s) + 4K(2s - 1)}, \quad s_{b3} := \frac{d(1 + s)}{d(7 - 9s) + 8K(s - 1)} \]

The speed \( s_{b2} \) is a simple rearrangement of the speed \( \frac{d + qs}{9d - (8K + q)}, \) where \( q = \frac{d + ds - 4K}{1-s} \), and \( s_{b3} \) is obtained by rearranging \( \frac{d + 2ds/(1 - s)}{9d - (8K + 2d + 2ds/(1 - s))} \).

Next we compute the energy consumption. For given \( K, d \) and \( s \), denote by \( E_L(K, d, s) \) the energy spent by robot \( L \) from time 3 to time 9d when it exits. Similarly, \( E_R(K, d, s) \) is the energy spent by \( R \) from time 6 to time 9d. Then, the energy consumption is \( E(K, d, s) := \frac{1}{3} + E_L(K, d, s) + E_R(K, d, s) \). For any \( K \) and \( s \), we also define \( F(K, s) := (K - 1)(5 - 4s(s + 1)) \).
Lemma 4.4.5. Suppose that robot \( X \in \{ L, R \} \) finds the exit at distance \( d \) when its round counter has value \( k \), and let \( K := K(X, k) \). Then

\[
E(K, d, s) = \begin{cases} 
F(K, s) + 3K + d(s^2 + s_1^2 + 2s_2^2) & \text{if } d \in D_1 \\
\frac{1}{3} + F(K, s) + 3K + \left( \frac{2d - 4Ks}{1 - s} \right) (1 + s^2 + 2s_2^2) & \text{if } d \in D_2 \\
F(K, s) + 3K - 4Ks(s + 1) + \frac{2d}{1 - s} (s^3 + s_3^2(s + 1) + 1) & \text{if } d \in D_3.
\end{cases}
\]

Denote by \( E_i(k, d, s) \) the value of \( E(K, d, s) \) when \( d \in D_i, i = 1, 2, 3 \). Our intention now is to fix speed value \( s \) that solves the following Nonlinear Program

\[
\min_{s \in [1/3, 1/2]} \left\{ \max \left\{ \sup_{d \in D_1, k \geq 1, X} \frac{E_1(K, d, s)}{d}, \sup_{d \in D_2, k \geq 1, X} \frac{E_2(K, d, s)}{d}, \sup_{d \in D_3, k \geq 1, X} \frac{E_3(K, d, s)}{d} \right\} \right\}.
\]

For every \( s \in [1/3, 1/2] \) we show in Lemma 4.4.6 that \( \frac{E_1(K, d, s)}{d} \) is decreasing in \( d \in D_1 \), that \( \frac{E_2(K, d, s)}{d} \) is increasing in \( d \in D_2 \), and that \( \frac{E_3(K, d, s)}{d} \) is decreasing in \( d \in D_3 \). Then, the best parameter \( s \) can be chosen so as to make all worst case valued \( \frac{E_i(K, d, s)}{d} \) equal (if possible) when \( i = 1, 2, 3 \). The optimal \( s \) can be found by numerically finding the roots of a high degree polynomial, and accordingly, we heuristically set \( s = 0.39403 \), inducing the best possible energy consumption for algorithm \( \mathcal{A}(s) \). All relevant formal arguments are within the proof of the next lemma.

Lemma 4.4.6. On instance \( d \) of \( \text{EE}^1_3 \), algorithm \( \mathcal{A}(s) \) induces energy consumption at most \( 8.42588d \), when \( s = 0.39403 \).

By Lemma 4.4.6, we conclude that for the specific value of \( s \), algorithm \( \mathcal{A}(s) \) has competitive ratio \( \frac{341.24814}{8.42588} \approx 341.24814 \), concluding the proof of Theorem 4.4.7.

Theorem 4.4.7. For every \( c, b > 0 \) with \( cb = 9 \), there is an evacuation algorithm for unbounded-memory autonomous robots solving \( \text{EE}^b_c \) inducing energy consumption \( 8.42588b^2d \) for instances \( d \), and competitive ratio \( 341.24814 \).

4.5 Discussion

The main contribution of our paper was to introduce an energy consumption model appropriate to linear search and investigate how the F2F communication model affects time/energy trade-offs until completion of the search by two robots, considering two different computational capabilities for the robots. Our approach inspired new algorithms that take better account of the impact of the change in the speed of the robots during the course of the search.
and leads to better understanding through evaluation of trade-offs of the overall performance
of the algorithms.

Our paper raises several interesting problems worth investigating. In addition to im-
proving the trade-offs in the algorithms proposed, one may wish to pursue new avenues for
research by examining additional search domains, like the unit disk, in the spirit of [57].
It would also be natural to consider more realistic models of linear search with multiple
agents some of which may be faulty [70, 77]. Further, it would be interesting to investi-
gate randomized search algorithms in our setting as well as more general models in which the
energy loss experienced by a robot traveling a distance $x$ at constant speed $s$ is given by $s^a x$,
for some fixed positive exponent $a$.

References


[17] E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and
D. Pająk. “Linear search by a pair of distinct-speed robots”. In: Algorithmica 81.1


pp. 221–228.


[29] M. Blum and D. Kozen. “On the power of the compass (or, why mazes are easier to


characterization of the optimal solution”. In: Theoretical Computer Science (2015),
569:24–42.


4.A Figures

Figure 4.3: The competitive ratio of algorithm $N_{s,r,k}$ (vertical axis) for the entire spectrum of $cb \geq 9$ (horizontal axis). Red curve corresponds to the case $cb \leq \gamma_1$, blue curve to the case $cb \in (\gamma_1, \gamma_2)$ and green curve to the case $cb \geq \gamma_2$. The curve is continuous and differentiable for all $cb \geq 9$. 


Figure 4.4: Comparison between the competitive ratio achieved by using optimal speed parameters to $\tau_{NL\mathcal{P}}^b$ of Theorem 4.3.2 (calculated numerically using software) and the competitive ratio achieved by the choices of Theorem 4.3.4. The vertical axis is the difference of the competitive ratios, and the horizontal axis corresponds to the values of $cb \in (\gamma_1, \gamma_2)$ (for all other values of $cb$ the difference is provably 0).

4.B Observation 4.B.1

Observation 4.B.1. $EE^b_c$ is well defined for each $b, c > 0$ with $bc \geq 1$, and the optimal solution, given that instance $d$ is known, equals $\frac{2d}{c^2}$.

Proof of Observation 4.B.1. Given that the location of the exit is known, and by symmetry, it is immediate that both robots have the same optimal speed, call it $s$, and they move in the direction of the exit. The induced evacuation time is then $d/s$, and the induced evacuation energy is $2d \cdot s^2$. For a feasible solution we require that $d/s \leq c \cdot d$ and that $0 < s \leq b$, and hence, the optimal offline solution is obtained as the solution to $\min\{s^2 : 1/c \leq s \leq b\}$. For a feasible solution to exist, we need $bc \geq 1$. Moreover, it is immediate that the optimal choice is $s = 1/c$, inducing energy consumption $2d \cdot s^2 = 2d/c^2$.

4.C Proofs Omitted from Section 4.3

4.C.1 Lemma 4.3.1

Proof of Lemma 4.3.1. Consider the moment that the exit is located, after time $d/s$ time of searching. The robot that now chases the other speed-$s$ robot at constant speed $r > s$ will reach it after $2d/(r - s)$ time. To see this note that the configuration is equivalent to that the speed-$s$ robot is immobile and the other robot moves at speed $r - s$, having to traverse a
total distance of $2d$. Moreover, the speed-$s$ robot traverses an additional length $2ds/(r-s)$ segment till it is caught, being a total of $2ds/(r-s) + 2d$ away from the exit. Once robots meet, the walk to the exit at speed $k$, which takes additional time $(2ds/(r-s) + 2d)/k$. Overall the evacuation time equals

$$\frac{d}{s} + \frac{2d}{r-s} + \frac{2ds}{(r-s)} + \frac{2d}{k} = d \left( \frac{2(k + r)}{k(r-s)} + \frac{1}{s} \right).$$

Similarly we compute the total energy till both robots reach the exit. The energy spent by the finder is

$$d \cdot s^2 + \left( \frac{2ds}{r-s} + 2d \right) \cdot r^2 + \left( \frac{2ds}{r-s} + 2d \right) \cdot k^2,$$

while the energy spent by the non finder is

$$\left( d + \frac{2ds}{r-s} \right) \cdot s^2 + \left( \frac{2ds}{r-s} + 2d \right) \cdot k^2.$$

Adding the two quantities and simplifying gives the promised formula.

4.C.2 Theorem 4.3.2

Proof of Theorem 4.3.2 Note that in $f NLPl_c^\infty$ that aims to provide a solution to $EE_c^b$, constraints $s, r, k \leq \infty$ are simply omitted. In particular, the theorem above claims that when $b = \infty$, i.e. when speeds are unbounded, algorithm $N_{s,r,k}$ always admits some feasible solution. In what follows, we prove all claims of the theorem.

By Lemma 4.3.1, the energy performance of $N_{s,r,k}$ equals $2d \cdot E(s, r, k)$, and the induced evacuation time is $d \cdot T(s, r, k)$. For the values of $s, r, k$ to be feasible, we need that $0 < s, r, k \leq b$, that $r > k$ and that $d \cdot T(s, r, k) \leq cd$. Clearly the latter time constraint simplifies to the time constraint of $f NLPl_c^b$, while the objective value can be scaled by $d > 0$ without affecting the optimizers to the NLP, if such optimizers exist. Finally note that even though the strict inequalities become non strict inequalities in the NLP, speeds evaluations for which any of $s, r, k$ is 0 or $r = k$ violates the time constraint (for any fixed $c > 0$). Therefore, $f NLPl_c^b$ correctly formulates the problem of choosing optimal values for $N_{s,r,k}$ for solving $EE_c^b$.

The next two lemmas show that the Naive algorithm can solve problem $EE_c^b$ for the entire spectrum of $c, b$ values for which the problem admits solutions, as per Lemma 4.2.1

Lemma 4.C.1. For every $c$, problem $f NLPl_c^\infty$ admits an optimal solution.

Proof of Lemma 4.C.1 Consider the redundant constraints $s, r, k \geq 1/c$ that can be derived by the existing constraints of $f NLPl_c^\infty$ (note that if all speeds are not at least $1/c$ then clearly
the time constraint is violated). For the same reason, it is also easy to see that \( r - s \geq 1/c \), since again we would have a violation of the time constraint.

Next, it is easy to check that \( s = 7/c, r = 14/c, k = 7/c \) is a feasible solution, hence the NLP is not infeasible. The value of the objective for this evaluation is \( 686/c^2 \). But then, notice that the objective is bounded from below by \( s^2 + r^2 + 2k^2 \). Hence, if an optimal solution exists, constraints \( s, r, k \leq \sqrt{686}/c \) are valid for the optimizers. We may add these constraints to \( \gamma \text{NLP}_{c}^{\infty} \), resulting into a compact (closed and bounded) feasible region. But then, note that the objective is continuous over the new compact feasible region, hence from the extreme value theorem it attains a minimum.

\[ \text{Lemma 4.C.2.} \quad \text{There exist } s, r, k \text{ for which } N_{s,r,k} \text{ induces a feasible solution to } EE_{c}^{b} \text{ if and only if } c \geq 9/b. \]

\[ \text{Proof of Lemma 4.C.2.} \quad \text{Consider the problem of minimizing completion time of the Naive Algorithm, given that the speeds are all bounded above by } b. \text{ The corresponding NLP that solves the problem reads as.} \]

\[ \min \frac{2(k + r)}{k(r - s)} + \frac{1}{s} \quad (4.2) \]

\[ \text{s.t. } r \geq s \]

\[ 0 \leq s, r, k \leq b \]

Note that it is enough to prove that the optimal value to \((4.2)\) is \( 9/b \). Indeed, that would imply that no speeds exist that induce completion time less than \( 9/b \), making the corresponding feasible region of \( \gamma \text{NLP}_{c}^{b} \) empty if \( c < 9/b \).

Now we show that the optimal value to \((4.2)\) is \( 9/b \), by showing that the unique optimizers to the NLP are \( r = k = b \) and \( s = b/3 \). Indeed, note that

\[ \frac{\partial}{\partial r} \left( \frac{2(k + r)}{k(r - s)} + \frac{1}{s} \right) = -\frac{2(k + s)}{k(r - s)^2} \]

which is strictly negative for all feasible \( s, r, k \) with \( r > s \). Hence, there is no optimal solution for which \( r < b \), as otherwise by increasing \( r \) one could improve the value of the objective. Similarly we observe that

\[ \frac{\partial}{\partial k} \left( \frac{2(k + r)}{k(r - s)} + \frac{1}{s} \right) = -\frac{2r}{k^2(r - s)} \]

which is again strictly negative for all feasible \( s, r, k \) with \( r > s \). Hence, there is no optimal solution for which \( k < b \), as otherwise by increasing \( k \) one could improve the value of the objective.
To conclude, in an optimal solution to (4.2) we have that \( r = k = b \), and hence one needs to find \( s \) minimizing \( g(s, b, b) = \frac{4}{b-s} + \frac{1}{s} \). For this we compute
\[
\frac{\partial}{\partial s} g(s, b, b) = \frac{4}{(b-s)^2} - \frac{1}{s^2}
\]
and it is easy to see that \( g(s, b, b) = 0 \) if and only if \( s = b/3 \) or \( s = -b \) (and the latter is infeasible). At the same time, \( g(s, b, b) \) is convex when \( s \leq b \) because \( \frac{\partial^2}{\partial s^2} g(s, b, b) = \frac{8}{(b-s)^3} + \frac{2}{s^3} \geq 0 \), hence \( s = b/3 \) corresponds to the unique minimizer.

The last component of Theorem 4.3.2 that requires justification pertains to the competitive ratio. Now fix \( b, c > 0 \) for which \( bc \geq 9 \), and let \( E(s_0, r_0, k_0) \) be the optimal solution to \( fNLP_c^\infty \) (corresponding to the optimal choices of algorithm \( \mathcal{N}_{s, r, k} \)). By Lemma 4.3.1 the induced energy consumption is \( 2d \cdot E(s_0, r_0, k_0) \). Then, the competitive ratio of the algorithm is \( \sup_{d > 0} \frac{c^2 \cdot E(s_0, r_0, k_0)}{2d} \).

4.C.3 Theorem 4.3.3

Proof of Theorem 4.3.3. First we observe that \( s = \frac{\sigma}{c}, r = \frac{\rho}{c}, k = \frac{\kappa}{c} \) are indeed feasible to \( fNLP_c^\infty \) (for every \( c > 0 \)), since
\[
T\left(\frac{\sigma}{c}, \frac{\rho}{c}, \frac{\kappa}{c}\right) = c\left(\frac{2(\kappa + \rho)}{\kappa(\rho - \sigma)} + \frac{1}{\sigma}\right)
\]
and in particular, for the values of \( \sigma, \rho, \kappa \) described above we have \( \left(\frac{2(\kappa + \rho)}{\kappa(\rho - \sigma)} + \frac{1}{\sigma}\right) \approx 1 \) (from the formal definition of \( \sigma, \rho, \kappa \) that appears later, it will be clear that expression will be exactly equal to 1). Moreover, by Theorem 4.3.2 the competitive ratio of \( \mathcal{N}_{s, r, k} \) is
\[
c^2 \cdot E\left(\frac{\sigma}{c}, \frac{\rho}{c}, \frac{\kappa}{c}\right) = \frac{\rho (2\kappa^2 + \rho^2 + \sigma^2)}{\rho - \sigma},
\]
as claimed.

In the remaining of the section we prove that the choices for \( s, r, k \) of Theorem 4.3.3 are indeed optimal for \( \mathcal{N}_{s, r, k} \). First we establish a structural property of optimal speeds choices for \( \mathcal{N}_{s, r, k} \).

Lemma 4.C.3. For any \( c > 0 \), optimal solutions to \( fNLP_c^\infty \) satisfy constraint \( T(s, r, k) \leq c \) tightly.

Proof. Consider an optimal solution \( \bar{s}, \bar{r}, \bar{k} \). As noted before, we must have \( \bar{s}, \bar{r}, \bar{k} > 0 \) and \( \bar{r} > \bar{s} \), as otherwise the values would be infeasible.
Next note that the time constraint can be rewritten as

\[ k \geq \frac{2rs}{crs - cs^2 - r - s} \]

For the sake of contradiction, assume that the time constraint is not tight for \( \bar{s}, \bar{r}, \bar{k} \). Then, there is \( \epsilon > 0 \) so that \( \bar{s}, \bar{r}, \bar{k}' \) is a feasible solution, where \( \bar{k}' = \bar{k} - \epsilon > 0 \). But then, the objective value strictly decreases, a contradiction to optimality. \( \square \)

We will soon derive the optimizers to \( f_{NLP}^\infty \) using Karush-Kuhn-Tucker (KKT) conditions. Before that, we observe that solutions are scalable with respect to \( c \), which will also allow us to simplify our calculations.

**Lemma 4.C.4.** Let \( s', r', k' \) be the optimizers to \( f_{NLP}^\infty \) inducing optimal energy \( E \). Then, for any \( c \), the optimizers to \( f_{NLP}^c \) are \( \bar{s} = s'/c, \bar{r} = r'/c, \bar{k} = k'/c \), and the induced optimal energy is \( \frac{1}{c^2}E \).

**Proof.** Note that the triplet \( (s, r, k) \) is feasible to \( f_{NLP}^c \) (for a specific \( c \)) if and only if the triplet \( (c \cdot s, c \cdot r, c \cdot k) \) is feasible to \( f_{NLP}^\infty \). Moreover, it is straightforward that when speeds are scaled by \( c \), the induced energy is scaled by \( c^2 \). Hence, for every \( c > 0 \) there is a bijection between feasible (and optimal) solutions to \( f_{NLP}^c \) and \( f_{NLP}^\infty \). \( \square \)

We are therefore motivated to solve \( f_{NLP}^\infty \), and that will allow us to derive the optimizers for \( f_{NLP}^c \), for any \( c > 0 \).

**Lemma 4.C.5.** The optimal solution to \( f_{NLP}^\infty \) is obtained for

\[ s = \sigma \approx 2.65976, \ r = \rho \approx 11.3414, \ k = \kappa \approx 6.63709 \]

and the optimal NLP value is \( \frac{\rho(2\kappa^2 + \rho^2 + \sigma^2)}{\rho - \sigma} \approx 292.37 \).

**Proof.** By KKT conditions, we know that, necessarily, all minimizers of \( E(s, r, k) \) is a conical combination of tight constraints (for the optimizers). Lemma 4.C.3 asserts that \( T(s, r, k) = 1 \) has to be satisfied for all optimizers \( s, r, k \). At the same time, recall that, by the proof of Lemma 4.C.1 none of the constraints \( r \geq s \) and \( s, r, k \geq 0 \) can be tight for an optimizer. Hence, KKT conditions imply that any optimizer \( s, r, k \) satisfies, necessarily, the following system of nonlinear constraints

\[
-\nabla E(s, r, k) = \lambda \nabla T(s, r, k) \\
T(s, r, k) = 1 \\
\lambda \geq 0
\]
More explicitly, the first equality constraints is

\[
\begin{pmatrix}
    r - \frac{2r(k^2 + r^2)}{(r-s)^2} \\
    \frac{4k^2 s^2 - 2r^2 + 3s^2 + s^3}{(r-s)^2} \\
    \frac{4kr}{s-r}
\end{pmatrix}
= \lambda \begin{pmatrix}
    \frac{2(k+r) - \frac{1}{s^2}}{k(r-s)^2} \\
    -\frac{2(k+s)}{k(r-s)^2} \\
    \frac{2r}{k^2(s-r)}
\end{pmatrix}
\]

From the 3rd coordinates of the gradients, we obtain that \( \lambda = 2k^3 \), which directly implies that the dual multiplier \( \lambda \) preserves the correct sign for the necessary optimality conditions.

Hence, the original system of nonlinear constraints is equivalent to that

\[
\begin{align*}
    r - \frac{2r(k^2 + r^2)}{(r-s)^2} &= 2k^3 \left( \frac{2(k+r) - \frac{1}{s^2}}{k(r-s)^2} \right) \\
    -2k^2 s + 2r^3 - 3r^2 s - s^3 &= 4k^2 (k + s) \\
    \frac{2(k+r)}{k(r-s)} + \frac{1}{s} &= 1
\end{align*}
\]

Using software numerical methods, we see that the above algebraic system admits the following 3 real roots for \((s, r, k)\):

\[
\begin{align*}
(2.659764883844293, 11.341425445393606, 6.637089776204052) &\text{(multiplicity 1)} \\
(-0.6115006613361799, 0.47813267995355124, 1.0972211311317337) &\text{(multiplicity 2)}
\end{align*}
\]

Since also all speeds are nonnegative, we obtain the unique candidate optimizer

\((\sigma, \rho, \kappa) = (2.65976, 11.3414, 6.63709)\).

To verify that indeed \((\sigma, \rho, \kappa)\) is a minimizer, we compute

\[
\nabla^2 E(s, r, k) = \begin{pmatrix}
    4r \frac{k^2 + r^2}{(r-s)^3} & (r+s) \frac{-2k^2 + 3r^2 - s^2 - 4rs}{(r-s)^3} & \frac{4kr}{(r-s)^2} \\
    \frac{4k^2 s}{(r-s)^3} & \frac{2(r^3 - 3s^2 + 3s^2 r^2 + s^3 + 2k^2 s)}{(r-s)^3} & \frac{4ks}{(r-s)^2} \\
    \frac{4k}{(r-s)^2} & \frac{-4k}{(r-s)^2} & \frac{4r}{r-s}
\end{pmatrix}
\]

Moreover,

\[
\nabla^2 E(\sigma, \rho, \kappa) = \begin{pmatrix}
    11.9718 & -1.56333 & 3.99485 \\
    -1.56333 & 2.83125 & -0.936864 \\
    3.99485 & -0.936864 & 5.22546
\end{pmatrix}
\]

which has eigenvalues 14.1183, 3.41098, 2.49927, hence it is PSD. As a result, \(f(s, r, k)\) is locally convex at \((\sigma, \rho, \kappa)\), and therefore \((\sigma, \rho, \kappa)\) is a local minimizer to \(f_{\text{NLP}}\). As we showed earlier, \((\sigma, \rho, \kappa)\) is the only candidate optimizer, hence a global minimizer as well. \(\square\)
Lemma 4.C.5 together with Lemma 4.C.4 imply that for any \( c > 0 \) the optimal solution to \( f_{NLP}^\infty \) is exactly for
\[
s = \frac{\sigma}{c}, \quad r = \frac{\rho}{c}, \quad k = \frac{\kappa}{c}
\]
and hence, the proof of Theorem 4.3.3 follows. \( \square \)

### 4.C.4 Lemma 4.3.6

**Proof of Lemma 4.3.6.** An immediate corollary from the proof of Lemma 4.C.2 (within the proof of Theorem 4.3.2) is the following

**Corollary 4.C.6.** The unique solution to \( f_{NLP}^b \) when \( c = 9/b \) is given by
\[
r = k = b, \quad s = \frac{b}{3},
\]
inducing energy \( \frac{28b^2d}{3} \), and competitive ratio \( \frac{14(\alpha b)^2}{3} = 378 \).

Next we find solutions for \( c > 9/b \) so that \( r, k \leq b \) remain tight. Since, when \( c = 9/b \), there is only one optimizer \( s = b/3, r = b, k = b \), two inequality constraints are tight. The next calculations investigate the spectrum of \( c \) for which the same constraints remain tight for the optimizer.

We write 1st order necessary optimality conditions for \( f_{NLP}^b \), given that the candidate optimizer satisfies the time constraint, and the two \( r, k \leq b \) speed bound constraints tightly

\[
-\nabla E(s, r, k) = \lambda_1 \nabla T(s, r, k) + \lambda_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \lambda_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}
\]
\[
T(s, r, k) = c
\]
\[
r = b
\]
\[
k = b
\]
\[
\lambda_1, \lambda_2, \lambda_3 \geq 0
\]

From the tight time constraint, and solving for \( s \) we obtain that
\[
s_{1,2} = \frac{\pm \sqrt{b^2c^2 - 10bc + 9} + bc - 3}{2c}
\]

For each \( s \in \{s_1, s_2\} \), the first gradient equality defines a linear system over \( \lambda_1, \lambda_2, \lambda_3 \) whose solutions are
\[
\lambda_1 = \frac{(s - 3)s^2}{3s - 1}, \quad \lambda_2 = -\frac{5s^3 - 9s + 2}{(s - 1)(3s - 1)}, \quad \lambda_3 = -\frac{2(s^3 - 3s^2 - 6s + 2)}{(s - 1)(3s - 1)}.
\]
\[ \lambda_1 = \frac{bs^2(3b-s)}{b-3s}, \quad \lambda_2 = -\frac{2b^3 - 9b^2s - 5s^3}{(b-3s)(b-s)}, \quad \lambda_3 = -\frac{2(2b^3 - 6b^2s - 3bs^2 + s^4)}{(b-3s)(b-s)} \]

respectively. As long as all dual multiplies \( \lambda_i = \lambda_i(s) \) are positive, corresponding solution \((s, b, b)\) is optimal to \( R_c^r \), provided that \( \nabla^2 f(s, b, b) \succ 0 \).

First we claim that \( s_1 \) cannot be part of an optimizer. Indeed,

\[ \lambda_1(s_1) = \frac{-b \left( 5bc - \sqrt{(bc-9)(bc-1)} + 3 \right) \left( bc + \sqrt{(bc-9)(bc-1)} - 3 \right)^2}{4c^2 \left( bc + 3\sqrt{(bc-9)(bc-1)} - 9 \right)} \]

Recall that \( bc > 9 \), and hence the denominator of \( \lambda_1(s_1) \) as well as \( bc + \sqrt{(bc-9)(bc-1)} - 3 \) are strictly positive. But then, the sign of \( \lambda_1(s_1) \) is exactly the opposite of \( 5bc - \sqrt{(bc-9)(bc-1)} + 3 \). Define function \( h(x) := 5x - \sqrt{(x-9)(x-1)} + 3 \) over the domain \( x > 9 \). It is easy to verify that \( h(x) \) preserves positive sign (in fact \( \min_{x \geq 9} h(x) = h \left( \frac{5}{3} \left( \sqrt{6} + 3 \right) \right) = 8\sqrt{6} + 28 > 0 \) Hence, \( \lambda_1(s_1) < 0 \) that concludes our claim.

Next we investigate the sign of \( \lambda_1(s_2), \lambda_2(s_2), \lambda_3(s_2) \). For this, introduce function \( t(x) = \sqrt{(x-9)(x-1)} \), and note that

\[ \lambda_1(s_2) = \frac{-b \left( -bc + t(bc) + 3 \right)^2 (5bc + t(bc) + 3)}{4c^2 \left( bc - 3 \left( t(bc) + 3 \right) \right)}, \]
\[ \lambda_2(s_2) = \frac{30 \left( t(bc) + 3 \right) + bc \left( -3bc + 3t(bc) + 32 \right) - 5 \left( 5t(bc) + 11 \right)}{4c(bc - 9)}, \]
\[ \lambda_3(s_2) = \frac{bc \left( -23t(bc) + bc \left( -3bc + 3t(bc) + 22 \right) + 49 \right) - 12 \left( t(bc) + 3 \right)}{4c(bc - 9)} \]

Claim 1: \( \lambda_1(s_2) > 0 \) for all \( c > 9/b \). Define \( d_1(x) = x - 3(3 + t(x)) \) and \( d_2(x) = 3 + 5x + t(x) \). Note that \( sign(\lambda_1(s_2)) = -sign(d_1(bc)) \cdot sign(d_2(bc)) \). Simple calculus shows that \( d_1(x) \) is strictly decreasing in \( x \geq 9 \), and \( d_1(9) = 0 \), and therefore \( d_1(bc) < 0 \) for all \( c > 9/b \). Similarly, it is easy to see that \( d_2(x) \) is strictly increasing in \( x \geq 9 \), and \( d_2(9) = 45 \). Therefore \( d_2(bc) > 0 \) for all \( c > 9/b \). Overall this implies that \( \lambda_1(s_2) \) is positive for all \( c > 9/b \).

Claim 2: \( \lambda_2(s_2) > 0 \) for all \( c \in (9/b, 9.72307/b) \).

First we observe that the denominator of \( \lambda_2(s_2) \) preserves positive sign for \( c > 9/b \). So we
We conclude that $\lambda_3(x) = 30(3 + t(x)) + x(32 - 3x + 3t(x)) - 5(11 + 5t(x)))$. Note that $d_3(x) = 0$ is equivalent to that
\[
\left(3x^2 - 25x + 30\right) t(x) - \left(3x^3 - 32x^2 + 55x - 90\right) = 0
\]
\[
\Leftrightarrow \left(3x^2 - 25x + 30\right)^2 t^2(x) - \left(3x^3 - 32x^2 + 55x - 90\right)^2 = 0
\]
\[
\Leftrightarrow - 8(x - 9)x(3x(2x - 25) + 60) - 175 = 0
\]
Degree-3 polynomial $3x(2x - 25) + 60 - 175$ has only one real root, which is
\[
\frac{1}{18} \left(3 \sqrt[3]{5 \left(192 \sqrt{10} + 1055\right)} + \sqrt[3]{142425 - 25920 \sqrt{10} + 75}\right) \approx 9.72307
\]
Hence, $\lambda_2(s_2) > 0$ for all $c \in (9/b, 9.72307/b)

Claim 3: $\lambda_3(s_2) > 0$ for all $c \in (9/b, 9.06609/b)$.
First we observe that the denominator of $\lambda_3(s_2)$ preserves positive sign for $c > 9/b$. So we focus on the sign of the numerator we abbreviate by $d_4(x) = x(3t(x) - 3x + 22) - 23t(x) + 49) - 12(t(x) + 3)$. Note that $d_4(x) = 0$ is equivalent to that
\[
\left(3x^2 - 23x - 12\right) t(x) - \left(3x^3 - 22x^2 - 49x + 36\right) = 0
\]
\[
\Leftrightarrow \left(3x^2 - 23x - 12\right)^2 t^2(x) - \left(3x^3 - 22x^2 - 49x + 36\right)^2 = 0
\]
\[
\Leftrightarrow - 16(x - 9)x(3x(2x - 9)x - 3) + 49 = 0
\]
The roots of degree-3 polynomial $3x(2x - 9)x - 3) + 49$ are
\[
\gamma_1 = 3 + \sqrt{38} \cos \left(\frac{1}{3} \tan^{-1} \left(\frac{127}{151 \sqrt{2}}\right)\right) \approx 9.06609
\]
\[
\gamma' = 3 + \sqrt{\frac{57}{2}} \sin \left(\frac{1}{3} \tan^{-1} \left(\frac{127}{151 \sqrt{2}}\right)\right) + \sqrt{\frac{19}{2}} \left(\cos \left(\frac{1}{3} \tan^{-1} \left(\frac{127}{151 \sqrt{2}}\right)\right)\right)
\]
\[
\approx 0.916629
\]
\[
\gamma'' = 3 - \sqrt{\frac{57}{2}} \sin \left(\frac{1}{3} \tan^{-1} \left(\frac{127}{151 \sqrt{2}}\right)\right) + \sqrt{\frac{19}{2}} \left(\cos \left(\frac{1}{3} \tan^{-1} \left(\frac{127}{151 \sqrt{2}}\right)\right)\right)
\]
\[
\approx -0.982723
\]
We conclude that $\lambda_3(s_2)$ preserves positive sign for all $c \in (9/b, \gamma_1/b)$.
Overall, we have shown that feasible solution $s_0 = -\sqrt{b^2 + 19 + 9 + bc - 3}$, $r_0 = k_0 = b$ satisfies necessary 1st order optimality conditions. We proceed by checking that $s_0, r_0, k_0$
satisfy 2nd order sufficient conditions, which amounts to showing that $\nabla^2 f(s_0, b, b) \succ 0$.
Indeed,

$$\nabla^2 f(s_0, b, b) = \frac{b^3}{(b - s_0)^3} \begin{pmatrix}
8 (b + s_0)(b^2 + 4s_0b - s_0^2) & -\frac{b^3}{4} & 4 - \frac{4s_0}{b} \\
\frac{b^3}{2} & 2(b - s_0)^2 & \frac{b^3}{4} \\
4 - \frac{4s_0}{b} & \frac{b^3}{4} & 4 - \frac{4s_0}{b} \\
\end{pmatrix} \left(\begin{array}{c}
\frac{8}{b^3} (b + s_0)(b^2 + 4s_0b - s_0^2) \\
\frac{b^3}{2} \\
4 - \frac{4s_0}{b} \\
\end{array}\right)$$

By setting $q := \frac{s_0}{b} = \frac{-\sqrt{b^2c^2 - 10bc + 9} + bc - 3}{2bc}$, we obtain the simpler form

$$\nabla^2 f(s_0, b, b) = \frac{b^3}{(b - s_0)^3} \begin{pmatrix}
8 (q - 1) (-q^2 + 4q + 1) & \frac{(-q - 1)(-q^2 + 4q + 1)}{4 - 4q} & \frac{4 - 4q}{4(q - 1)q} \\
\frac{b^3}{4} & \frac{2(q^3 + 3q^2 - q + 1)}{4(q - 1)q} & \frac{4(q - 1)q}{(1 - q)^2} \\
4 - \frac{4s_0}{b} & \frac{b^3}{4} & 4 - \frac{4s_0}{b} \\
\end{pmatrix} \left(\begin{array}{c}
\frac{8}{b^3} (q - 1) (-q^2 + 4q + 1) \\
\frac{b^3}{4} \\
4 - \frac{4s_0}{b} \\
\end{array}\right)$$

When $bc > 9$ we have that $q < 1/3$, $q$ is decreasing in the product of $bc > 9$, and it remains positive. The eigenvalues of the matrix that depends only on $q$ and for any $q \in (0, 1/3]$ can be obtained using a closed formula (they are real roots of a degree-3 polynomial). In Figure 4.5 we depict their behavior. Since all eigenvalues are all positive, the candidate optimizer is indeed a minimizer.

![Figure 4.5: The eigenvalues of matrix (4.3) as a function of $q \in (0, 1/3]$ (and scaled by $b^3/(b - s_0)^3$).](image)

### 4.C.5 Lemma 4.3.7

**Proof of Lemma 4.3.7** By Theorem 4.3.3, we know the optimizers to $f_{NLP}^\infty_c$: $s = \sigma/c$, $r = \rho/c$, $k = \kappa/c$. These optimizers satisfy the speed bound constraints $s, r, k \leq b$ as long as $\max\{s, r, k\} \leq b$, i.e. $\rho/c \leq b$. Hence, when $c \geq \rho/b$, Non Linear Programs $f_{NLP}^\infty_c$, $f_{NLP}^b_c$ have the same optimizers.

124
4.C.6 Lemma 4.3.8

Proof of Lemma 4.3.8. First, we observe that constraint $r \leq b$ is tight for the provable optimizers for all $c, b$ when $cb \in [9, \gamma_1] \cup \{\gamma_2\}$. As the only other constraint that switches from being tight to non-tight in the same interval is $k \leq b$, we are motivated to maintain tightness for constraints $r \leq b$ and the time constraint.

Given that speed $s$ is chosen (to be determined later), we fix $r = b$, and we set $k = k_2(c,b)$ where

$$k_2(c,b) := \frac{2bs}{bcs - b - cs^2 - s}$$

so as to satisfy the time constraint tightly (by solving $T(s,r,b) = c$ for $k$). It remains to determine values for speed $s$. To this end, we heuristically require that $s = s_2(c,b)$ where

$$s_2(c,b) := \alpha \cdot c + \beta$$

for some constants $\alpha, \beta$ that we allow to depend on $b$. In what follows we abbreviate $s_2(c,b)$ by $s_2(c)$. Let $s_1(c), s_3$ be the chosen values for speed $s$ as summarized by the statement of Theorem 4.3.4 when $cb \leq \gamma_1$ and $cb \geq \gamma_2$, respectively. We require that

$$s_2(\gamma_1/b) = s_1(\gamma_1/b), \quad s_2(\gamma_2/b) = s_3(\gamma_2/b)$$

inducing a linear system on $\alpha, \beta$. By solving the linear system, we obtain

$$\alpha = \frac{b^2 \left(-2\gamma_1\sigma + \gamma_2\gamma_1 - \sqrt{\gamma_1^2 - 10\gamma_1 + 9\gamma_2 - 3\gamma_2}\right)}{2\gamma_1 \left(\gamma_1 - \gamma_2\right) \gamma_2}$$

$$\beta = \frac{b \left(-2\gamma_1^2\sigma + \gamma_2^2\gamma_1 - \sqrt{\gamma_1^2 - 10\gamma_1 + 9\gamma_2^2 - 3\gamma_2^2}\right)}{2\gamma_1 \gamma_2 \left(\gamma_2 - \gamma_1\right)}$$

Using the known values for $\gamma_1, \gamma_2, \sigma$, we obtain $s_2(c) = 0.532412b - 0.0262661b^2c$, as promised. It remains to argue that $s_2(c,b)$, together with $r = b$, and $k_2(c,b)$ are feasible when $\gamma_1 < cb < \gamma_2$.

The fact that $s_2(c)$ complies with bounds $0 \leq s \leq b$ follows immediately, since $s_2(c)$ is a linear strictly decreasing function in $c$, and both $s_2(\gamma_1/b), s_2(\gamma_2/b)$ satisfy the bounds by construction. We are therefore left with checking that $0 \leq k_2(c,b) \leq b$ which is equivalent to that

$$bcs - b - cs^2 - 3s \geq 0$$

$$\Leftrightarrow b(bc(b(0.00170268 - 0.000689908bc) + 0.327748) - 1.59724) \geq 0$$

Define degree-2 polynomial function $g(x) = x(x(0.00170268 - 0.000689908x) + 0.327748) - 1.59724$ and observe that it suffices to prove that $g(x) \geq 0$ for all $x \in (\gamma_1, \gamma_2)$. The roots
of $g$ can be numerically computed as $-22.8094, 5.0074, 20.2699$, proving that $g$ preserves positive sign in $(\gamma_1, \gamma_2)$ as wanted.

Finally, the claims regarding the induced energy and competitive ratio is implied by Theorem 4.3.2 and obtained by evaluating the given choices of $s, r, k$ in $E(s, r, k)$.  

4.D  **Proofs Omitted from Section 4.4.**

4.D.1  **Proposition 4.4.2**

**Proof of Proposition 4.4.2** The exploration algorithm explicitly ensures that round $k - 1$ ends (and hence that round $k$ begins) at the claimed position, so only the time needs to be proved. Note that the statement is true for both robots when $k = 0$. Suppose that when $L$ starts its $(k - 1)$-th round, it is at position $-4k - 1$ at time $3 \cdot 4k - 1 + 4k - 1 + \frac{s}{1-s} (4k - 4k - 1, \frac{s}{1-s}) = 3 \cdot 4k$, and the $k$-th round will start at the claimed time. The proof is identical for $R$.  

4.D.2  **Lemma 4.4.3**

**Proof of Lemma 4.4.3**  Case 1: $d \in D_1$. Let $Y = \{L, R\} \setminus \{X\}$ be the robot other than $X$. Observe that by Proposition 4.4.2, $X$ finds the exit at time $3K + K + d/s = 4K + d/s$. Then $X$ goes towards $Y$ at speed $s' = \frac{d}{4K - d/s}$, and so it will reach position 0 at time $4K + d/s + \frac{4K - d/s}{d}d = 8K$. We know that at time $6K$, robot $Y$ is starting a round at $2K$ or $-2K$, then goes towards 0 at full speed. Hence $Y$ is at position 0 at time $8K$, where it meets $X$.

*Case 2:* $d \in D_2 = [4Ks/(s + 1), 4Ks/(1 - s)]$. As before, the exit is found at time $4K + d/s$. Assume for simplicity that $X = L$ (the case $X = R$ is identical by symmetry). After finding the exit at position $-d$, $L$ goes full speed to the right. Thus at time $t = 4K + d/s + d + \frac{d + ds - 4Ks}{1-s}$, it arrives at position $\frac{d + ds - 4Ks}{1-s}$. We show that at this time, $R$ is in its second phase and is at this position. Notice that

$$t = 4K + d/s + d + \frac{d + ds - 4Ks}{1-s} = 8K + d/s + d - 4K + \frac{d + ds - 4Ks}{1-s} \leq 8K(1 + s/(1 - s))$$

the latter inequality being obtained from $d \leq 4Ks/(1 - s)$. Now, $R$ enters its second *travel* phase when at position 0 at time $8K$, and the phase ends a time $8K + 1/s \cdot 8Ks/(1 - s) = 8K(1 + 1/(1 - s))$. Since $s \leq 1/2$, we get $8K(1 + 1/(1 - s)) \geq 8K(1 + s/(1 - s)^2) \geq t$. Therefore $R$ is still in its second phase at time $t$, and it follows that its position at this time
is \( s(t - 8K) = \frac{d + ds - 4Ks}{1 - s} \). Hence \( L \) and \( R \) meet. It is straightforward to see that \( L \) and \( R \) could not have met before time \( t \), and thus \( L \) and \( R \) meet at the claimed time and position.

**Case 3:** \( d \in D_3 = [4Ks/(1-s), 4K] \). Again assume that \( X = L \). This time \( L \) finds the exit at position \( -d \) at time \( 3K + K + \frac{1}{s} \cdot \frac{4Ks}{1-s} + d - \frac{4Ks}{1-s} = 8K + d \). Going full speed to the right, at time \( t = 8K + 2d + 2ds/(1-s) \), it reaches position \( 2ds/(1-s) \). Since \( d \leq 4K \), we have \( t \leq 8K(2+s/(1-s)) = 8K(1 + 1/(1-s)) \). As in the previous case, the second phase of \( R \) ends at time \( 8K(1 + 1/(1-s)) \geq t \). Thus at time \( t \), \( R \) is at position \( s(t - 8K) = 2ds + 2ds^2/(1-s) = 2ds/(1-s) \). Again, one can check that \( L \) and \( R \) could not have met before, which concludes the proof.

**4.D.3 Lemma 4.4.4**

**Proof of Lemma 4.4.4** Let \( X \) be the robot that finds the exit at distance \( d \), and let \( K := K(X, k) \). We show that all speeds are at most 1, as well as that the evacuation time is at most \( 9d \). There are three cases to consider.

**Case 1:** \( d \in D_1 \). By Lemma 4.4.3, after meeting, both robots need to travel distance \( d \) and have \( 9d - 8K \) time remaining. By the last line of the exit phase algorithm, they go back at speed \( s_{b1} := \frac{d}{9d - 8K} \) and make it in time, provided that speed \( s_{b1} \) is achievable, i.e. \( 0 < s_{b1} \leq 1 \). Clearly \( s_{b1} > 0 \) since \( 9d - 8K > 0 \). If we assume \( \frac{d}{9d - 8K} > 1 \), we get \( d > 9d - 8K \), leading to \( d < K \), a contradiction.

**Case 2:** \( d \in D_2 \). By Lemma 4.4.3, the robots meet at position \( p \) such that \( |p| = \frac{d + ds - 4Ks}{1-s} \). The robots use the smallest speed \( s_{b2} := \frac{p + d}{1-9d} \) that allows the them to reach the exit in time \( 9d \). We must check that \( 0 < s_{b2} \leq 1 \). We argue that if the two robots used speed 1 to get to the exit after meeting, they would make it before time \( 9d \). Since \( s_{b2} \) allows the robots to reach the exit in time exactly \( 9d \), it follows that \( 0 < s_{b2} \leq 1 \).

First note that since \( d \leq 4Ks/(1-s) \), we have \( 4K \geq d(1-s)/s \). Using speed 1 from the point they meet, the robots would reach the exit at time

\[
8K + \frac{d + ds - 4K}{1-s} + d + \frac{d + ds - 4Ks}{1-s} = 4K \left( 2 - \frac{1 + s}{1-s} \right) + d \left( 1 + \frac{2 + s + 1/s}{1-s} \right) = 4K \cdot \frac{1 - 3s}{1-s} + d \cdot \frac{3s + 1}{s(1-s)}
\]

\[
\leq d \cdot \frac{1 - s}{s} \cdot \frac{1 - 3s}{1-s} + d \cdot \frac{3s + 1}{s(1-s)} = d \left( \frac{1 - 3s}{1-s} + \frac{3s + 1}{s(1-s)} \right)
\]

where we have used the fact that \( 1 - 3s \leq 0 \) in the inequality. It is straightforward to show that \( d \left( \frac{1-3s}{1-s} + \frac{3s+1}{s(1-s)} \right) \leq 9d \) when \( 1/3 \leq s \leq 1/2 \), proving our claim.

127
Case 3: $d \in D_3$. Again according to Lemma 4.4.3, the robots meet at position $p$ satisfying $|p| = 2ds/(1 - s)$ at time $t = 8K + 2d + 2ds/(1 - s)$. The robots go towards the exit at speed $s_{c3} := \frac{d + ds}{9d - t}$. As in the previous case, we show that $s_{c3}$ is a valid speed by arguing that the robots have enough time if they used their full speed. If they do use speed $1$ after they meet, they reach the exit at time $t' = 8K + 3d + 4ds/(1 - s)$. Since $d \geq 4Ks/(1 - s)$, we have $K \leq d(1-s)/(4s)$. Therefore $t' \leq 8 \cdot (d(1-s)/(4s)) + 3d + 4ds/(1 - s) = d \cdot \frac{3s^2-s+2}{s(1-s)}$. One can check that this is $9d$ or less whenever $1/3 \leq s \leq 1/2$.

4.D.4 Lemma 4.4.5

Proof of Lemma 4.4.5 For any $K' := 4^i$ power of 4 with $i \geq 1$, define $B_L(K', s)$ as the energy spent by $L$ after reaching position $-K'$ for the first time without having found the exit, ignoring the initial $1/9$ energy spent to get to position 1. The quantity $B_L(K', s)$ is the sum of energy spent in each of the first $i - 1$ rounds, and so

$$B_L(K', s) = \sum_{j=0}^{i-1} \left( 4^j + s^2(4^{j+1}s/(1 - s)) + 4^{j+1} - 4^{j+1} s/(1 - s) \right)$$

$$= \sum_{j=0}^{i-1} \left( 4^{j+1}(5/4 + (s^3 - s)/(1 - s)) \right)$$

$$= 4/3(4^i - 1)(5/4 - s(s + 1))$$

$$= 1/3(K' - 1)(5 - 4s(s + 1))$$

We define $B_R(2K', s)$ similarly for $R$, i.e. $B_R(2K', s)$ is the energy spent by $R$ when its $(i - 1)$-th round is finished and it reached position $2K'$ for the first time, ignoring the initial $2/9$ energy to get to position 2. We get

$$B_R(2K', s) = \sum_{j=0}^{i-1} \left( 2 \cdot 4^j + s^2(2 \cdot 4^{j+1}s/(1 - s)) + 2 \cdot 4^{j+1} - 2 \cdot 4^{j+1} s/(1 - s) \right)$$

$$= 2B_L(K', s) = 2/3(K' - 1)(5 - 4s(s + 1))$$

We may now calculate the three possible cases of energy. Assume that $X \in \{L, R\}$ finds the exit and $Y = \{L, R\} \setminus \{X\}$. Observe that

$$B_X(K, s) + B_Y(2K, s) = (K - 1)(5 - 4s(s + 1)) = F(K, s)$$

We implicitly use Lemma 4.4.3 for the distance traveled by $X$ to catch up to $Y$ after finding the exit, and the distance traveled back by both robots. In the $E(K, d, s)$ expressions that follow, for clarity we partition the terms into 3 brackets, which respectively represent
the energy spent by $X$ to find the exit and catch up to $Y$, the energy spent by $Y$ before being caught, and the energy spent by both robots to go to the exit.

**Case 1:** $d \in D_1$. The total energy spent is
\[
E(K, d, s) = [B_X(K, s) + K + s^2 d + s_{c1}^2 d] + [B_Y(2K, s) + 2K] + [2s_{b1}^2 d]
\]
\[
= F(K, s) + 3K + d(s^2 + s_{c1}^2 + 2s_{b1}^2)
\]

**Case 2:** $d \in D_2$. In this case, the energy spent is
\[
E(K, d, s) = [B_X(K, s) + K + s^2 d + d + \frac{d + ds - 4Ks}{1 - s}] + [B_Y(2K, s) + 2K + s^2 \left( \frac{d + ds - 4Ks}{1 - s} \right)] + [2s_{b2}^2 \left( d + \frac{d + ds - 4Ks}{1 - s} \right)]
\]
\[
= F(K, s) + 3K + \left( \frac{2d - 4Ks}{1 - s} \right) (1 + s^2 + 2s_{b2}^2)
\]

**Case 3:** $d \in D_3$. The energy spent is
\[
E(K, d, s) = [B_X(K, s) + K + s^2 4Ks/(1 - s) + d - 4Ks/(1 - s) + d + 2ds/(1 - s)] + [B_Y(2K, s) + 2K + s^2 (2ds/(1 - s))] + [2s_{b3}^2 (d + 2ds/(1 - s))]
\]
\[
= F(K, s) + 3K + 4Ks/(1 - s)(s^2 - 1) + d \left( \frac{s}{1 - s} \left( 2 + 2s^2 + 2s_{b3}^2 \right) + 2 + 2s_{b3}^2 \right)
\]
\[
= F(K, s) + 3K - 4Ks(s + 1) + \frac{2d}{1 - s}(s^3 + s_{b3}^2(s + 1) + 1)
\]

### 4.D.5 Lemma 4.4.6

**Proof of Lemma 4.4.6** We analyze each case separately.

**Case 1:** $d \in D_1 = [K, 4Ks/(s + 1)]$. According to Lemma 4.4.5 we have

\[
E(K, d, s)/d = 1/d \cdot ((K - 1)(5 - 4s(s + 1)) + 3K + d(s^2 + s_{c1}^2 + 2s_{b1}^2))
\]
\[
= \frac{(K - 1)(5 - 4s(s + 1))}{d} + s^2 + \left( \frac{d}{4K - d/s} \right)^2 + 2 \left( \frac{d}{9d - 8K} \right)^2
\]
Consider the case \( d = K \). Plugging in \( s = 0.39403 \), the above evaluates to \( 8.4258714091 - 2.8028414364/K \). We claim that \( E(K, d, s)/d \) is a decreasing function over interval \( D_1 \), and therefore attains its maximum when \( d = K \). Assuming this is true, adding the initialization energy of \( 1/3 \) omitted so far and given that \( d \geq K \), the energy ratio is at most

\[
8.4258714091 - 2.8028414364/K + 1/(3K) \leq 8.42588
\]

We now prove that \( E(K, d, s)/d \) is decreasing over the interval \( D_1 \). Let

\[
\begin{align*}
  f_1(K, d, s) &:= \frac{d^2}{(9d - 8K)^2} \\
  f_2(K, d, s) &:= \frac{d^2}{(d - 4Ks)^2} \\
  f_3(K, d, s) &:= -4K \frac{(s^2 + s - 2) + 4s(s + 1) - 5}{d}
\end{align*}
\]

and observe that

\[
E(K, d, s) \frac{d}{d} = 2f_1(K, d, s) + s^2f_2(K, d, s) + f_3(K, d, s) + s^2.
\]

The plan is to prove that

\[
\frac{\partial}{\partial d} E(K, d, s)/d < 0.
\]

For this we calculate

\[
\begin{align*}
  \frac{\partial}{\partial d} f_1(K, d, s) &= -\frac{16dK}{(9d - 8K)^3} \\
  \frac{\partial}{\partial d} f_2(K, d, s) &= -\frac{8dKs}{(d - 4Ks)^3} \\
  \frac{\partial}{\partial d} f_3(K, d, s) &= \frac{4K (s^2 + s - 2) - 4s(s + 1) + 5}{d^2}
\end{align*}
\]

Now we claim that all \( \frac{\partial}{\partial d} f_i(K, d, s) \) are increasing functions in \( d \), for \( i = 1, 2, 3 \). Indeed, first,

\[
\frac{\partial^2}{\partial d^2} f_1(K, d, s) = \frac{32K (9d + 4K)}{(9d - 8K)^4} > 0
\]

since \( d \geq K \). Hence \( \frac{\partial}{\partial d} f_1(K, d, s) \) is increasing in \( d \). Second,

\[
\frac{\partial^2}{\partial d^2} f_2(K, d, s) = \frac{16Ks(d + 2Ks)}{(d - 4Ks)^4}
\]
is positive (and well defined), since \( d \leq 4ks/(1 + s) \). Hence \( \frac{\partial}{\partial d} f_2(K, d, s) \) is increasing in \( d \).

Third, we show that \( \frac{\partial}{\partial d} f_3(K, d, s) \) is increasing in \( d \). For this it is enough to prove that
\[
4K (s^2 + s - 2) - 4s(s + 1) + 5 < 0
\]
For \( s = 0.39403 \) (and in fact for all \( s \in (-2, 1) \)) the strict inequality can be written as \( K \geq \frac{4s^2 + 4s - 5}{4s^2 + 4s - 8} \), which we show next it is satisfied. Indeed, it is easy to see that \( \frac{4s^2 + 4s - 5}{4s^2 + 4s - 8} \leq 5/8 \) (which is attained for \( s = 0 \)), while \( K \geq 4 \), hence the claim follows.

To resume, we showed that \( \frac{\partial}{\partial d} f_i(K, d, s) \) are increasing functions in \( d \), for \( i = 1, 2, 3 \). Recalling that \( s = 0.39403 \), and since \( d \leq 4ks/(1 + s) \), we obtain that

\[
\frac{\partial}{\partial d} E(K, d, s)/d \\
\leq 2 \frac{\partial}{\partial d} f_1(K, 4Ks/(1 + s), s) + s^2 \frac{\partial}{\partial d} f_2(K, 4Ks/(1 + s), s) + \frac{\partial}{\partial d} f_3(K, 4Ks/(1 + s), s) \\
= \frac{1}{16K^2 s^2 (7s - 2)^3} \left[ (s + 1)^2 (s(4K(s(s + 1)(49s(7s - 6) + 76) - 8) \\
- s(7s(28s(7s + 1) - 365) + 1774) + 452) - 40) \right] \\
= \frac{2.19262 - 1.79465K}{K^2}
\]

Since \( K \geq 4 \), the latter quantity is clearly negative. This shows that \( \frac{\partial}{\partial d} E(K, d, s)/d \) is negative (in the given domain), hence \( E(K, d, s)/d \) is decreasing in \( d \).

**Case 2:** \( d \in D_2 = [4Ks/(s + 1), 4Ks/(1 - s)] \). In this case, the energy ratio \( E(K, d, s)/d \) is

\[
1/d \cdot \left( 1/3 + (K - 1)(5 - 4s(s + 1)) + 3K + \left( \frac{2d - 4Ks}{1 - s} \right) (1 + s^2 + 2s^2_{b2}) \right) \\
= \frac{(K - 1)(5 - 4s(s + 1)) + 3K}{d} \\
+ \left( \frac{2d - 4Ks}{d(1 - s)} \right) \left( 1 + s^2 + 2 \left( \frac{2d - 4Ks}{d(8 - 9s - 1/s)} + 4K(2s - 1) \right) \right)
\]

We will show that this expression achieves its maximum at \( d = 4Ks/(1 - s) \). When \( s = 0.39403 \), then above yields \( 8.425786060 - 1.0776069241/K \). Given that \( 1/(3d) \leq 1/(3.39186K) \), this implies that the energy ratio is at most

\[
8.425786060 - 1.0776069241/K + 1/(3.39186K) \leq 8.42588
\]

We prove that \( E(K, d, s)/d \) is an increasing function over interval \( D_2 \). First we compute \( \frac{\partial}{\partial d} E(K, d, s)/d \) and we substitute \( s = 0.39403 \) to find \( \frac{1}{d^2(d - 0.442471K)^2} g(K, d) \), where
\[ g(K, d) := d^3(7.8438K + 2.80279) + d^2K(-15.5674K - 3.72046) \\
+ dK^2(8.91957K + 1.6462) + K^3(-1.31555K - 0.242798). \]

Note that \( d \geq 4Ks/(1 + s) \approx 1.13064K \), and hence \( d^2(d - 0.442471K)^3 > 0 \) for all values of \( d \) under consideration. Therefore the lemma will follow if we show that \( g(K, d) \geq 0 \) as well.

\( g(K, d) \) is a degree-3 polynomial with positive leading coefficient. It attains a local minimum at the largest real root of

\[
\frac{\partial}{\partial d} g(K, d) = 3d^2(7.8438K+2.80279)+2dK(-15.5674K-3.72046)+K^2(8.91957K+1.6462)
\]

which is

\[
d_0(K) := K \left( 0.661559K + 0.0212482\sqrt{K(129.818K + 8.39821) + 0.158106} \right) / (K + 0.357325)
\]

Now we observe that for all \( K > 0 \), we have \( d_0(K) < 4Ks/(1 + s) \approx 1.13064K \).

From the above, it follows that \( g(K, d) \) is monotonically increasing for \( d \geq 4Ks/(1 + s) \), and therefore

\[ g(K, d) \geq g(K, 4Ks/(1 + s)) = (0.205742K + 0.913416)K^3 \geq 0 \]

as wanted.

**Case 3:** \( d \in D_3 = [4Ks/(1 - s), 4K] \). The energy ratio \( E(K, d, s)/d \) is

\[
1/d \cdot \left( 1/3 + (K - 1)(5 - 4s(s + 1)) + 3K - 4Ks(s + 1) + \frac{2d}{1-s}(s^3 + s_{b3}^2(s + 1) + 1) \right)
\]

\[
= \frac{1/3 + (K - 1)(5 - 4s(s + 1)) + 3K - 4Ks(s + 1)}{1-s} + \frac{2s^3 + 2}{1-s} + \frac{2s + 2}{1-s} \left( \frac{d(1+s)}{d(7-9s) + 8K(s-1)} \right)^2
\]

In this case, we claim that this expression is decreasing over \( D_3 \) and achieves its maximum at \( d = 4Ks/(1 - s) \). When \( s = 0.39403 \), the above gives \( 8.425786060 - 1.0776069241/K \) (which is the same as in case 2, as one should expect). Given that \( 1/(3d) \leq 1/(7.80296K) \), we get that the energy ratio is at most

\[ 8.425786060 - 1.0776069241/K + 1/(7.80296K) \leq 8.42588 \]
Let us prove that $E(K, d, s)/d$ is indeed decreasing. Note that $E(K, d, s)/d$ equals
\[
\frac{2(s + 1)^3}{1 - s} g(K, d, s) + h(K, d, s) + \frac{2(s^3 + 1)}{1 - s}
\]
where
\[
g(K, d, s) := \frac{d^2}{(d(7 - 9s) + 8K(s - 1))^2},
\]
\[
h(K, d, s) := \frac{8K(-s^2 - s + 1) + 4s(s + 1) - 5}{d}.
\]

In what follows we prove that both $g(K, d, s), h(K, d, s)$ are strictly decreasing when $d \geq 4Ks/(1 - s)$, implying the claim of the lemma.

First we show that $h(K, d, s)$ is decreasing. For that note that, using the fixed value of $s = 0.39403$, we have $h(K, d, s) = (3.60558K - 2.80279)/d$, and the latter expression (in $d$) is clearly strictly decreasing for all constants $K > 1$.

Now we show that $g(K, d, s)$ is strictly decreasing for all $d \geq 4Ks/(1 - s)$. For that observe that for the specific constant $s$, and since $d \geq 4Ks/(1 - s) \approx 2.60107K$, we have that
\[
|d(7 - 9s) + 8K(s - 1)| = d(7 - 9s) + 8K(s - 1) > 0.
\]
Hence, to show that $g(K, d, s)$ is strictly decreasing, it is enough to prove that
\[
g(K, d, s) := \frac{d}{d(7 - 9s) + 8K(s - 1)}
\]
is strictly decreasing in $d \geq 4Ks/(1 - s)$. First observe that the rational function is well defined for these values of $d$, since the denominator becomes 0 only when $d = \frac{8K(s-1)}{9s-7} < 4Ks/(1 - s)$ (the last inequality is easy to verify). To that end, we compute
\[
\frac{\partial}{\partial d}g(K, d, s) = \frac{8K(s - 1)}{(d(7 - 9s) + 8K(s - 1))^2}
\]
which is of course negative for the given value of $s < 1$. \qed
Chapter 5

Priority evacuation from a disk: The case of \( n = 1, 2, 3 \)

This chapter and the next concern the problem of priority evacuation introduced in Section 1.1.2. In this chapter we present the paper “Priority evacuation from a disk: The case of \( n = 1, 2, 3 \)” [66] appearing in Theoretical Computer Science, and which is the journal version of a paper first appearing in the proceedings of FUN2018 (where it had the alternate title “God save the Queen”). The analysis of the problem was sufficiently different for small numbers of agents \( (n \leq 3) \) versus large numbers \( (n \geq 4) \) that it warranted separate investigations spanning two papers. This chapter specifically treats the case of small \( n \) and the next chapter treats the case of large \( n \).
Priority Evacuation from a Disk: the case of 
\[ n = 1, 2, 3 \]

Jurek Czyzowicz\textsuperscript{1}, Konstantinos Georgiou\textsuperscript{2}, Ryan Killick\textsuperscript{3}, Evangelos Kranakis\textsuperscript{3}, Danny Krizanc\textsuperscript{4}, Lata Narayanan\textsuperscript{5}, Jaroslav Opatrny\textsuperscript{5}, Sunil Shende\textsuperscript{6}

\textsuperscript{1}Université du Québec en Outaouais, Gatineau, Québec, Canada
\textsuperscript{2}Department of Mathematics, Ryerson University, Toronto, Ontario, Canada
\textsuperscript{3}School of Computer Science, Carleton University, Ottawa, Ontario, Canada
\textsuperscript{4}Department of Mathematics & Comp. Sci., Wesleyan University, Middletown, CT, USA
\textsuperscript{5}Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada
\textsuperscript{6}Department of Computer Science, Rutgers University, Camden, NJ, USA

Abstract

An exit (or target) is at an unknown location on the perimeter of a unit disk. A group of \( n + 1 \) robots (in our case, \( n = 1, 2, 3 \)), initially located at the center of the disk, are tasked with finding the exit. The robots have unique identities, share the same coordinate system, move at maximum speed 1 and are able to communicate wirelessly the position of the exit once found. Among them there is a distinguished robot called the \textit{queen} and the remainder of the robots are referred to as \textit{servants}. It is known that with two robots searching, the room can be evacuated (i.e., with both robots reaching the exit) in \( 1 + \frac{2\pi}{3} + \sqrt{3} \approx 4.8264 \) time units and this is optimal \cite{57}. Somewhat surprisingly, in this paper we show that if the goal is to have the queen reach the exit, not caring if her servants make it, there is a slightly better strategy for the case of one servant. We prove that this “priority” version of evacuation can be solved in time at most \( 4.81854 \). Furthermore, we show that any strategy for saving the queen with one servant requires time at least \( 3 + \pi/6 + \sqrt{3}/2 \approx 4.3896 \) in the worst case. If more servants are available, we show that the time bounds can be improved to 3.8327 for two servants, and 3.3738 for three servants which are better than the known lower bound for the corresponding problems of evacuating three or four robots. Finally, we show lower bounds for these cases of 3.6307 (two servants) and 3.2017 (three servants). The case of more than three servants uses substantially different techniques and is discussed in a separate paper \cite{64}.

\textbf{Key words:} Mobile Robots, Priority, Evacuation, Exit, Group Search, Disk, Wireless Communication, Queen, Servants.
5.1 Introduction

In traditional search, a group of searchers (modelled as mobile autonomous agents or robots) may collaboratively search for an exit (or target) placed within a given search domain \[2, 8, 155\]. Although the searchers may have differing capabilities (communication, perception, mobility, memory) search algorithms, previously employed, generally make no distinction between them as they usually play identical roles throughout the execution of the search algorithm and with respect to the termination time (with the exception of faulty robots, which may not contribute to searching or may even try to slow its progress). In this work we are motivated by real-life safeguarding-type situations where a number of agents have the exclusive role of facilitating the execution of the task by a distinguished agent. We introduce and study *Priority Evacuation*, a new form of search in which the search time of the algorithm is measured by the time it takes a special searcher, called the queen, to reach the exit. The remaining searchers in the group, called servants, are participating in the search but are not required to exit.

5.1.1 Problem Definition of *Priority Evacuation with n servants* (PE\(_n\))

An exit (or target) is hidden at an unknown location on the boundary of a unit disk. The exit can be located by any of the \(n + 1\) robots (searchers) when they walk over it. Robots have unique identities, share the same coordinate system, start from the center of the circle, and have maximum speed 1. Among them there is a distinguished robot, called the *queen*, and the remaining \(n\) robots are referred to as *servants*. All servants are known to the queen by their identities. Robots may run asymmetric algorithms, and can communicate their findings wirelessly and instantaneously (each message contains the robots identity and location). Feasible solutions to this problem are *evacuation algorithms*, i.e., a set of robot trajectories that guarantee the finding of the hidden exit and the queen reaching it. The cost of an evacuation algorithm is the *evacuation time* of the queen, i.e., the worst case total time until the queen reaches the exit. None of the \(n\) servants needs to evacuate.

5.1.2 Related work

Much of the work related to ours started with the problem of linear search which refers to search on an infinite line. There have been several interesting studies attempting to optimize the search time which were initiated by the influential works of Bellman \[25\] and Beck \[22\]. A long list of results followed for numerous variants of the problem, citing all of which is outside the scope of this work. For a comprehensive study of seminal search-type problems see \[8, 9\].

The problem of searching in the plane by one or more searchers, has been considered by \[13, 15\]. The unit disk model considered in our present paper is a form of two-dimensional
search that was initiated in the work of [57]. In that paper the authors obtained evacuation algorithms in the wireless and face-to-face communication models both for a small number of robots as well optimal asymptotic results for a large number of robots. Additional evacuation algorithms in the face-to-face communication model were subsequently analyzed for two robots in [73] and later in [36], while, recently, [43] considered worst-case average-case tradeoffs for the same problem. Other variations of the problem include the case of more than one exit, see [55] and [144], triangular and square domains in [79], robots with different moving speeds [137], and evacuation in the presence of crash or byzantine faulty robots [59].

Notably, all relevant previous work in search-type problems considered the objective of minimizing the time it takes either by the first or the last agent to reach the hidden exit or target. In contrast, this paper considers an evacuation (search-type) problem where the completion time is defined with respect to a distinguished mobile agent, the queen, while the remaining $n$ servants are not required to reach the exit. Notably, the algorithms we propose significantly improve upon evacuation costs induced by naive trajectories, and in fact the trajectories we propose are surprisingly complex. Our main contribution concerns priority evacuation for each of the cases of $n = 1, 2, 3$ servants, all of which require special treatment. Moreover, all our algorithms are characterized by the fact that the queen contributes effectively to the search for the exit. In sharp contrast, the independent and concurrent work of [64] studies the same problem for $n \geq 4$ servants where in the best known algorithms the queen does not contribute to the search. More importantly, the proposed algorithms of [64] admit a unified description and analysis that does not intersect with the current work.

5.1.3 Our Results & Paper Organization

Section 5.2 introduces necessary notation and terminology and discusses preliminaries. Section 5.3 is devoted to upper bounds for $PE_n$ for $n = 1, 2, 3$ servants (see Subsections 5.3.1, 5.3.2, and 5.3.3, respectively). All our upper bounds are achieved by fixing optimal parameters for families of parameterized algorithms. In Section 5.4 we derive lower bounds for $PE_n$, $n = 1, 2, 3$. An interesting corollary of our positive results is that priority evacuation with $n = 1, 2, 3$ servants (i.e., with $n + 1$ searchers) can be performed strictly faster than ordinary evacuation with $n + 1$ robots where all robots have to evacuate. Indeed, an argument found in [57] can be adjusted to show that the evacuation problem with $n + 1$ robots cannot be solved faster than $1 + \frac{4\pi}{3(n+1)} + \sqrt{3}$. We show that when one needs to evacuate only one designated robot, the task can provably (due to our upper bounds) be executed faster. All our results, together with the comparison to the lower bounds of [57], are summarized in Table 5.1. We conclude the paper in Section 5.5 with a discussion of open problems.
Table 5.1: Upper bounds (UB) and lower bounds (LB) for priority evacuation.

<table>
<thead>
<tr>
<th># of Servants</th>
<th>UB for PE&lt;sub&gt;n&lt;/sub&gt;</th>
<th>LB for PE&lt;sub&gt;n&lt;/sub&gt;</th>
<th>LB for Ordinary Evac.</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 1</td>
<td>4.8185 (Thm. 5.3.1)</td>
<td>4.3896 (Thm. 5.4.1)</td>
<td>4.826445 (see [57])</td>
</tr>
<tr>
<td>n = 2</td>
<td>3.8327 (Thm. 5.3.3)</td>
<td>3.6307 (Thm. 5.4.6)</td>
<td>4.128314 (see [57])</td>
</tr>
<tr>
<td>n = 3</td>
<td>3.3738 (Thm. 5.3.7)</td>
<td>3.2017 (Thm. 5.4.10)</td>
<td>3.779248 (see [57])</td>
</tr>
</tbody>
</table>

5.2 Notation and Preliminaries

We use \( n \) to denote the number of servants, and we set \([n] = \{1, \ldots, n\}\). Queen and servant \( i \) will be denoted by \( Q \) and \( S_i \), respectively, where \( i \in [n] \). We assume that all robots start from the origin \( O = (0, 0) \) of a unit circle in \( \mathbb{R}^2 \). As usual, points in \( A \in \mathbb{R}^2 \) will be treated, when it is convenient, as vectors from \( O \) to \( A \), and \( \|A\| \) will denote the euclidean norm of that vector.

5.2.1 Problem Reformulation & Solutions’ Description

Robots’ trajectories will be defined by parametric functions \( \mathcal{F}(t) = (f(t), g(t)) \), where \( f, g : \mathbb{R} \mapsto \mathbb{R} \) are continuous and piecewise differentiable. In particular, search algorithms for all robots will be given by trajectories

\[
\mathcal{S}_n := \left\{ Q(t), \{S_i(t)\}_{i \in [n]} \right\},
\]

where \( Q(t), S_i(t) \) will denote the position of \( Q \) and \( S_i \), respectively, at time \( t \geq 0 \).

**Definition 5.2.1** (Feasible Trajectories). We say that trajectories \( \mathcal{S}_n \) are *feasible* for PE\(_n\) if:

(a) \( Q(0) = S_i(0) = O \), for all \( i \in [n] \),

(b) \( Q(t), \{S_i(t)\}_{i \in [n]} \) induce speed-1 trajectories for \( Q \), \( \{S_i \}_{i \in [n]} \) respectively, and

(c) there is some time \( t_0 \geq 1 \), such that each point of the unit circle is visited (searched) by at least one robot in the time window \([0, t_0]\). We refer to the smallest such \( t_0 \) as the *search time* of the circle.

Note that feasible trajectories do indeed correspond to robots’ movements for PE\(_n\) in which, eventually the entire circle is searched, and hence the search time is bounded. We will describe all our search/evacuation algorithms as feasible trajectories, and we will assume that once the target is reported, \( Q \) will go directly to the location of the exit.

For feasible trajectories \( \mathcal{S}_n \) with search time \( t_0 \), and for any trajectory \( \mathcal{F}(t) \) (either of the queen or of a servant), we denote by \( \mathbb{I}(\mathcal{F}) \) the subinterval of \([0, t_0]\) that contains all
$x \in [0, t_0]$ such that $\|F(x)\| = 1$ (i.e., the robot is on the circle) and no other robot has been to $F(x)$ before. Since robots start from the origin, it is immediate that $\mathbb{I}(\mathcal{F}) \subseteq [1, t_0]$. With this notation in mind, note that the exit can be discovered by some robot $\mathcal{F}$, say at time $x$, only if $x \in \mathbb{I}(\mathcal{F})$. In this case, the finding is instantaneously reported, so $Q$ goes directly to the exit, moving along the corresponding line segment between her current position $Q(x)$ and the reported position of the exit $F(x)$. Hence, the total time that $Q$ needs to evacuate equals

$$x + \|Q(x) - F(x)\|.$$ 

Therefore, the evacuation time of feasible trajectories $\mathbb{S}_n$ to $\text{PE}_n$ is given by expression

$$\max_{\mathcal{F} \in \mathbb{S}_n} \sup_{x \in \mathbb{I}(\mathcal{F})} \left\{ x + \|Q(x) - F(x)\| \right\}.$$ 

Notice that for “non-degenerate” search algorithms for which the last point on the circle is not searched by $Q$ alone, the previous maximum can be simply computed over the servants, i.e., the evacuation cost will be

$$\max_{i \in [n]} \sup_{x \in \mathbb{I}(\mathcal{S}_i)} \left\{ x + \|Q(x) - \mathcal{S}_i(x)\| \right\}. \quad (5.1)$$

In other words, we can restate $\text{PE}_n$ as the problem of determining feasible trajectories $\mathbb{S}_n$ so as to minimize $(5.1)$.

### 5.2.2 Useful Trajectories’ Components

Feasible trajectories induce, by definition, robots that are moving at (maximum) speed 1. The speed restriction will be ensured by the next condition.

**Lemma 5.2.2.** An object following trajectory $\mathcal{F}(t) = (f(t), g(t))$ has unit speed if and only if

$$(f'(t))^2 + (g'(t))^2 = 1, \ \forall t \geq 0.$$ 

**Proof.** For any $t \geq 0$, the velocity of $\mathcal{F}$ is given by $\mathcal{F}'(t) = (df(t)/dt, dg(t)/dt)$, and its speed is calculated as $||\mathcal{F}'(t)||$. \qed

Robots’ trajectories will be composed by piecewise smooth parametric functions. In order to describe them, we introduce some further notation. For any $\theta \in \mathbb{R}$, we introduce abbreviation $C_\theta$ for point $\{\cos(\theta), \sin(\theta)\}$. Next we introduce parametric equations for moving along the perimeter of a unit circle (Lemma 5.2.3), and along a line segment (Lemma 5.2.4).
**Lemma 5.2.3.** Let $b \in [0, 2\pi)$ and $\sigma \in \{-1, 1\}$. The trajectory of an object moving at speed 1 on the perimeter of a unit circle with initial location $C_b$ is given by the parametric equation

$$C(b, \sigma t) := (\cos (\sigma t + b), \sin (\sigma t + b)).$$

If $\sigma = 1$ the movement is counter-clockwise (ccw), and clockwise (cw) otherwise.

**Proof.** Clearly, $C(b, 0) = C_b$. Also, it is easy to see that $\|C(b, t)\| = 1$, i.e., the object is moving on the perimeter of the unit circle. Lastly,

$$\left(\frac{d}{dt} \cos (\sigma t + b)\right)^2 + \left(\frac{d}{dt} \sin (\sigma t + b)\right)^2 = \sigma^2 (-\sin (\sigma t + b))^2 + \sigma^2 (\cos (\sigma t + b))^2 = 1,$$

so the claim follows by Lemma 5.2.2.

**Lemma 5.2.4.** Consider distinct points $A = (a_1, a_2), B = (b_1, b_2)$ in $\mathbb{R}^2$. The trajectory of a speed 1 object moving along the line passing through $A, B$ and with initial position $A$ is given by the parametric equation

$$L(A, B, t) := \left(\frac{b_1 - a_1}{\|A - B\|} t + a_1, \frac{b_2 - a_2}{\|A - B\|} t + a_2\right).$$

**Proof.** It is immediate that the parametric equation corresponds to a line. Also, it is easy to see that $L(A, B, 0) = A$ and $L(A, B, \|A - B\|) = B$, i.e., the object starts from $A$, and eventually visits $B$. As for the object’s speed, we calculate

$$\left(\frac{d}{dt} \left(\frac{b_1 - a_1}{\|A - B\|} t + a_1\right)\right)^2 + \left(\frac{d}{dt} \left(\frac{b_2 - a_2}{\|A - B\|} t + a_2\right)\right)^2 = \left(\frac{b_1 - a_1}{\|A - B\|}\right)^2 + \left(\frac{b_2 - a_2}{\|A - B\|}\right)^2 = 1$$

so, by Lemma 5.2.2, the speed is indeed 1.

Robots trajectories will be described in phases. In each phase, robot, say $F$, will be moving between two explicit points, and the corresponding trajectory $F(t)$ will be implied by the previous description, using most of the times Lemma 5.2.3 and Lemma 5.2.4. We will summarize the details in tables of the following format.

<table>
<thead>
<tr>
<th>Robot</th>
<th>#</th>
<th>Description</th>
<th>Trajectory</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>0</td>
<td>$F(t)$</td>
<td>$t_0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$F(t)$</td>
<td>$t_1$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Phase 0 will usually correspond to the deployment of $F$ from the origin to some point of the circle. Also, for each phase, we will summarize its duration. With that in mind, trajectory $F(t)$ during phase $i$, with duration $t_i$, will be valid for all $t \geq 0$ with $t_0 + t_1 + \ldots + t_{i-1} \leq t \leq t_0 + t_1 + \ldots + t_i$.

Lastly, the following abbreviation will be useful for the exposition of the trajectories. For any $\rho \in [0, 1]$ and $\theta \in [0, 2\pi)$, we introduce notation

$$K(\theta, \rho) := (1 - \rho)C_{\pi - \theta} + \rho C_{-\theta}.$$  

In other words, $K(\theta, \rho)$ is a convex combination of antipodal points $C_{\pi - \theta}, C_{-\theta}$ of the unit circle, i.e., it lies on the diameter of the unit circle passing through these two points. Moreover, it is easy to see that $\|C_{\pi - \theta} - K(\theta, \rho)\| = 2\rho$, and hence

$$\|K(\theta, \rho) - C_{-\theta}\| = 2 - 2\rho.$$  

As it will be handy later, we also introduce abbreviation

$$AK(\theta, \rho) := \|C_{\pi} - K(\theta, \rho)\|.$$  

The choice of the abbreviation is clear, if the reader denotes $C_{\pi} = (-1, 0)$ by $A$.

### 5.2.3 Critical Angles

The following definition introduces a key concept. In what follows, abstract trajectories will be assumed to be continuous and differentiable, which in particular implies that corresponding velocities are continuous.

**Definition 5.2.5 (Critical Angle).** Let $S(t) \in \mathbb{R}^2$ denote the trajectory of a speed-1 object, where $t \geq 0$. For some point $Q \in \mathbb{R}^2$, we define the $(S, Q)$-critical angle at time $t = \tau$ to be the angle between the velocity vector $S'(\tau)$ and vector $S(\tau)Q$, i.e. the vector from $S(\tau)$ to $Q$. 

---

Figure 5.1: An illustration of trajectories $S(t), Q(t)$, and their critical angles at some fixed time $\tau$, with $S(\tau) = S, Q(\tau) = Q, S'(\tau) = u, Q'(\tau) = v$. 

**Definition 5.2.5 (Critical Angle).** Let $S(t) \in \mathbb{R}^2$ denote the trajectory of a speed-1 object, where $t \geq 0$. For some point $Q \in \mathbb{R}^2$, we define the $(S, Q)$-critical angle at time $t = \tau$ to be the angle between the velocity vector $S'(\tau)$ and vector $S(\tau)Q$, i.e. the vector from $S(\tau)$ to $Q$. 

---

141
We make the following critical observation, see also Figure 5.1.

**Theorem 5.2.6.** Consider trajectories \( S(t), Q(t) \) of two speed-1 objects \( S, Q \), where \( t \geq 0 \). Let also \( \phi, \theta \) denote the \( (S, Q(t)) \)-critical angle and the \( (Q, S(t)) \)-critical angle at time \( t \), respectively. Then \( t + \|Q(t) - S(t)\| \) is strictly increasing if \( \cos(\phi) + \cos(\theta) < 1 \), strictly decreasing if \( \cos(\phi) + \cos(\theta) > 1 \), and constant otherwise.

Theorem 5.2.6 is an immediate corollary of the following lemma.

**Lemma 5.2.7.** Consider trajectories \( S(t), Q(t) \) and their critical angles \( \pi, \theta \), as in the statement of Theorem 5.2.6. Then
\[
\frac{d}{dt} \|Q(t) - S(t)\| = \cos(\phi) + \cos(\theta).
\]

**Proof.** For any fixed \( t \), let \( d \) denote \( D(t) \), and \( S, Q \) denote points \( S(t), Q(t) \), respectively. Denote also by \( u, v \) the velocities of \( S, Q \) at time \( t \), respectively, i.e. \( u = S'(t), v = Q'(t) \). See also Figure 5.1.

With that notation, observe that \( \|\overrightarrow{SQ}\| = d \). Since \( \|u\| = \|v\| = 1 \), we see that
\[
\text{proj}_{SQ}u = \frac{\cos(\phi)}{d} \overrightarrow{SQ}
\]
and
\[
\text{proj}_{SQ}v = \frac{\cos(\theta)}{d} \overrightarrow{QS}.
\]

Now consider two imaginary objects \( \overline{S}, \overline{Q} \), with corresponding velocities \( \overline{S}'(t) = \text{proj}_{SQ}u \) and \( \overline{Q}'(t) = \text{proj}_{SQ}v \). It is immediate that \( \|Q(t) - S(t)\| = \|\overline{Q}(t) - \overline{S}(t)\| \).

In particular, \( \text{proj}_{SQ}u - \text{proj}_{SQ}v \) is the projection of the relative velocities of \( S, Q \) on the line segment connecting \( S(t), Q(t) \). As such, the distance between \( S, Q \) changes at a rate determined by velocity
\[
\text{proj}_{SQ}u - \text{proj}_{SQ}v = \frac{\cos(\phi) + \cos(\theta)}{d} \overrightarrow{SQ},
\]
where \( \|\text{proj}_{SQ}u - \text{proj}_{SQ}v\| = |\cos(\phi) + \cos(\theta)| \). Moreover, \( \text{proj}_{SQ}u, \text{proj}_{SQ}v \) are antiparallel if and only if \( \cos(\phi), \cos(\theta) > 0 \), in which case the two objects come closer to each other. \( \square \)

### 5.3 Upper Bounds

#### 5.3.1 Evacuation Algorithm for PE$_1$

This subsection is devoted in proving the following.
Theorem 5.3.1. Consider the real function \( f(x) = x + \sin(x) \), and denote by \( \alpha_0 > 0 \) the solution to equation
\[
f(f(\alpha - \sin(\alpha))) = \sin(\alpha),
\]
with \( \alpha_0 \approx 1.14193 \). Then \( PE_1 \) can be solved in time \( 1 + \pi - \alpha_0 + 2\sin(\alpha_0) \approx 4.81854 \).

The value of \( \alpha_0 \) is well defined in the statement of Theorem 5.3.1. Indeed, by letting \( g(x) = f(f(x - \sin(x))) - \sin(x) \), we observe that \( g \) is continuous, while \( g(1) \approx -0.213934 \) and \( g(\pi/2) \approx 1.00729 \), hence there exists \( \alpha_0 \in (1, \pi/2) \) with \( g(\alpha_0) = 0 \).

In order to prove Theorem 5.3.1 and given parameters \( \alpha, \beta \), we introduce the family of trajectories \( \text{SEARCH}_1(\alpha, \beta) \), see also Figure 5.2.

Figure 5.2: Algorithm \( \text{SEARCH}_1(\alpha, \beta) \) depicted for the optimal parameters of the algorithm. In all subsequent figures, as well as here, the orange points on the perimeter of the disc correspond to the worst adversarial placements of the treasure, which due to our optimality conditions induce the same evacuation cost. The orange points in \( Q \)'s trajectories correspond to the \( Q \)'s positioning when the treasures are reported, in the worst cost induced cases. The green dashed line depict \( Q \)'s trajectory after \( Q \) abandons her trajectory and moves toward the reported exit following a straight line.

| Algorithm \text{SEARCH}_1(\alpha, \beta) |
|---|---|---|---|
| **Robot** | **Description** | **Trajectory** | **Duration** |
| Q | 0 Move to \( C_{\pi} \) | \( \mathcal{L}(O, C_{\pi}, t) \) | 1 |
| | 1 Search ccw till \( C_{-\alpha} \) | \( \mathcal{C}(\pi, t - 1) \) | \( \pi - \alpha \) |
| | 2 Move to \( C_{-\alpha + \beta} \) | \( \mathcal{L}(C_{-\alpha}, C_{-\alpha + \beta}, t - (1 + \pi - \alpha)) \) | \( 2\sin(\beta/2) \) |
| | 3 Search ccw till \( C_{-\alpha} \) | \( \mathcal{C}(\beta - \alpha, 1 + \pi - \alpha + 2\sin(\beta/2) - t) \) | \( \beta \) |
| S_1 | 0 Move to \( C_{\pi} \) | \( \mathcal{L}(O, C_{\pi}, t) \) | 1 |
| | 1 Search ccw till \( C_{\beta - \alpha} \) | \( \mathcal{C}(\pi, t - 1) \) | \( \pi + \alpha - \beta \) |
Partitioning the circle clockwise, we see that the arc with endpoints \( C_\alpha, C_{\alpha+\beta} \) is searched by \( S_1 \), while the remaining of the circle is searched by \( Q \). Therefore, robots’ trajectories in \( \text{ SEARCH}_1(\alpha, \beta) \) are feasible, and it is also easy to see that they are continuous as well. The search time equals \( 1 + \pi + \max\{\alpha - \beta, 2\sin(\beta/2) + \beta - \alpha\} \), as well as

\[
\begin{align*}
\mathbb{I}(Q) &= [1, 1 + \pi - \alpha] \cup [1 + \pi - \alpha + 2\sin(\beta/2), 1 + \pi - \alpha + 2\sin(\beta/2) + \beta], \\
\mathbb{I}(S_1) &= [1, 1 + \pi + \alpha - \beta].
\end{align*}
\]

An illustration of the above trajectories for certain values of \( \alpha, \beta \) can be seen in Figure 5.2.

First we make some observations pertaining to the monotonicity of the evacuation cost.

**Lemma 5.3.2.** Assuming that \( \alpha > \pi/3 \) and that \( \cos(\alpha) + \cos(\alpha - \beta/2) > 1 \), the evacuation cost of \( \text{ SEARCH}_1(\alpha, \beta) \) is monotonically increasing if the exit is found by \( S_1 \) during \( Q \)’s phase 1 and monotonically decreasing if the exit is found by \( S_1 \) during \( Q \)’s phase 2.

**Proof.** Suppose that the exit is found by \( S_1 \) during \( Q \)’s phase 1, i.e., at time \( x \) after robots start searching for the first time, where \( 0 \leq x \leq \pi - \alpha \). It is easy to see that the critical angles between \( Q, S_1 \) are both equal to \( \pi - x \). But then \( 2\cos(\pi - x) \geq 2\cos(\alpha) > 2\cos(\pi/3) = 1 \). Hence, by Theorem 5.2.6, the evacuation cost is decreasing in this case.

Now suppose that the exit is found by \( S_1 \) during \( Q \)’s phase 2, i.e., at time \( x \) after \( Q \) starts moving along the chord with endpoints \( C_\alpha, C_{\alpha+\beta} \), where \( 0 \leq x \leq 2\sin(\beta/2) \). If \( \phi_x, \theta_x \) denote the \( S_1, Q \) critical angles, then it is easy to see that \( \phi_0 = \cos(\alpha) \) and that \( \theta_0 = \alpha - \beta/2 \). Since \( \cos(\phi_0) + \cos(\theta_0) > 1 \), Theorem 5.2.6 implies that the evacuation cost is initially decreasing in this phase. For the remaining of \( Q \)’s phase 2, it is easy to see that both \( \phi_x, \theta_x \) are decreasing in \( x \), hence \( \cos(\phi_x) + \cos(\theta_x) \) is increasing in \( x \), hence, the evacuation cost will remain decreasing in this phase. \( \square \)

Now we can prove Theorem 5.3.1 by fixing certain values for parameters \( \alpha, \beta \) of \( \text{ SEARCH}_1(\alpha, \beta) \). In particular, we set \( \alpha_0 \) as in the statement of Theorem 5.3.1 and \( \beta_0 = 2f(\alpha_0 - \sin(\alpha_0)) \approx 0.925793 \). The trajectories of the robots, for the exact same values of the parameters, can be seen in Figure 5.2.

**Proof.** Let \( f, \alpha_0 \) be as in the statement of Theorem, and set \( \beta_0 = 2f(\alpha_0 - \sin(\alpha_0)) \approx 0.925793 \). We argue that the worst evacuation time of \( \text{ SEARCH}_1(\alpha_0, \beta_0) \) is \( 1 + \pi - \alpha_0 + 2\sin(\alpha_0) \). Note that for the given values of the parameters, we have that \( \alpha_0 > \pi/3 \), that \( \alpha_0 - \sin(\beta_0/2) \leq \beta_0 \), and that \( \cos(\alpha_0) + \cos(\alpha_0 - \beta_0/2) > 1 \).

First we observe that if the exit is found by \( Q \), then the worst case evacuation time \( E_0(\alpha_0, \beta_0) \) is incurred when the exit is found just before \( Q \) stops searching, that is

\[
E_0(\alpha_0, \beta_0) = 1 + \pi - \alpha_0 + 2\sin(\beta_0/2) + \beta_0.
\]
Next we examine some cases as to when the exit is found by \( S_1 \). If the exit is found by \( S_1 \) during the 1st phase of \( Q \), then the evacuation time is, due to Lemma 5.3.2, given as

\[
E_1(\alpha_0, \beta_0) = \sup_{1 \leq x \leq 1 + \pi - \alpha_0} \{x + \|Q(x) - S_1(x)\|\} = 1 + \pi - \alpha_0 + 2 \sin(\alpha_0).
\]

Recall that \( \cos(\alpha_0) + \cos(\alpha_0 - \beta_0/2) > 1 \), and so, again by Lemma 5.3.2 we may omit the case that the exit is found by \( S_1 \) while \( Q \) is in phase 2. The end of \( Q \) ’s phase 2 happens at time \( \tau := 1 + \pi - \alpha_0 + 2 \sin(\beta_0/2) \), when have that \( Q(\tau) = C_{-\alpha+\beta} \), and \( S_1(\tau) = C_{\alpha-2 \sin(\beta_0/2)} \), and both robots are intending to search ccw. Condition \( \alpha_0 - \sin(\beta_0/2) \leq \beta_0 \) says that \( S_1 \) will finish searching prior to \( Q \), and this happens when \( S_1 \) reaches point \( C_{-\alpha+\beta} \). During this phase, the distance between \( Q, S_1 \) stays invariant and equal to \( 2\alpha_0 - \beta_0 - 2 \sin(\beta_0/2) \). We conclude that the cost in this case would be

\[
E_2(\alpha_0, \beta_0) = 1 + \pi + \alpha_0 - \beta_0 + 2 \sin(\alpha_0 - \beta_0/2 - \sin(\beta_0/2)).
\]

Then, we argue that that the choice of \( \alpha_0, \beta_0 \) guarantees that \( E_0(\alpha_0, \beta_0) = E_1(\alpha_0, \beta_0) = E_2(\alpha_0, \beta_0) \), as wanted.

Indeed, \( E_0(\alpha_0, \beta_0) = E_1(\alpha_0, \beta_0) \) implies that \( \sin(\beta_0/2) + \beta_0/2 = \sin(\alpha_0) \). But then, we can rewrite \( E_2(\alpha_0, \beta_0) \) as

\[
E_2(\alpha_0, \beta_0) = 1 + \pi + \alpha_0 - \beta_0 + 2 \sin(\alpha_0 - \sin(\alpha_0)).
\]

Equating the last expression with \( E_1(\alpha_0, \beta_0) \) implies that

\[
\beta_0/2 = \alpha_0 - \sin(\alpha_0) + \sin(\alpha_0 - \sin(\alpha_0)) = f(\alpha_0 - \sin(\alpha_0)).
\]

Substituting twice \( \beta_0/2 \) in the already derived condition \( \sin(\beta_0/2) + \beta_0/2 = \sin(\alpha_0) \) implies that

\[
f(f(\alpha - \sin(\alpha_0))) = \sin(\alpha_0).
\]

Figure 5.2 depicts the worst placements of the exit, along with the trajectories of the queen (in dashed green lines) after the exit is reported.


It should be stressed that \( Q \) ’s Phases 2,3 are essential for achieving the promised bound. Indeed, had we chosen \( \alpha = \beta = 0 \), the worst case evacuation time would have been

\[
\sup_{1 \leq x \leq 1 + \pi} \{x + \|Q(x) - S_1(x)\|\} = \sup_{0 \leq x \leq \pi} \{1 + x + 2 \sin(x)\}.
\]

The maximum is attained at \( x_0 = 2\pi/3 \) (and indeed, both critical angles in this case are \( \pi/3 \) and in particular \( 2 \cos(\pi/3) = 1 \)), inducing a cost of \( 1 + 2\pi/3 + \sqrt{3} \approx 4.82645 \). The latter is the cost of the evacuation algorithm for two robots without priority of [57].
5.3.2 Evacuation Algorithm for PE\textsubscript{2}

In this subsection we prove the following theorem.

**Theorem 5.3.3.** PE\textsubscript{2} can be solved in time 3.8327.

Given parameters $\alpha$, $\rho$, we introduce the family of trajectories $\text{SEARCH}_{2}(\alpha, \rho)$, see also Figure 5.3.

![Figure 5.3: Algorithm $\text{SEARCH}_{2}(\alpha, \beta)$ depicted for the optimal parameters of the algorithm.](image)

<table>
<thead>
<tr>
<th>Algorithm $\text{SEARCH}_{2}(\alpha, \rho)$</th>
<th>Robot</th>
<th>Description</th>
<th>Trajectory</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>0</td>
<td>Move to $C_{\pi - \alpha}$</td>
<td>$L(O, C_{\pi - \alpha}, t)$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Search ccw till $C_{\pi}$</td>
<td>$C(\pi - \alpha, t - 1)$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Move to $K(\alpha/2, \rho)$</td>
<td>$L(C_{\pi}, K(\alpha/2, \rho), t - (1 + \alpha))$</td>
<td>$AK(\alpha/2, \rho)$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Move to $C_{-\alpha/2}$</td>
<td>$L(K(\alpha/2, \rho), C_{-\alpha/2})$</td>
<td>$2 - 2\rho$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>0</td>
<td>Move to $C_{\pi - \alpha}$</td>
<td>$L(O, C_{\pi - \alpha})$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Search cw till $C_{-\alpha/2}$</td>
<td>$C(\pi - \alpha, -t + 1)$</td>
<td>$\pi - \alpha/2$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0</td>
<td>Move to $C_{\pi}$</td>
<td>$L(O, C_{\pi})$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Search cw till $C_{-\alpha/2}$</td>
<td>$C(\pi, t - 1)$</td>
<td>$\pi - \alpha/2$</td>
</tr>
</tbody>
</table>

Notice that, by definition of $\text{SEARCH}_{2}(\alpha, \rho)$, robots’ trajectories are continuous and feasible, meaning that the entire circle is eventually searched. Indeed, partitioning the circle clockwise, we see that: the arc with endpoints $C_{\pi}, C_{\pi - \alpha}$ is searched by $Q$, the arc with endpoints $C_{\pi - \alpha}, C_{-\alpha/2}$ is searched by $S_1$, and the arc with endpoints $C_{-\alpha/2}, C_{\pi}$ is searched by $S_2$. 

146
It is immediate from the description of the trajectories that the search time is $1 + \pi - \alpha/2$. Moreover

$$\mathbb{I}(Q) = [1, 1 + \alpha], \mathbb{I}(S_1) = \mathbb{I}(S_2) = [1, 1 + \pi - \alpha/2].$$

An illustration of the above trajectories for certain values of $\alpha, \rho$ can be seen in Figure 5.3. Now we make some observations, in order to calculate the worst case evacuation time.

**Lemma 5.3.4.** Suppose that $\pi - \alpha/2 \geq \alpha + AK(\alpha/2, \rho) + 2 - 2\rho$. Then $\|Q(x) - S_1(t)\|$ is continuous and differentiable in the time intervals $I_1, I_2, I_3$ of $Q$’s phases 1,2,3, respectively. Moreover, the worst case evacuation time of $\text{SEARCH}_2(\alpha, \rho)$ can be computed as

$$\max \begin{cases} 
1 + \alpha + 2\sin(\alpha), \\
\sup_{t \in I_2} \{t + \|Q(t) - S_1(t)\|\}, \\
\sup_{t \in I_3} \{t + \|Q(t) - S_1(t)\|\}, \\
1 + \pi - \alpha/2
\end{cases}$$

where

$I_2 = [1 + \alpha, 1 + \alpha + AK(\alpha/2, \rho)], I_3 = [1 + \alpha + AK(\alpha/2, \rho), 3 - 2\rho + \alpha + AK(\alpha/2, \rho)].$

**Proof.** Note that the line passing through $O$ and $C_{-\alpha/2}$, call it $\epsilon$, has the property that each point of it, including $K(\alpha/2, \rho)$ is equidistant from $S_1, S_2$. Moreover, in the time window $[1 + \alpha, 1 + \alpha + AK(\alpha/2, \rho)]$ that only $S_1, S_2$ are searching, $Q$ stays below line $\epsilon$. At time $1 + \alpha + AK(\alpha/2, \rho)$, $Q$ is, by construction, equidistant from $S_1, S_2$, a property that is preserved for the remaining of the execution of the algorithm. As a result, the evacuation time of $\text{SEARCH}_2(\alpha, \rho)$ is given by $\sup_{1 \leq t \leq 1 + \pi - \alpha/2} \{t + \|Q(t) - S_1(t)\|\}$.

Now note that condition $\pi - \alpha/2 \geq \alpha + AK(\alpha/2, \rho) + 2 - 2\rho$ guarantees that $Q$ reaches point $C_{-\alpha/2}$ no later than $S_1$. Moreover, in each time interval $I_1, I_2, I_3$, $Q$’s trajectory is differentiable (and so is $S_1$’s trajectory).

Now Theorem 5.3.3 can be proven by fixing parameters $\alpha, \rho$ for $\text{SEARCH}_2(\alpha, \rho)$, in particular, $\alpha = 0.6361, \rho = 0.7944$.

**Proof.** We choose $\alpha = 0.6361, \rho = 0.7944$. The trajectories of Figure 5.3 correspond exactly to those values. The time that $Q$ needs to reach $C_{-\alpha/2}$ equals $1 + \alpha + AK(\alpha/2, \rho) + 2 - 2\rho = 3.6174$, while the time that $S_1, S_2$ reach the same point is $1 + \pi - \alpha/2 = 3.82354$. Hence, Lemma 5.3.4 applies.

The worst case evacuation time during phase 1 is $1 + a + 2\sin(\alpha) = 2.82423$. The worst case evacuation time after $Q$ reaches $C_{-\alpha/2}$, equals $1 + \pi - \alpha/2 = 3.82354$. Hence, it remains to compute the maxima of $t + \|Q(t) - S_1(t)\|$ in the two intervals $I_2, I_3$, which can be done numerically using the trajectories of $\text{SEARCH}_2(\alpha, \rho)$, since the expression is differentiable in each of the intervals.
To that end, when $t \in I_2 = [1.6361, 3.2062]$ we have that

$$Q(t) = (0.9931t - 2.62481, 0.191866 - 0.11727t)$$

$$S_1(t) = (\cos (3.50549 - t), \sin (3.50549 - t)),$$

so that $t + \|Q(t) - S_1(t)\|$ becomes

$$t + [\sin (3.50549 - t) + 0.11727t - 0.19187]^2$$

$$+ [\cos (3.50549 - t) - 0.9931t + 2.62481]^2]^{1/2}$$

When $t \in I_3 = [3.2062, 3.6174]$ we have that

$$Q(t) = (0.949847t - 2.48613, 0.818501 - 0.312715t)$$

while $S_1$’s trajectory equation remains unchanged, so that $t + \|Q(t) - S_1(t)\|$ becomes

$$t + [\sin (3.50549 - t) + 0.312715t + 0.81850]^2$$

$$+ [\cos (3.50549 - t) - 0.94985t - 2.48613]^2]^{1/2}.$$

In particular, it follows that

$$\sup_{t \in I_2} \{t + \|Q(t) - S_1(t)\|\} \approx \sup_{t \in I_3} \{t + \|Q(t) - S_1(t)\|\} \approx 3.8327$$

with corresponding maximizers (with approximate values) $\tau_2 = 3.10066$ and $\tau_3 = 3.32114$, respectively. Figure 5.3 also depicts the locations of the optimizers, i.e., the worst case locations on the circle for the exit to be found, along with the corresponding evacuation trajectory in dashed green colour. □

5.3.3 Evacuation Algorithm for PE_3

A Simple Algorithm

In this section we prove the following preliminary theorem (to be improved in Section 5.3.3).

**Theorem 5.3.5.** PE_3 can be solved in time 3.37882.

Given parameters $\alpha, \beta, \rho$, we introduce the family of trajectories SEARCH_3($\alpha, \beta, \rho$), corresponding to robots $Q, S_1, S_2, S_3$, see also Figure 5.4.
As before, it is immediate that, in SEARCH\(_3(\alpha, \beta, \rho)\), robots’ trajectories are continuous and feasible, meaning that the entire circle is eventually searched. In particular, the arc with endpoints \(C_{\pi}, C_{\pi - \alpha}\) is searched by \(Q\), the arc with endpoints \(C_{\pi - \alpha - \beta}, C_{-\alpha + \beta/2}\) is searched by \(S_1\), the arc with endpoints \(C_{-\pi}, C_{-\alpha + \beta/2}\) is searched by \(S_2\), and the arc with endpoints \(C_{\pi - \alpha}, C_{\pi - \alpha - \beta}\) is searched by \(S_3\). Also, the search time is \(1 + \pi - \frac{\alpha + \beta}{2}\), and

\[
\mathbb{I}(Q) = [1, 1 + \alpha], \quad \mathbb{I}(S_1) = \mathbb{I}(S_2) = [1, 1 + \pi - \frac{\alpha + \beta}{2}], \quad \mathbb{I}(S_3) = [1, 1 + \beta].
\]

An illustration of the above trajectories for certain values of \(\alpha, \beta, \rho\) can be seen in Figure 5.4.

Before we prove Theorem 5.3.5, we need to make some observation, in order to calculate the worst case evacuation time.
Lemma 5.3.6. Suppose that $\alpha \leq \beta$, $\alpha + AK(\frac{\alpha + \beta}{2}, \rho) \geq \beta$, and $\pi - \frac{\alpha + \beta}{2} \geq \alpha + AK(\frac{\alpha + \beta}{2}, \rho) + 2 - 2\rho$. Then the following functions are continuous and differentiable in each associated time intervals: $\|Q(x) - S_3(t)\|$ in $I_1 = \{t \geq 0 : \alpha \leq t - 1 \leq \beta\}$, $\|Q(x) - S_1(t)\|$ in $I_2 = \{t \geq 0 : |t-1-\alpha| \leq AK(\frac{\alpha + \beta}{2}, \rho)\}$ and in $I_3 = \{t \geq 0 : |t-1-\alpha-\beta| \leq 2 - 2\rho\}$. Moreover, the worst case evacuation time of $\text{SEARCH}_3(\alpha, \beta, \rho)$ can be computed as

$$\max \left\{ \sup_{t \in I_1} \{t + \|Q(t) - S_3(t)\|\}, \sup_{t \in I_2} \{t + \|Q(t) - S_1(t)\|\}, \sup_{t \in I_3} \{t + \|Q(t) - S_1(t)\|\} \right\}.$$ 

Proof. Conditions $\alpha \leq \beta$ and $\alpha + AK(\frac{\alpha + \beta}{2}, \rho) \geq \beta$ mean that $Q$ stops searching no later than $S_3$, and that when $S_3$ stops searching $Q$ is still in her phase 2, respectively.

The line passing through $O$ and $C_{-(\alpha + \beta)/2}$, call it $\epsilon$, has the property that each point of it, including $K(\frac{\alpha + \beta}{2}, \rho)$ is equidistant from $S_1, S_2$. Moreover, while $S_1, S_2$ are searching, $Q$ never goes above line $\epsilon$. At time $1 + \alpha + AK(\frac{\alpha + \beta}{2}, \rho)$, $Q$ is, by construction, equidistant from $S_1, S_2$, a property that is preserved for the remaining of the execution of the algorithm. As a result, $S_2$ can be ignored in the performance analysis, and when it comes to the case that $S_1$ finds the exit, the evacuation cost is given by the supremum of $t + \|Q(t) - S_1(t)\|$ in the time interval $I_2$ or in the interval $I_3$. Note that in both intervals, the evacuation cost is continuous and differentiable, by construction.

If the exit is reported by $S_3$ then the evacuation cost is $t + \|Q(t) - S_3(t)\|$ for $t \in [1, 1 + \beta]$. However, it is easy to see that the cost is strictly increasing for all $t \in [1, 1 + \alpha]$ (in fact it is linear). Since the evacuation cost is also continuous, we may restrict the analysis in interval $I_1$.

Lastly, observe that $\pi - \frac{\alpha + \beta}{2} \geq \alpha + AK(\frac{\alpha + \beta}{2}, \rho) + 2 - 2\rho$ implies that $S_1, S_2$ reach point $C_{-(\alpha + \beta)/2}$ no earlier than $Q$. Hence $Q$ waits at $C_{-(\alpha + \beta)/2}$ until the search of the circle is over, which can be easily seen to induce the worse evacuation time after $Q$ reaches $C_{-(\alpha + \beta)/2}$. \hfill $\square$

Next, we prove Theorem 5.3.5 by fixing parameters $\alpha, \beta, \rho$ for $\text{SEARCH}_3(\alpha, \beta, \rho)$.

Proof. We choose $\alpha = 0.26738, \beta = 1.2949, \rho = 0.70685$. The trajectories of Figure 5.4 correspond exactly to those values. The time that $Q$ needs to reach $C_{-(\alpha + \beta)/2}$ equals $1 + \alpha + AK(\frac{\alpha + \beta}{2}, \rho) + 2 - 2\rho = 3.17984$, while the time that $S_1, S_2$ reach the same point is $1 + \pi - \frac{\alpha + \beta}{2} = 3.36045$. Hence, Lemma 5.3.6 applies.

From the above, it is immediate that the worst evacuation time after $Q$ reaches $C_{-(\alpha + \beta)/2}$ equals $1 + \pi - \frac{\alpha + \beta}{2} = 3.36045$. Hence, it remains to compute the maxima of $t + \|Q(t) - S_3(t)\|$ in interval $I_1$, and of $t + \|Q(t) - S_1(t)\|$ in intervals $I_2, I_3$. 

150
To that end, when \( t \in I_1 = [1.26738, 2.2949] \) we have that
\[
Q(t) = (-2.23643 + 0.97558t, 0.278372 - 0.219643t)
\]
\[
S_3(t) = (\cos(t + 0.579313), \sin(t + 0.579313))
\]
so that \( t + \|Q(t) - S_3(t)\| \) becomes
\[
t + [(-0.21964t - \sin(t + 0.57931) + 0.27837)^2
+ (0.97558t - \cos(t + 0.57931) - 2.23643)^2]^{1/2}
\]
in which case
\[
\sup_{t \in I_1} \{ t + \|Q(t) - S_3(t)\| \} = 1 + \beta + \|Q(1 + \beta) - S_3(1 + \beta)\| \approx 3.37882
\]
When \( t \in I_2 = [1.26738, 2.59354] \), \( Q \)'s trajectory is the same as in \( I_1 \) and
\[
S_1(t) = (\cos(2.57931 - t), \sin(2.57931 - t))
\]
so that \( t + \|Q(t) - S_1(t)\| \) becomes
\[
t + [(-\sin(2.57931 - t) - 0.21964t + 0.27837)^2
+ (-\cos(2.57931 - t) + 0.97558t - 2.23643)^2]^{1/2}
\]
When \( t \in I_3 = [2.59354, 3.17984] \), \( S_1 \)'s trajectory is the same as in \( I_2 \) and
\[
Q(t) = (-1.54793 + 0.710111t, 1.5348 - 0.704089t)
\]
so that \( t + \|Q(t) - S_1(t)\| \) becomes
\[
t + [(\sin(2.57931 - t) + 0.704089t - 1.5348)^2
+ (\cos(2.57931 - t) - 0.710111t + 1.54793)^2]^{1/2}
\]
Numerically
\[
\sup_{t \in I_2} \{ t + \|Q(t) - S_1(t)\| \} = \tau_2 + \|Q(\tau_2) - S_1(\tau_2)\| \approx 3.37882
\]
\[
\sup_{t \in I_3} \{ t + \|Q(t) - S_1(t)\| \} = \tau_3 + \|Q(\tau_3) - S_1(\tau_3)\| \approx 3.37882
\]
where \( \tau_2 \approx 2.34029 \) and \( \tau_3 \approx 2.84758 \). \( \square \)
Improved Search Algorithm

In this section we improve the upper bound of Theorem 5.3.5 by 0.00495 additive term.

**Theorem 5.3.7.** PE$_3$ can be solved in time 3.37387.

The main idea can be described, at a high level, as a cost preservation technique. By the analysis of Algorithm SEARCH$_3(\alpha, \beta, \rho)$ for the value of parameters of $\alpha, \beta, \rho$ as in the proof of Theorem 5.3.5 we know that there are is a critical time window $[\tau_2, \tau_3]$ so that the total evacuation time is the same if the exit is found by $S_1$ either at time $\tau_2$ or $\tau_3$, and strictly less for time moments strictly in-between. In fact, during time $[\tau_2, 1 + \alpha + AK(\frac{\alpha + \beta}{2}, \rho)]$ $Q$ is executing phase 2, and in the time window $[1 + \alpha + AK(\frac{\alpha + \beta}{2}, \rho), \tau_3]$ $Q$ is executing phase 3 of SEARCH$_3(\alpha, \beta, \rho)$.

From the above, it is immediate that we can lower $Q$’s speed in the time window $[\tau_2, \tau_3]$ so that the evacuation time remains unchanged no matter when $S_1$ finds the exit in the same time interval (notably, $S_3$ has finished searching prior to $\tau_2$ and $\|Q(t) - S_1\| \geq \|Q(t) - S_2\|$). But this also implies that we must be able to maintain the evacuation time even if we preserve speed 1 for $Q$, that will in turn allow us to twist parameters $\alpha, \beta, \rho$, hopefully improving the worst case evacuation time. We show this improvement is possible by using the following technical observation:

**Theorem 5.3.8.** Consider point $Q = (q_1, q_2) \in \mathbb{R}^2$. Let $S(t)$ be the trajectory of an object $S$ moving at speed 1, where $t \geq 0$, and denote by $\phi$ the $(S, Q)$-critical angle at time $t = 0$. Assuming that $\cos(\phi) \geq 0$, then there is some $\tau > 0$, and a trajectory $Q(t) = (f(t), g(t))$ of a speed-1 object, where $t \geq 0$, so that $t + \|Q(t) - S(t)\|$ remains constant, for all $t \in [0, \tau]$. Moreover, $Q(t)$ can be determined by solving the system of differential equations

\[
(f'(t))^2 + (g'(t))^2 = 1 \tag{5.2}
\]

\[
t + \|Q(t) - S(t)\| = \|S(0) - Q\| \tag{5.3}
\]

\[
(f(0), g(0)) = (q_1, q_2). \tag{5.4}
\]

**Proof.** An object with trajectory $(f(t), g(t))$ satisfying (5.2) and (5.4) has speed 1 (by Lemma 5.2.2), and starts from point $Q = (q_1, q_2)$. We need to examine whether we can choose $f, g$ so as to satisfy (5.3).

By Lemma 5.2.7 such a trajectory $Q(t)$ exists exactly when we can guarantee that $\cos(\phi) + \cos(\theta) = 1$ over time $t$. When $t = 0$ we are given that $\cos(\phi) > 0$, hence there exists $\theta$ satisfying $\cos(\phi) + \cos(\theta) = 1$. This uniquely determines the velocity of $Q$ at $t = 0$.

By continuity of the velocities, there must exist a $\tau > 0$ such that $\cos(\phi) + \cos(\theta) = 1$ admits a solution for $\theta$ also as $\phi$ changes over time $t \in [0, \tau]$, in which time window the cosine of the $(S, Q(t))$-critical angle at time $t$ remains non-negative. \qed
Note that condition \( \cos(\phi) \geq 0 \) of Theorem 5.3.8 translates to \( \|S(t) - Q\| \) is not increasing at \( t = \tau \), i.e., that \( S \) does not move away from point \( Q \).

Now fix parameters \( \alpha, \beta, \rho \) together with the trajectories of \( S_1, S_2, S_3 \) as in the description of Algorithm SEARCH\(_3\)(\( \alpha, \beta, \rho \)). The description of our new algorithm N-SEARCH\(_3\)(\( \alpha, \beta, \rho \)) will be complete once we fix a new trajectory for \( Q \). Naming specific values for parameters \( \alpha, \beta, \rho \) will eventually prove Theorem 5.3.7. In order to do so, we introduce some further notation and conditions, denoted below by \( \) (Conditions i-iv), that we later make sure are satisfied.

Consider \( Q \)'s trajectory as in SEARCH\(_3\)(\( \alpha, \beta, \rho \)). Let \( \tau_0 \) denote a local maximum of \( t + \|Q(t) - S_1(t)\| \) as it reads for \( t \geq 0 \) with \( |t - 1 - \alpha| \leq AK(\frac{\alpha + \beta}{2}, \rho) \) (recall that in this time window, the expression is differentiable by Lemma 5.3.6), i.e.,

\[
|\tau_0 - 1 - \alpha| \leq AK(\frac{\alpha + \beta}{2}, \rho).
\]

(Condition i)

Set \( Q = Q(\tau_0) \), and assume that

“The cosine of the \((S, Q)\)-critical angle at time \( \tau_0 \) is non-negative.”

(Condition ii)

Then obtain from Theorem 5.3.8 trajectory \((f(t), g(t))\) that has the property that it preserves \( \tau_0 + \|Q(\tau_0) - S_1(\tau_0)\| \) in the time window \([\tau_0, \tau']\). Assume also that

“There is time \( \tau_1 \leq \tau' \) such that point \( K_1 := (f(\tau_1), g(\tau_1)) \) is equidistant from \( S_1(\tau_1), S_2(\tau_1) \),”

(Condition iii)

for the first time after time \( \tau_0 \), such that

\[
\tau_1 \leq 1 + \pi - \frac{\alpha + \beta}{2}.
\]

(Condition iv)

Then consider the following modification of SEARCH\(_3\)(\( \alpha, \beta, \rho \)), where the trajectories of \( S_1, S_2, S_3 \) remain unchanged, see also Figure 5.5

<table>
<thead>
<tr>
<th>Algorithm N-SEARCH(_3)(( \alpha, \beta, \rho ))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Robot</strong></td>
</tr>
<tr>
<td>Q</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
Figure 5.5: Algorithm SEARCH$_3(\alpha, \beta, \rho)$ depicted for the optimal parameters of the algorithm.

Note that in phase 2, $Q$ is not reaching (necessarily) point $K$ rather it moves toward it for a certain duration. The search time is still $1 + \pi - \frac{\alpha + \beta}{2}$. Trajectories of $S_1, S_2, S_3$ are continuous as before, and

$$\mathbb{I}(S_1) = \mathbb{I}(S_2) = [1, 1 + \pi - \frac{\alpha + \beta}{2}], \quad \mathbb{I}(S_3) = [1, 1 + \beta],$$

as well as $\mathbb{I}(Q) = [1, 1 + \alpha]$.

Condition i makes sure that while $Q$ is in phase 2, and before it reaches $K(\frac{\alpha + \beta}{2}, \rho)$, there is a time moment $\tau_0$ when the rate of change of $t + \|Q(t) - S_i(t)\|$ is 0. Together with condition ii, this implies that Theorem 5.3.8 applies. In fact, for the corresponding critical angles $\phi, \theta$ between $S_1, Q$ at time $\tau_0$, we have that $\cos(\phi) + \cos(\theta) = 1$ by construction. Hence the trajectory $(f(t), g(t))$ of phase 3 is well defined, and indeed, $Q$ jumps from phase 2 to phase 3 while $Q$ is still moving toward point $K$. Notably, $Q$’s trajectory is even differentiable at $t = \tau_0$ (but not necessarily at $t = \tau_1$). Then, Condition iii says that $Q$ eventually will enter phase 4, and that this will happen before $S_1, S_2$ finish the exploration of the circle. Overall, we conclude that in N-SEARCH$_3(\alpha, \rho)$, robots’ trajectories are continuous and feasible. An illustration of the above trajectories for certain values of $\alpha, \beta, \rho$ can be seen in Figure 5.5

Now we make some observations, in order to calculate the worst case evacuation time.

**Lemma 5.3.9.** Suppose that $\alpha \leq \beta, 1 + \beta \leq \tau_0$, and $1 + \pi - \frac{\alpha + \beta}{2} \geq \tau_1 + \|K_1 - C_{-\frac{\alpha + \beta}{2}}\|$ as well as Conditions i-iv are satisfied. Then the following functions are continuous and differentiable in each associated time intervals: $\|Q(x) - S_i(t)\|$ in $I_1 = \{t \geq 0 : \alpha \leq t - 1 \leq \beta\}$, $\|Q(x) - S_i(t)\|$ in $I_2 = \{t \geq 0 : 1 + \alpha \leq t \leq \tau_0\}$ and in
\[ I_3 = \left\{ t \geq 0 : |t - \tau_1| \leq \|K_1 - C_{-\alpha+\beta}\| \right\}. \] Moreover, the worst case evacuation time of N-SEARCH\(_3(\alpha, \beta, \rho)\) can be computed as

\[
\max \left\{ \sup_{t \in I_1} \{ t + \|Q(t) - S_3(t)\| \}, \sup_{t \in I_2} \{ t + \|Q(t) - S_1(t)\| \}, \sup_{t \in I_3} \{ t + \|Q(t) - S_1(t)\| \} \right\}.
\]

**Proof.** Conditions \( \alpha \leq \beta \) and \( 1 + \beta \leq \tau_0 \) mean that \( Q \) stops searching no later than \( S_3 \), and that when \( Q \) enters phase 3 after \( S_3 \) is done searching, respectively.

The line passing through \( O \) and \( C_{-(\alpha+\beta)/2} \), call it \( \epsilon \), has the property that each point of it, including \( K\left(\frac{\alpha+\beta}{2}, \rho\right) \) is equidistant from \( S_1 \), \( S_2 \). Moreover, while \( S_1 \), \( S_2 \) are searching, \( Q \) never goes above line \( \epsilon \). Also, while \( Q \) is executing phase 3, \( Q \) remains equidistant from \( S_1 \), \( S_2 \) and this is preserved for the remainder of the execution of the algorithm. As a result, \( S_2 \) can be ignored in the performance analysis, and when it comes to the case that \( S_1 \) finds the exit, the evacuation cost is given by the supremum of \( t + \|Q(t) - S_1(t)\| \) in the time interval \( I_2 \) or in the interval \( I_3 \). Note that in both intervals, the evacuation cost is continuous and differentiable, by construction.

If the exit is reported by \( S_3 \) then the evacuation cost is \( t + \|Q(t) - S_3(t)\| \) for \( t \in [1, 1 + \beta] \). However, it is easy to see that the cost is strictly increasing for all \( t \in [1, 1 + \alpha] \) (in fact it is linear). Since the evacuation cost is also continuous, we may restrict the analysis in interval \( I_1 \).

Lastly, observe that \( 1 + \pi - \frac{\alpha+\beta}{2} \geq \tau_1 + \|K_1 - C_{-\alpha+\beta}\| \) implies that \( S_1 \), \( S_2 \) reach point \( C_{-(\alpha+\beta)/2} \) no earlier than \( Q \). Hence \( Q \) waits at \( C_{-(\alpha+\beta)/2} \) till the search of the circle is over, which can be easily seen to induce the worse evacuation time after \( Q \) reaches \( C_{-(\alpha+\beta)/2} \).

Next we prove **Theorem 5.3.7** by fixing parameters \( \alpha, \beta, \rho \) for N-SEARCH\(_3(\alpha, \beta, \rho)\).

**Proof.** We choose \( \alpha = 0.27764, \beta = 1.29839, \rho = 0.68648 \). The trajectories of Figure 5.4 correspond exactly to those values. For these values we see that \( AK\left(\frac{\alpha+\beta}{2}, \rho\right) = 1.29041 \), while \( \tau_0 - \alpha - 1 = 1.04877 \). Hence the transition between phase 1 and phase 2 of \( Q \) is well defined.

The time that \( Q \) needs to reach \( C_{-\alpha+\beta} \) equals \( 1 + \tau_1 + \|K_1 - C_{-\alpha+\beta}\| = 3.18073 \), while the time that \( S_1 \), \( S_2 \) reach the same point is \( 1 + \pi - \frac{\alpha+\beta}{2} = 3.35358 \). Therefore we may attempt to solve numerically the differential equation of **Theorem 5.3.8**. It turns out that for the resulting trajectory \((f(t), g(t))\), and for \( \tau_1 = 2.89288 \), point \((f(\tau_1), g(\tau_1))\) is equidistant from \( S_1 \), \( S_2 \). Moreover, \( Q \) enters phase 4 at time \( \tau_1 = 2.89288 \), prior to \( 1 + \pi - \frac{\alpha+\beta}{2} \). Hence, Conditions i-iv are all met, as well as **Lemma 5.3.9** applies.

From the above, it is immediate that the worst evacuation time after \( Q \) reaches \( C_{-(\alpha+\beta)/2} \) equals \( 1 + \pi - \frac{\alpha+\beta}{2} = 3.35358 \). Hence, it remains to compute the maxima of \( t + \|Q(t) - S_3(t)\| \) in interval \( I_1 \), and of \( t + \|Q(t) - S_1(t)\| \) in intervals \( I_2, I_3 \).

155
To that end, when \( t \in I_1 = [1.27764, 2.29839] \) we have that

\[
Q(t) = (0.978782t - 2.25053, 0.261795 - 0.204905t)
\]

\[
S_3(t) = (\cos(t + 0.565563), \sin(t + 0.565563))
\]

so that \( t + \|Q(t) - S_3(t)\| \) becomes

\[
t + [(-0.204905t - \sin(t + 0.565563) + 0.261795)^2
\]

\[
+ (0.978782t - \cos(t + 0.565563) - 2.25053)^2]^{1/2}
\]

in which case

\[
\sup_{t \in I_1} \{t + \|Q(t) - S_3(t)\|\} = 1 + \beta + \|Q(1 + \beta) - S_3(1 + \beta)\| \approx 3.37387
\]

When \( t \in I_2 = [1.27764, 2.32641] \), \( Q \)'s trajectory is the same as in \( I_1 \) and

\[
S_1(t) = (\cos(2.56556 - t), \sin(2.56556 - t))
\]

so that \( t + \|Q(t) - S_1(t)\| \) becomes

\[
t + [(-\sin(2.56556 - t) - 0.204905t + 0.261795)^2
\]

\[
+ (-\cos(2.56556 - t) + 0.978782t - 2.25053)^2]^{1/2}
\]

When \( t \in I_3 = [2.89288, 3.18073] \), \( S_1 \)'s trajectory is the same as in \( I_2 \) and

\[
Q(t) = (0.705254t - 1.53797, 1.54604 - 0.708955t,
\]

\[
0.706399t - 1.53762, 1.5407 - 0.707814t),
\]

so that \( t + \|Q(t) - S_1(t)\| \) becomes

\[
t + [(-\sin(2.56556 - t) - 0.708955t + 1.54604)^2
\]

\[
+ (-\cos(2.56556 - t) + 0.705254t - 1.53797)^2]^{1/2}
\]

Numerically,

\[
\sup_{t \in I_2} \{t + \|Q(t) - S_1(t)\|\} = \tau_0 + \|Q(\tau_0) - S_1(\tau_0)\| = \tau_1 + \|Q(\tau_1) - S_1(\tau_1)\|
\]

\[
= \sup_{t \in I_3} \{t + \|Q(t) - S_1(t)\|\} \approx 3.37387.
\]

The reader may also consult Figure 5.5.
Figure 5.6: The queen must be in region $R$ at time $f(s_3)$. Here $s_3 = E$ and $q_3 = F$.

5.4 Lower Bounds

In this section we derive lower bounds for evacuation. In Section 5.4.1 we treat the case of $n = 1$ (see Theorem 5.4.1) and in Section 5.4.2 we treat the case of $n = 2$ and 3 (see Theorem 5.4.3).

5.4.1 Lower Bound for $\text{PE}_1$

We will derive the lower bound using an adversarial argument placing the exit at an unknown vertex of a regular hexagon.

Theorem 5.4.1. The worst-case evacuation time for $\text{PE}_1$ is at least $3 + \pi/6 + \sqrt{3}/2 \approx 4.3896$.

Proof. At time $1 + \pi/6$, at most $\pi/3$ of the perimeter of the circle can have been explored by the queen and servant. Thus, there is a regular hexagon, none of whose vertices have been explored. If the exit is at one of these vertices, by Theorem 5.4.2 it takes $2 + \sqrt{3}/2$ for the queen to evacuate. The total time is $1 + \pi/6 + 2 + \sqrt{3}/2$.

Next we proceed to provide a lower bound on a unit-side hexagon. Label the vertices of the hexagon $V$ as $A, \ldots, F$ as shown in Figure 5.6. Fix an evacuation algorithm $\mathcal{A}$. For any vertex $v$ of the hexagon, we call $f(v)$ the time of first visit of the vertex $v$ by either the servant or the queen, according to algorithm $\mathcal{A}$. We call $q(v)$ the time that the queen gets to the vertex $v$. Clearly, $q(v) \geq f(v)$, and if the queen arrives at the vertex no later than the servant, $q(v) = f(v)$. 

157
Theorem 5.4.2. For any algorithm \( A \) and any initial position of the queen, the evacuation time for the queen when the exit is at one of the vertices of the hexagon is \( \max_{v \in V} \{q(v)\} \geq 2 + \sqrt{3}/2 \).

Proof. Suppose there is an algorithm in which the queen can always evacuate in time \( < 2 + \sqrt{3}/2 \). Consider the trajectories of the servant and the queen. If either the queen or the servant are the first to visit 4 vertices, then for the fourth such vertex \( v \), we have \( f(v) \geq 3 \), a contradiction. Therefore, the queen is the first to visit three vertices, and the servant is the first to visit three vertices. We denote the three vertices visited first by the servant as \( s_1, s_2, s_3 \) (in the order they are visited) and the three vertices visited first by the queen as \( q_1, q_2, q_3 \), and note that they are all distinct.

Notice that neither \( s_3 \) nor \( q_3 \) can be visited before time 2, that is, \( f(s_3), f(q_3) \geq 2 \). If \( f(q_3) \leq f(s_3) \), then we place the exit at \( s_3 \), and the queen needs time at least 1 to get to \( s_3 \), which implies that \( T \geq q(s_3) \geq f(q_3) + 1 \geq 3 \), a contradiction. We conclude that at time \( f(s_3) \), the queen is yet to visit \( q_3 \). Since the exit can be at either \( s_3 \) or \( q_3 \), at time \( f(s_3) \), the queen must be at distance \( < 2 + \sqrt{3}/2 - f(s_3) \leq \sqrt{3}/2 \) from both \( s_3 \) and \( q_3 \).

Assume without loss of generality that \( s_3 = E \) (see Figure 5.6). Since \( A, B, D \) are all at distance at least \( \sqrt{3} \) from \( E \), we conclude that \( q_3 \) is either \( C \) or \( F \). Assume without loss of generality that \( q_3 = F \). Let \( R \) denote the lens-shaped region that is at distance \( < 2 + \sqrt{3}/2 - f(s_3) \) from both \( E \) and \( F \). Recall that at time \( f(s_3) \), the queen must be inside the region \( R \). Notice that if \( f(s_3) \geq 1.5 + \sqrt{3}/2 \), the region \( R \) is empty, yielding a contradiction. So it must be that \( 2 \leq f(s_3) < 1.5 + \sqrt{3}/2 \).

We now work backwards to deduce the trajectories of the servant and the queen. Clearly \( s_2 \neq F \) since \( q_3 = F \). If \( s_2 \neq C \), then \( f(s_3) \geq \sqrt{3} + 1 > 1.5 + \sqrt{3}/2 \), a contradiction. Therefore, \( s_2 = C \). By the same reasoning, \( s_1 = A \). Therefore, the queen is the first to visit \( D \) and \( B \). If \( q_1 = D \) and \( q_2 = B \), we place the exit at \( E \); since \( f(q_2) \geq 1 \) and \( \text{dist}(B, E) = 2 \), we have \( T \geq q(E) \geq 3 \), a contradiction. Thus, \( q_2 = D \) and \( q_1 = B \).

Consider the location of the queen at time 1. If she is at distance \( \geq 1 + \sqrt{3}/2 \) from \( C \) at time 1, then if the exit is at \( C \), \( q(C) \geq 2 + \sqrt{3}/2 \). So at time 1, the queen must be at distance \( < 1 + \sqrt{3}/2 \) from \( C \) and consequently she is at distance \( \geq 1 - \sqrt{3}/2 \) from \( D \). Therefore \( f(q_2) = f(D) \geq 2 - \sqrt{3}/2 \). Also, \( f(D) < 1.5 \) since if the queen reaches \( D \) at or after time 1.5, she cannot reach the region \( R \) before time \( 1.5 + \sqrt{3}/2 > f(s_3) \). So \( f(D) \leq f(s_3) \). If the exit is at \( E = s_3 \), the queen cannot reach the exit before time \( f(D) + \text{dist}(D, E) \geq 2 - \sqrt{3}/2 + \sqrt{3} = 2 + \sqrt{3} \), concluding the proof by contradiction. \( \square \)

We remark that the above bound is optimal, and is achieved by the algorithm depicted in Figure 5.7.
5.4.2 Lower Bounds for PE_{2} and PE_{3} - Proof Outline

In the case of \( n = 2 \) and \( n = 3 \) the proof is rather technical. Next we present a high level outline as to why the lower bounds hold.

**Theorem 5.4.3.** The worst-case evacuation time for PE_{2} is at least 3.6307 and for PE_{3} at least 3.2017.

Throughout this section we will use \( T \) to refer to the evacuation time of an arbitrary algorithm and use \( U \) to refer to the unit circle which must be evacuated.

The main thrust of the proof relies on a simple idea – the queen should aid in the exploration of \( U \). This is immediately evident for the particular case of \( n = 2 \) since, if the queen does not explore, it will take time at least \( 1 + \pi \) for the servants to search all of \( U \) and we already have an upper bound smaller than this (Theorem 5.3.3). Thus, a general overview of the proof is as follows: we show that in order to evacuate in time \( T \) the queen must explore some minimum length of the perimeter of \( U \). We will then demonstrate that the queen is not able to explore this minimum amount in any algorithm with evacuation time smaller than what is given in Theorem 5.4.3.

To be concrete, consider the case of \( n = 2 \) and assume that we have an algorithm with evacuation time \( T < 1 + \pi \). Then, in order for the robots to have explored all of \( U \) in time \( T \), the queen must explore a subset of the perimeter of total length at least \( 2(1 + \pi - T) \). Intuitively, this minimum length of perimeter will increase in size as \( T \) decreases.

Now consider that it is not possible for the queen to always remain on the perimeter (indeed, in each of the algorithms presented, the queen leaves the perimeter). To see why this is consider that, in any algorithm with evacuation time \( T \), it must be the case that all unexplored points of \( U \) are located a distance no more than \( T - t \) from the queen at all times \( t \leq T \). If the queen is on the perimeter at any time \( t \) satisfying \( T - t \leq 2 \), then, there will
be some arc \( \theta(t, T) \subset U \) (see Lemma 5.4.4) such that all points of \( \theta(t, T) \) are at a distance at least \( T - t \) from the queen. Thus, if the queen is to be on the perimeter at the time \( t \) we can conclude that all of the arc \( \theta(t, T) \) must have already been discovered. However, we will find (see Lemma 5.4.5) that \( \theta(t, T) \) will often grow at a rate much larger than the robots can collectively explore and at some point the queen will have to leave the perimeter. In fact, there will be an interval of time during which it is not possible for the queen to be exploring and this in turn implies that there is a maximum amount of perimeter that can be explored by the queen. Intuitively, the maximum length of perimeter that can be explored by the queen will decrease as \( T \) decreases. The lower bound will result by balancing the minimum amount of perimeter the queen needs to search and the maximum amount of perimeter that the queen is able to search.

The above argument will need a slight modification in the case of \( n = 3 \). In this case we will show that there is some critical time \( t_\ast \) before which the queen must have explored some minimum amount of perimeter. Again, the lower bound follows by balancing the maximum amount of perimeter the queen can explore by the time \( t_\ast \) and the minimum amount of perimeter the queen needs to explore before the time \( t_\ast \).

### 5.4.3 Lower Bounds for PE\(_2\) and PE\(_3\) - Proof Details

In this section we present the complete details of the proofs for the lower bounds in the cases \( n = 2 \) and \( n = 3 \). Throughout this section we will use \( T \) to refer to the evacuation time of an arbitrary algorithm and use \( U \) to refer to the unit circle which must be evacuated.

The idea of the proofs are to bound the amount of perimeter the queen can search for a given evacuation time \( T \) and then show that the queen must search a minimum amount of the perimeter in order to achieve the evacuation time \( T \). The lower bounds result by balancing the minimum amount of perimeter the queen must search with the maximum amount of perimeter the queen can search.

We begin with two lemmas which will be used for both the \( n = 2 \) and \( n = 3 \) bounds. Their necessity will become apparent shortly.

**Lemma 5.4.4.** Consider any \( r < 2 \) and a point \( P \in U \). Define the circle \( D_P \) as the disk centered on \( P \) with radius \( r \). Then the subset of the perimeter of \( U \) which is not contained in \( D_P \) has length \( \theta = 4 \cos^{-1} \left( \frac{r}{2} \right) \).

**Proof.** Without loss of generality assume that the point \( P \) is located at \((-1, 0)\). Since \( r < 2 \) the disks \( U \) and \( D_P \) will intersect at two boundary points \( A \) and \( B \) between which the distance along the perimeter of \( U \) is \( \theta \). This situation is depicted in Figure 5.8. Referring to this figure, one can easily observe that \( r = 2 \sin \left( \frac{\pi}{2} - \frac{\theta}{4} \right) = 2 \cos \left( \frac{\theta}{4} \right) \). Rearranging for \( \theta \) we find that \( \theta = 4 \cos^{-1} \left( \frac{r}{2} \right) \). \( \square \)
Lemma 5.4.5. Consider the function \( \theta(t, T) = 4 \cos^{-1}\left(\frac{T-t}{2}\right) \) with \( T > 0 \). Then \( \frac{d\theta}{dt} > 2 \) for all \( t \) satisfying \( T - 2 < t < T \) and \( \frac{d\theta}{dT} > 3 \) for \( t \) satisfying \( T - 2 < t < T - \frac{2}{3}\sqrt{5} \). Furthermore, \( \frac{d\theta}{dT} < -2 \) for all \( T - 2 < t < T \).

Proof. The rate of change of \( \theta(t, T) \) with \( t \) is given by

\[
\frac{d\theta}{dt} = \frac{4}{\sqrt{4 - (T - t)^2}}.
\]

From this relation it is simple to confirm that \( \frac{d\theta}{dt} > 2 \) for \( T - 2 < t < T \) and that \( \frac{d\theta}{dT} > 3 \) for \( T - 2 < t < T - \frac{2}{3}\sqrt{5} \). It should also be obvious by the symmetry of \( T \) and \( t \) in the function \( \theta(t, T) \) that \( \frac{d\theta}{dT} < -2 \) for all \( T - 2 < t < T \). 

Lower bound for \( n = 2 \)

We begin with the main result of the section.

Theorem 5.4.6. For \( n = 2 \) and any algorithm the queen cannot be evacuated in time less than \( T_2 \) which is the solution to the equations

\[
\tau = T_2 - 2 \cos \left(\frac{T_2 - 1}{2}\right)
\]

\[
t_* = \frac{1}{2}(T_2 + 1)
\]
\[ T_2 = t_* + 2\cos\left(\frac{2t_* + \tau}{4} - \frac{3}{4}\right). \]

Solving these equations numerically gives \( \tau \approx 1.7815 \), \( t_* \approx 2.3154 \), and \( T_2 \approx 3.6307 \).

We will see that the queen cannot be located on the perimeter of the circle during the interval of time \((\tau, t_*)\) and thus \( \tau - 1 \) represents the maximum amount of perimeter that can be explored by the queen before the time \( t_* \). The time \( t_* \) is chosen such that for all \( T_2 < T \) a solution to the equations in Theorem 5.4.6 do not exist, and, as such, \( \tau - 1 \) will represent the maximum length of the perimeter that can be explored by the queen. In the following lemma we show that the queen must explore a length of the perimeter greater than \( \tau - 1 \) in order to evacuate in time less than \( T_2 \).

**Lemma 5.4.7.** For \( n = 2 \) and any evacuation algorithm with \( T < 1 + \pi \), the queen must explore a subset of the perimeter of length \( y \geq 2(1 + \pi - T) \). In particular, if \( T < T_2 \), we need \( y > 2(1 + \pi - T_2) \approx 1.0217 \).

**Proof.** If the queen explores a subset of the perimeter of length \( y \) then the robots will take time \( 1 + \frac{2\pi - y}{2} \) to explore the circle. The robots need to at least explore the entire circle in time \( T \) and therefore \( 1 + \frac{2\pi - y}{2} \leq T \), or, equivalently, \( y \geq 2(1 + \pi - T_2) \). For \( T < T_2 \approx 3.6307 \) we need \( y > 1.0217 \).

We will now show that the maximum length of perimeter the queen can explore is less than \( \tau - 1 \) if \( T < T_2 \). This will be the goal of the next two lemmas.

**Lemma 5.4.8.** Consider the equation \( T = t + 2\cos\left(\frac{1}{2}(t-1) + \frac{1}{4}\alpha\right) \) with \( T > 0 \), \( \alpha \) satisfying \( 0 < \alpha \leq t \) and \( t \) satisfying \( 1 < t \leq T \). Then \( \frac{dt}{dT} \geq \frac{1}{2} \), and, if \( 0 < t < 1 + 2\pi - \frac{\alpha}{2} \) then \( \frac{dt}{d\alpha} > 0 \).

**Proof.** Implicitly differentiating the equation \( T = t + 2\cos\left(\frac{1}{2}(t-1) + \frac{1}{4}\alpha\right) \) with respect to \( T \) gives us

\[ \frac{dt}{dT} = \frac{1}{1 - \sin\left(\frac{1}{2}(t-1) + \frac{1}{4}\alpha\right)}. \]

Since the sine function ranges from \(-1\) to \(1\) we can easily see that \( \frac{dt}{dT} \geq \frac{1}{2} \).

Implicitly differentiating the equation \( T = t + 2\cos\left(\frac{1}{2}(t-1) + \frac{1}{4}\alpha\right) \) with respect to \( \alpha \) gives us

\[ \frac{dt}{d\alpha} = \frac{1}{2} \cdot \frac{\sin\left(\frac{1}{2}(t-1) + \frac{1}{4}\alpha\right)}{1 - \sin\left(\frac{1}{2}(t-1) + \frac{1}{4}\alpha\right)}. \]

We can easily see that the denominator of \( \frac{dt}{d\alpha} \) will never be negative and thus \( \frac{dt}{d\alpha} > 0 \) provided that the numerator is positive. This clearly occurs for \( \frac{1}{2}(t-1) + \frac{1}{4}\alpha < \pi \) or \( t < 1 + 2\pi - \frac{\alpha}{2} \). \( \square \)
Lemma 5.4.9. Define \( \tau \) as in Theorem [5.4.6]. Then, for \( n = 2 \) and any evacuation algorithm with \( T < T_2 \), the queen cannot explore a subset of the perimeter with length \( y > \tau - 1 \).

**Proof.** We start with an observation: if the queen is to evacuate in time \( T \), then, at any time \( t < T \), all points of \( U \) that are a distance greater than \( T - t \) from the queen must be explored by a robot. If the queen is located on the perimeter at the time \( t > T - 2 \) then by Lemma [5.4.4] there is an arc of length

\[
\theta(t, T) = 4 \cos^{-1}\left(\frac{T - t}{2}\right)
\]

all points of which lie a distance greater than \( T - t \) from the queen (as an abuse of notation we will refer to the arc with length \( \theta(t, T) \) as \( \theta(t, T) \)). Thus, in order for the queen to be on the perimeter at the time \( t \), the arc \( \theta(t, T) \) must be explored. As we have 3 robots in total the maximum length of \( \theta(t, T) \) that can be explored at any time \( t \) is \( 3(t - 1) \). However, we claim that the queen cannot have explored any of \( \theta(t, T) \) if the time \( t \) satisfies \( t < \frac{1}{2}(T + 1) \). Indeed, observe that the endpoints of \( \theta(t, T) \) lie a distance \( T - t \) away from the queen (by definition) and the queen – who took a unit of time to reach the perimeter – could have explored a point on the perimeter at most a distance \( t - 1 \) from her current position. Thus, if \( t - 1 < T - t \), or, alternatively, \( t < \frac{1}{2}(T + 1) \), the queen cannot have explored any of the arc \( \theta(t, T) \). We must therefore have \( \theta(t, T) \leq 2(t - 1) \) for times \( t \) that satisfy \( t < \frac{1}{2}(T + 1) \).

We note that there is a trivial lower bound of \( 1 + \frac{2\pi}{3} > 3 \) and thus we can assume that \( T > 3 \). We make the following claim: if \( T < T_2 \) then the smallest time \( t_0 > 0 \) solving \( \theta(t_0, T) = 2(t_0 - 1) \) satisfies \( \frac{d\theta}{dt}_{t=t_0} > 2 \) and \( t_0 < \frac{1}{2}(T + 1) \). We note that, if this is the case, the queen will have to leave the perimeter at the time \( t_0 \) (since she has not explored any of the arc \( \theta(t, T) \) and, immediately after the time \( t_0 \), \( \theta(t, T) \) will be too large to have been explored by the servants alone).

We first show that \( t_0 < \frac{1}{2}(T + 1) \). To this end we rearrange the equation \( \theta(t_0) = 2(t_0 - 1) \) to get

\[
t_0 = T - 2 \cos\left(\frac{t_0 - 1}{2}\right)
\]

which is the definition of \( \tau \) in Theorem [5.4.6] (in the case that \( T = T_2 \)). One can easily confirm that in the case of \( T = T_2 \) we have \( \frac{d\theta}{dt}_{t=t} \approx 5.2511 > 2 \) and \( \tau < \frac{1}{2}(T + 1) \). Now observe that \( \theta(t, T) \) is a decreasing function of \( T \) and this implies that for \( T < T_2 \) we have \( \theta(\tau, T) > \theta(\tau, T_2) \). We can therefore conclude that the time \( t_0 \) must occur earlier than the time \( \tau \). We note that \( \tau < 2 \) and, since \( T \geq 3 \), we have \( \tau < \frac{1}{2}(T + 1) \). Since \( t_0 < \tau \) we can conclude that \( t_0 < \frac{1}{2}(T + 1) \).

The second part of the claim follows directly from Lemma [5.4.5] where we show that \( \frac{d\theta}{dt} > 2 \) for all \( t \) satisfying \( T - 2 < t < T \).

As the queen must leave the perimeter at the time \( t_0 < \tau \), by Lemma [5.4.7] we can say that the queen must be able to return to the perimeter and explore before the algorithm
terminates. Thus, consider the smallest time $t_1 > t_0$ at which the queen may return to the perimeter. In order for the queen to be on the perimeter we will still need the arc $\theta(t, \mathcal{T})$ to be completely explored. However, in this case it may be possible that $t_1 \geq \frac{1}{2}(\mathcal{T} + 1)$ and as such the queen could have explored at most a length $t_0 - 1$ of $\theta(t, \mathcal{T})$ at the time $t_1$. We can therefore conclude that $t_1$ will satisfy $\theta(t_1) = 2(t_1 - 1) + y$ with $y = 0$ if $t_1 < \frac{1}{2}(\mathcal{T} + 1)$, and $y \leq t_0 - 1$ if $t_1 \geq \frac{1}{2}(\mathcal{T} + 1)$. Writing the equation $\theta(t_1) = 2(t_1 - 1) + y$ in full and rearranging we find that

$$t_1 = \mathcal{T} - 2 \cos \left( \frac{1}{2}(t_1 - 1) + \frac{1}{4}y \right).$$

We will now consider the cases $t_1 < \frac{1}{2}(\mathcal{T} + 1)$ and $t_1 \geq \frac{1}{2}(\mathcal{T} + 1)$ separately.

**Case 1:** $t_1 < \frac{1}{2}(\mathcal{T} + 1)$

In this case $t_1$ can be observed to satisfy the same equation as $t_0$. We claim that this is not possible if $t_1 > t_0$. Indeed, by Lemma 5.4.5 we have $\frac{dt}{dy} > 2$ and the arc $\theta(t, \mathcal{T})$ will always grow at a rate larger than the servants alone can explore. Thus, a solution to the equation $\theta(t_1) = 2(t_1 - 1)$ with $t_1 > t_0$ does not exist. This implies that the queen can explore a maximum subset of the perimeter of total length $t_0 - 1 < \tau - 1$ if $t_1 < \frac{1}{2}(\mathcal{T} + 1)$.

**Case 2:** $t_1 \geq \frac{1}{2}(\mathcal{T} + 1)$

In this case $t_1$ satisfies

$$t_1 = \mathcal{T} - 2 \cos \left( \frac{1}{2}(t_1 - 1) + \frac{1}{4}y \right).$$

Although it can be confirmed that $\frac{dt_1}{dy} > 0$ (see Lemma 5.4.8) we will show that, even when $t_1$ is as large as possible (i.e. $y = t_0 - 1$), we cannot have $t_1 \geq \frac{1}{2}(\mathcal{T} + 1)$. Thus we assume that $t_1$ satisfies

$$t_1 = \mathcal{T} - 2 \cos \left( \frac{1}{2}(t_1 - 1) + \frac{1}{4}(t_0 - 1) \right).$$

Now write $t_1 = t_1(\mathcal{T})$ as a function of $\mathcal{T}$ and note that, by Lemma 5.4.8, we have $\frac{dt_1}{dT} > \frac{1}{2}$. Using this we can say that $t_1(\mathcal{T}_2) - t_1(\mathcal{T}) > \frac{1}{2}(\mathcal{T}_2 - \mathcal{T})$. By definition of $\mathcal{T}_2$ we have $t_1(\mathcal{T}_2) = \frac{1}{2}(\mathcal{T}_2 + 1)$ and we can therefore write $\frac{1}{2}(\mathcal{T}_2 + 1) - t_1(\mathcal{T}) > \frac{1}{2}(\mathcal{T}_2 - \mathcal{T})$. Rearranging this inequality gives us $t_1(\mathcal{T}) < \frac{1}{2}(\mathcal{T} + 1)$ which contradicts with our assumption that $t_1 \geq \frac{1}{2}(\mathcal{T} + 1)$ and we must conclude that $t_1 < \frac{1}{2}(\mathcal{T} + 1)$. This concludes the proof.

At this point the proof of Theorem 5.4.6 is rather trivial.

**Proof.** Assume that we have an algorithm with evacuation time $\mathcal{T} < \mathcal{T}_2$. Then, by Lemma 5.4.7, the queen must explore a subset of the perimeter of length at least $y > 1.0217$. However, by Lemma 5.4.9, the queen can only explore a subset of the perimeter of length $y < \tau - 1 \approx 0.7815$ if $\mathcal{T} < \mathcal{T}_2$. It is therefore not possible for the queen to evacuate in time less than $\mathcal{T}_2$. 

164
Lower bound for \( n = 3 \)

The main result of this section is given below:

**Theorem 5.4.10.** For \( n = 3 \) and any algorithm the queen cannot be evacuated in time less than \( T_3 \) which is the solution to the equations

\[
\begin{align*}
\tau &= T_3 - 2 \cos \left( \frac{3}{4} (\tau - 1) \right) \\
t_\ast &= 1 + \frac{2}{3} \cos^{-1} \left( \frac{-2}{3} \right) - \frac{(\tau - 1)}{3} \\
T_3 &= t_\ast + \sin \left( \frac{3(t_\ast - 1) + (\tau - 1)}{2} \right).
\end{align*}
\]

Solving these equations numerically gives \( \tau \approx 1.2319 \), \( t_\ast \approx 2.4564 \), and \( T_3 \approx 3.2017 \).

As before, \( \tau \) represents the beginning of an interval of time during which the queen cannot be located on the perimeter. In this case, however, \( t_\ast \) is not the first time at which it is possible for the queen to return to the perimeter. Instead it represents a particularly critical time of any algorithm with \( n = 3 \) at which the evacuation time is maximized (although it will happen that \( t_\ast \) occurs before the queen can return to the perimeter). We will show that the queen must explore a subset of the perimeter with total length more than \( \tau - 1 \) before the time \( t_\ast \) in order to evacuate in time less than \( T_2 \).

We begin with a lemma that was first introduced in [57]:

**Lemma 5.4.11.** Consider a perimeter of a disk whose subset of total length \( u + \epsilon > 0 \) has not been explored for some \( \epsilon > 0 \) and \( \pi \geq u > 0 \). Then there exist two unexplored boundary points between which the distance along the perimeter is at least \( u \).

This next lemma is used to determine the critical time \( t_\ast \).

**Lemma 5.4.12.** Consider an evacuation algorithm with \( n \) servants and assume that at the time \( t \) the queen has explored a total subset of the perimeter of length \( y \). Then, for \( x \) and \( y \) satisfying \( 1 + \frac{\pi - y}{n} \leq t \leq 1 + \frac{2\pi - y}{n} \), it takes time at least \( T = t + \sin \left( \frac{n(t-1)+y}{2} \right) \) to evacuate the queen.

**Proof.** Consider an algorithm with evacuation time \( T \) and with \( n \) servants. Then, at the time \( t \), the total length of perimeter that the robots have explored is at most \( n(t-1)+y \geq \pi \) (since each robot may search at a maximum speed of one, the queen has explored a subset of length \( y \), and the robots need at least a unit of time to reach the perimeter). Thus, by Lemma 5.4.11, there exists two unexplored boundary points between which the distance along the perimeter is at least \( 2\pi - n(t-1) - y - \epsilon \) for any \( \epsilon > 0 \). The chord connecting these points has length at least \( 2 \sin \left( \frac{\pi - n(t-1)+y}{2} - \frac{\epsilon}{2} \right) \) and an adversary may place the exit at either endpoint.
of this chord. The queen will therefore take at least \( \sin \left( \pi - \frac{n(t - 1) + y}{2} - \frac{\epsilon}{2} \right) \) more time to evacuate and the total evacuation time will be at least \( t + \sin \left( \frac{n(t - 1) + y}{2} - \frac{\epsilon}{2} \right) \). As this is true for any \( \epsilon > 0 \) taking the limit \( \epsilon \to 0 \) we obtain

\[
\mathcal{T} \geq t + \sin \left( \frac{\pi - n(t - 1) + y}{2} \right) = t + \sin \left( \frac{n(t - 1) + y}{2} \right).
\]

\[\square\]

In the next two lemmas we show that in order to evacuate in time \( \mathcal{T} < \mathcal{T}_2 \) the queen must explore a length of the perimeter greater than \( \tau - 1 \) and then demonstrate that this is not possible.

**Lemma 5.4.13.** Define \( \tau \) and \( t_* \) as in Theorem 5.4.10. Then, for \( n = 3 \) and any evacuation algorithm with \( \mathcal{T} < \mathcal{T}_3 \), the queen must explore a subset of \( \mathcal{U} \) with total length \( y > \tau - 1 \) before the time \( t_* \).

**Proof.** Consider an algorithm with evacuation time \( \mathcal{T} < \mathcal{T}_3 \). We make the assumption that the queen has only explored a subset of total length \( y < \tau - 1 \) at the time \( t_* \) and show that this leads to a contradiction.

Observe that \( t_* \) satisfies \( 1 + \frac{\pi - y}{3} \leq t_* \leq 1 + \frac{2\pi - y}{3} \) for all \( y \) satisfying \( 0 \leq y \leq \tau - 1 \) and thus, by Lemma 5.4.12, we can write

\[
\mathcal{T} \geq t_* + \sin \left( \frac{3(t_* - 1) + y}{2} \right).
\]

Since \( \mathcal{T} < \mathcal{T}_3 \) we also have

\[
\mathcal{T}_3 > t_* + \sin \left( \frac{3(t_* - 1) + y}{2} \right).
\]

Since \( \mathcal{T}_3 = t_* + \sin \left( \frac{3(t_* - 1) + (\tau - 1)}{2} \right) \) we further have

\[
\sin \left( \frac{3(t_* - 1) + (\tau - 1)}{2} \right) > \sin \left( \frac{3(t_* - 1) + y}{2} \right),
\]

Finally, since \( t_* \geq 1 + \frac{\pi - y}{3} \) we know that \( \sin \left( \frac{3(t_* - 1) + y}{2} \right) \) is a decreasing function of its argument and thus we get

\[
\frac{3(t_* - 1) + (\tau - 1)}{2} < \frac{3(t_* - 1) + y}{2}
\]

which implies that \( y > \tau - 1 \) which contradicts with our assumption that \( y < \tau - 1 \). \[\square\]
Lemma 5.4.14. Define \( \tau \) and \( t^* \) as in Theorem 5.4.10. Then, for \( n = 3 \) and any evacuation algorithm with \( T < T_3 \), the queen cannot explore a subset of the perimeter with length \( y > \tau - 1 \) before the time \( t^* \).

Proof. As was the case for \( n = 2 \), if the queen is to be on the perimeter at the time \( t \) then all of the arc \( \theta(t, T) = 4 \cos^{-1} \left( \frac{T-t}{2} \right) \) must be explored. Since we have 4 robots in total, the maximum length of arc that can be explored at any time \( t \) is \( 4(t-1) \). However, we can again say that the queen cannot search any of the arc \( \theta(t) \) if \( t \leq \frac{1}{2}(T+1) \). We must therefore have \( \theta(t, T) \leq 3(t-1) \) for times \( t \) that satisfy \( t < \frac{1}{2}(T+1) \).

Assume first that \( T \geq 3 \). We make the following claim: if \( 3 \leq T < T_3 \) then the smallest time \( t_0 > 0 \) solving \( \theta(t_0, T) = 3(t_0-1) \) satisfies \( \frac{d\theta}{dt} \bigg|_{t=t_0} > 3 \) and \( t_0 < \frac{1}{2}(T+1) \). If this is the case the queen will have to leave the perimeter at the time \( t_0 \).

We first demonstrate that \( t_0 < \frac{1}{2}(T+1) \). Let us rearrange the equation \( \theta(t_0, T) = 3(t_0-1) \) to get

\[ t_0 = T - 2 \cos \left( \frac{3}{4} (t_0 - 1) \right) \]

which is the definition of \( \tau \) in Theorem 5.4.10 (in the case that \( T = T_3 \)). One can easily confirm that in the case of \( T = T_3 \), both \( \frac{d\theta}{dt} \big|_{t=\tau} > 3 \) and \( \tau < \frac{1}{2}(T+1) \). Now observe that \( \theta(t, T) \) is a decreasing function of \( T \) and this implies that for \( T < T_3 \) we have \( \theta(\tau, T) > \theta(\tau, T_3) \). The time \( t_0 \) must therefore occur earlier than the time \( \tau \). We note that \( \tau < 2 \) and, since we are assuming that \( T \geq 3 \), we have \( \tau < \frac{1}{2}(T+1) \). Since \( t_0 < \tau \) we can finally conclude that \( t_0 < \frac{1}{2}(T+1) \).

The second part of the claim follows from Lemma 5.4.5 if we can show that \( t_0 < T - \frac{2}{3} \sqrt{5} \). We note that \( T \geq 3 \) and thus \( T - \frac{2}{3} \sqrt{5} \geq 1.5093 \). Since \( \tau \approx 1.2319 \) and \( t_0 < \tau \) we can clearly see that \( t_0 < T - \frac{2}{3} \sqrt{5} \).

If \( T < 3 \) then it should be obvious that the queen cannot even be at the perimeter at the time \( t = 1 \). Thus, in this case, we take \( t_0 = 1 \).

Since the queen must leave the perimeter at the time \( t_0 < \tau \), by Lemma 5.4.13, we know that the queen must be able to return to the perimeter and explore before the time \( t^* \). We claim that this is not possible. Indeed, observe that the queen cannot return to the perimeter until the earliest time \( t > t_0 \) at which \( \theta(t) = 3(t-1) + y \) (where we have set \( y < \tau - 1 \) as the length of the arc \( \theta(t) \) explored by the queen). Thus, in order for the queen to have returned to the perimeter before the time \( t^* \) we must have \( \theta(t^*) \leq 3(t-1) + y \). However, since \( T < T_3 \) we have

\[ \theta(t^*) = 4 \cos^{-1} \left( \frac{T-t^*}{2} \right) > 4 \cos^{-1} \left( \frac{T_3-t^*}{2} \right). \]

We note that

\[ T_3 - t^* = \sin \left( \frac{3(t^* - 1) + (\tau - 1)}{2} \right) = \sin \left( \cos^{-1} \left( \frac{-2}{3} \right) \right) = \sqrt{\frac{5}{9}} \]

167
and thus
\[ \theta(t_*) > 4 \cos^{-1}\left(\frac{\sqrt{5}}{6}\right) \approx 4.7556. \]

Since \( \tau \approx 1.2319 \), and \( t_* \approx 2.4564 \) we have
\[ 3(t_* - 1) + y \leq 3(t_* - 1) + (\tau - 1) \approx 4.6010. \]

We can therefore see that it is not the case that \( \theta(t_*) \leq 3(t - 1) + y \) and thus the queen cannot have returned to the perimeter before the time \( t_* \). We can finally conclude that the queen can only explore a subset of the perimeter of length \( t_0 - 1 < \tau - 1 \) before the time \( t_* \).

At this point the proof of Theorem 5.4.10 is trivial.

**Proof.** Assume we have an algorithm with evacuation time \( T < \mathcal{T}_3 \). Then, by Lemma 5.4.13, the queen must explore a subset of the perimeter of length at least \( \tau - 1 \) by the time \( t_* \). However, by Lemma 5.4.14, the queen can only explore a subset of the perimeter of length \( y < \tau - 1 \) if \( T < \mathcal{T}_3 \). We must therefore conclude that it is not possible for the queen to evacuate in time less than \( \mathcal{T}_3 \).

### 5.5 Conclusion

We considered an evacuation problem concerning priority searching on the perimeter of a unit disk where only one robot (the queen) needs to reach the exit. In addition to the queen, there are \( n \leq 3 \) other robots (servants) aiding the queen by contributing to the exploration of the disk but which do not need to evacuate. We proposed evacuation algorithms and studied non-trivial tradeoffs on the queen’s evacuation time depending on the number \( n \) of servants. In addition to analyzing tradeoffs and improving the bounds obtained for the wireless communication model, an interesting open problem would be to investigate other communication models, e.g., the face-to-face model studied in [57] and elsewhere.

### References


Chapter 6

Priority Evacuation from a Disk: The case of $n \geq 4$

This chapter includes the paper “Priority Evacuation from a Disk: The case of $n \geq 4$” which appears in Theoretical Computer Science and which is the journal version of a paper first appearing in the proceedings of SIROCCO2018. As the title of the paper suggests, this chapter will concern the priority evacuation problem when there are large numbers of agents.
Priority Evacuation from a Disk: the case of $n \geq 4$

Jurek Czyzowicz$^1$, Konstantinos Georgiou$^2$, Ryan Killick$^3$, Evangelos Kranakis$^3$, Danny Krizanc$^4$, Lata Narayanan$^5$, Jaroslav Opatrny$^5$, Sunil Shende$^6$

$^1$Université du Québec en Outaouais, Gatineau, Québec, Canada
$^2$Department of Mathematics, Ryerson University, Toronto, Ontario, Canada
$^3$School of Computer Science, Carleton University, Ottawa, Ontario, Canada
$^4$Department of Mathematics & Comp. Sci., Wesleyan University, Middletown, CT, USA
$^5$Department of Comp. Sci. and Software Eng., Concordia University, Montreal, Québec, Canada
$^6$Department of Computer Science, Rutgers University, Camden, NJ, USA

Abstract

We introduce and study a new search-type problem with $(n + 1)$-robots on a disk. The searchers (robots) all start from the center of the disk, have unit speed, and can communicate wirelessly. The goal is for a distinguished robot (the queen) to reach and evacuate from an exit that is hidden on the perimeter of the disk in as little time as possible. The remaining $n$ robots (servants) are there to facilitate the queen’s objective and are not required to reach the hidden exit. We provide upper and lower bounds for the time required to evacuate the queen from a unit disk. Namely, we propose an algorithm specifying the trajectories of the robots which guarantees evacuation of the queen in time always better than $2 + 4(\sqrt{2} - 1)\pi n$ for $n \geq 4$ servants. We also demonstrate that for $n \geq 4$ servants the queen cannot be evacuated in time less than $2 + \frac{\pi}{n} + \frac{2}{n^2}$.

6.1 Introduction

A fundamental research topic in mathematics and computer science concerns search, whereby a group of mobile robots need to collectively explore an environment in order to find a hidden target. In the scenarios considered so far, the goal was to optimize the time when the first searcher reaches the target position. More recently, researchers studied the evacuation problem in which it is required to minimize the time of arrival to the target position of the last mobile robot in the group. In the work done on search so far, all robots are generally assumed to have exactly the same capabilities. However, it is quite natural to consider collaborative tasks in which the participant robots have different capabilities.
For example, robots may have different maximum speeds, or have different communication capabilities. Robots with different speeds have been studied in the context of rendezvous \cite{95} and evacuation \cite{137}. In the context of search, a natural situation may be that only one of the robots has the capability to address an urgent need at the target, for example, performing an emergency procedure, or closing a breach in the perimeter. The remaining robots can help in searching for the target, but their arrival at the target does not accomplish the main purpose of finding the target. Therefore, the collective goal of the robots is to get the special robot to the target as soon as possible. In this paper, we are interested in such a type of search problem, which grants priority to a pre-selected participant. In other words, we assume that the collection of robots contains a leader, known in advance, and as long as the leader does not get to the target position, search is considered incomplete.

In this paper we propose and investigate the priority evacuation problem, a new form of group search in which a given selected searcher in the group is deemed more important than the rest. This distinguished robot is given priority over all other searchers during the evacuation process in that it should be evacuated as early as possible upon the exit being located by any searcher.

6.1.1 Model

In the priority evacuation, or PE$_n$ problem, $n+1$ robots (searchers) are placed at the center of a unit disk. There is a target (exit), placed at an unknown location on the boundary of the disk. The target can be discovered by any robot walking over it. A robot that finds the exit instantaneously broadcasts its current position. Among the robots there is a distinguished one called the queen and the remaining $n$ robots are referred to as servants. The goal is to minimize the queen’s evacuation time, i.e. the worst case total time until the queen reaches the target. We assume that all robots, including the queen, may walk using maximum unit speed. We note that the queen may or may not actively participate in the search of the exit.

6.1.2 Related work

Search and exploration have been extensively studied in mathematics and various fields of computer science. If the environment is not known in advance, search implies exploration, and it usually involves mapping and localizing searchers within the environment \cite{3,85,118,143}. However, even for the case of a known, simple domain like a line, there have been several interesting studies attempting to optimize the search time. These were initiated with the seminal works of Bellman \cite{25} and Beck \cite{22}, in which the authors attempted to minimize the competitive ratio in a stochastic setting. After the appearance of \cite{13}, where a search by a single robot was studied for infinite lines and planes, several other works on linear search followed (cf. \cite{8}) and more recently the search by a single searcher was studied for different models, e.g., when the turn cost was considered \cite{84}, when a bound on the
distance to the target is known in advance [32], and when the target is moving or for more general linear cost functions [31].

For the case of a collection of searchers, numerous scenarios have been studied, such as: graph or geometric terrains, known or unknown environments, stationary or mobile targets, etc. (cf. [103]). In many papers, the objective is to decide the feasibility of the search or to minimize its search time.

The evacuation problem from the disk was introduced in [57] where two types of robots’ communication were studied – the wireless one and communication by contact (also called face-to-face). The bounds for evacuation of two robots communicating face-to-face were later improved in [73] and in [36]. Recently, Chuangpishit et al. considered worst-case average-case tradeoffs for the same problem, see [43]. The case of a disk environment with more than one exit was considered in [55] and [144]. Other variations included evacuation from environments such as regular triangles and squares [79], the case of two robots having different maximal speeds [137], and the evacuation problem when one of the robots is crash or byzantine faulty [59].

Group search and evacuation in the line environment were studied in [17, 40]. The authors of [40] proved, somewhat surprisingly, that having many robots using maximal speed 1 does not reduce the optimal search time as compared to the search using only a single robot. However, interestingly, [40] shows that the same bound for group search (and evacuation) is achieved for two robots having speeds 1 and 1/3. For both types of robots’ communication scenarios, [17] presents optimal evacuation algorithms for two robots having arbitrary, possibly distinct, maximal speeds in the line environment.

A priority evacuation-type problem has been previously considered in [112, 111] but with different terminology. Using the jargon of the current paper, an immobile queen is hidden somewhere on the unit disk, and a number of robots try to locate her, and fetch (evacuate) her to an exit which is also hidden. The performance of the evacuation algorithm is measured by the time it takes the queen to reach the exit. Apart from these results, and to the best of our knowledge nothing is known about the priority evacuation problem. In this work we provide a general strategy for the case of \( n \geq 4 \) servants. When there are fewer than 4 servants more ad hoc strategies must be employed which do not fit with the general framework developed here and they are therefore treated elsewhere [63].

### 6.1.3 Results of the paper

Section 6.2 introduces nomenclature and notation and discusses preliminaries. In Section 6.3 we provide an algorithm that evacuates the queen in time always smaller than \( 2 + 4(\sqrt{2} - 1)\frac{\pi}{n} \) for \( n \geq 4 \) servants (the exact evacuation times of our algorithm must be calculated numerically). In Section 6.4 we demonstrate that for \( n \geq 4 \) servants the queen cannot be evacuated in time less than \( 1 + \frac{2}{n} \cdot \arccos\left(-\frac{2}{n}\right) + \sqrt{1 - \frac{4}{n^2}} \), or, asymptotically, \( 2 + \frac{\pi}{n} + \frac{2}{n^2} \).

These results improve upon naive upper and lower bounds of \( 2 + \frac{2\pi}{n} \) and \( 2 + \frac{\pi}{n+1} \) respectively...
(see Section 6.2.2 and 6.4). A summary of the evacuation times for our algorithm (numerical results) as well as the upper and lower bounds (non-trivial and naive) is provided in Table 6.1 and in Figure 6.1. We conclude the paper in Section 6.5 with a discussion of open problems.

Table 6.1: Evacuation times $T$ of the queen using Algorithm 2 (numerical results). The upper bound of $2 + 4(\sqrt{2} - 1)\frac{\pi}{n}$ (Theorem 6.3.1), and the lower bound of $1 + \frac{2}{n} \cos^{-1} \left( \frac{-2}{n} \right) + \sqrt{1 - \frac{4}{n^2}}$ (Theorem 6.4.1) are also provided. For comparison, the naive upper bound and lower bound of $2 + \frac{2\pi}{n}$ (see Section 6.2.2) and $2 + \frac{\pi}{n+1}$ (see Section 6.4) are included.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$T$ (Alg 2)</th>
<th>UB (Thm 6.3.1)</th>
<th>LB (Thm 6.4.1)</th>
<th>UB Naive</th>
<th>LB Naive</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3.113</td>
<td>3.301</td>
<td>2.913</td>
<td>3.571</td>
<td>2.628</td>
</tr>
<tr>
<td>5</td>
<td>2.905</td>
<td>3.041</td>
<td>2.709</td>
<td>3.257</td>
<td>2.524</td>
</tr>
<tr>
<td>6</td>
<td>2.762</td>
<td>2.868</td>
<td>2.580</td>
<td>3.047</td>
<td>2.449</td>
</tr>
<tr>
<td>7</td>
<td>2.660</td>
<td>2.744</td>
<td>2.490</td>
<td>2.898</td>
<td>2.393</td>
</tr>
<tr>
<td>8</td>
<td>2.582</td>
<td>2.651</td>
<td>2.424</td>
<td>2.785</td>
<td>2.349</td>
</tr>
</tbody>
</table>

Figure 6.1: Evacuation times $T$ of Algorithm 2 for $n \in [4, 2^7]$ (left) and $n \in [4, 2^{18}]$ (right). The upper bound of $2 + 4(\sqrt{2} - 1)\frac{\pi}{n}$ (Theorem 6.3.1), the lower bound of $1 + \frac{2}{n} \cos^{-1} \left( \frac{-2}{n} \right) + \sqrt{1 - \frac{4}{n^2}}$ (Theorem 6.4.1) are also provided. For comparison, a naive upper bound and lower bound of $2 + \frac{2\pi}{n}$ (see Section 6.2.2) and $2 + \frac{\pi}{n+1}$ (see Section 6.4) are included.
6.2 Notation and Preliminaries

6.2.1 Notation

We denote by $\mathcal{U}$ the unit circle in $\mathbb{R}^2$ centered at the origin $O = (0, 0)$ which must be evacuated by the queen and we assume that all robots start from the origin. We use $n$ to denote the number of servants, and use $Q(t)$ and $S_k(t)$, $k = 1, \ldots, n$, to represent the trajectories of the queen and $k^{th}$ servant respectively. The set of all servant trajectories is represented by $\mathcal{S} = \{S_k(t); \ k = 1, \ldots, n\}$. A trajectory will be given as a parametric function of time and, when referring to a robot’s trajectory, it will be implied that we mean the path taken by the robot in the case that the exit has not been found.

6.2.2 Evacuation algorithms

A priority evacuation algorithm $\mathcal{A}$ is specified by the trajectories of the queen and servants, $\mathcal{A} = \{Q(t)\} \cup \mathcal{S}$. We say that $\mathcal{A}$ solves the PE$_n$ problem if, in finite time, all points of $\mathcal{U}$ are visited/discovered by at least one robot. The evacuation time $T$ of an algorithm solving the PE$_n$ problem is defined to be the worst-case time taken for the queen to reach the exit. As such, the evacuation time will be composed of two parts: the time taken until the exit is discovered plus the time needed for the queen to reach the exit once it has been found.

We will find it useful to define the restricted class of evacuation algorithms $\mathbb{S}$ containing all those algorithms in which: a) the queen does not participate in searching for the exit, b) the servants initially move as quickly as they can to the perimeter of $\mathcal{U}$, c) each servant searches either counter-clockwise or clockwise along the perimeter of $\mathcal{U}$ at full speed, and, d) each servant stops and is no longer used once it reaches an already discovered point of $\mathcal{U}$. Algorithms in this class can be defined by the trajectory of the queen $Q(t)$ together with the sequences $\Phi = [\phi_k \in [0, 2\pi]; \ k = 1, \ldots, n]$ and $\Sigma = [\sigma_k = \pm1; \ k = 1, \ldots, n]$ which respectively specify the angular positions on $\mathcal{U}$ to which the servants initially move, and the directions in which each servant searches. We will assume without loss of generality that $\Phi$ is specified such that $\phi_k \leq \phi_{k+1}$. With this notation we can express the trajectory of the $k^{th}$ servant during the time it is searching as $S_k(t) = (\cos(\phi_k + \sigma_k(t-1)), \sin(\phi_k + \sigma_k(t-1)))$.

We additionally define the class of algorithms $\mathbb{S}_{\text{sym}} \subset \mathbb{S}$ containing those algorithms for which we can split the set of servants into two groups $\mathcal{S} = \mathcal{S}_+ \cup \mathcal{S}_-$ where: a) servants in $\mathcal{S}_+$ follow trajectories which are reflections about the $x$-axis of servants in $\mathcal{S}_-$, and, b) all servants in $\mathcal{S}_+$ search counter-clockwise. In the case that $n$ is odd we permit one servant to follow a trajectory that is symmetric about the $x$-axis. For an algorithm

---

1. The choice of the $x$-axis is arbitrary since we may always rotate $\mathcal{U}$. What is important is that a diameter of symmetry exists.
2. Again, these choices of search directions are arbitrary since we can reflect $\mathcal{U}$ about the $y$-axis. What is important is that all servants within a group search in the same direction.
in $\mathbb{S}_{sym}$ we may write $\Phi = \Phi_+ \cup \Phi_-$ where $\Phi_+$ (resp. $\Phi_-$) specifies the positions on $\mathcal{U}$ to which the servants above (resp. below) the $x$-axis initially move. Formally we may write $\Phi_+ = \left[ \phi_k \in [0, \pi]; k = 1, \ldots, \left\lceil \frac{n}{2} \right\rceil \right]$ and $\Phi_- = -\Phi_+$ for even $n$ and $\Phi_- = \left[ -\phi_k; k = 2, \ldots, \left\lceil \frac{n}{2} \right\rceil \right]$ for odd $n$. In the class $\mathbb{S}_{sym}$ the directions in which the servants search are always counter-clockwise (resp. clockwise) for robots in $\Phi_+$ (resp. $\Phi_-$) and thus an algorithm $\mathcal{A} \in \mathbb{S}_{sym}$ is entirely specified by the set $\{Q(t)\} \cup \Phi_+$.

As a warm-up to the next section, and to demonstrate the intuitive nature of these definitions, consider the following trivial algorithm which achieves an evacuation time of $2 + \frac{2\pi}{n}$: the queen remains at the origin until the exit is found and the servants move directly to equally spaced locations on the perimeter of $\mathcal{U}$ each searching an arc of length $\frac{2\pi}{n}$ in the counter-clockwise direction. This algorithm can be seen to be in the class $\mathbb{S}$ and we can succinctly represent the algorithm as follows

**Algorithm 1 Trivial Evacuation 1, $\mathcal{A} \in \mathbb{S}$**

1. $Q(t) = (0, 0)$.
2. $\Phi = \{ \frac{(k-1)}{n}2\pi; k = 1, \ldots, n \}$
3. $\Sigma = \{1; k = 1, \ldots, n\}$

Observe that the above algorithm is not in $\mathbb{S}_{sym}$. We can, however, give an equivalent algorithm in $\mathbb{S}_{sym}$ which achieves the same evacuation time. This algorithm is depicted in Figure 6.2 along with Algorithm 1 for the case that $n = 8$.

![Figure 6.2: Depiction of the two trivial algorithms each achieving an evacuation time of $2 + \frac{2\pi}{n}$. Both algorithms are in the class $\mathbb{S}$ and the algorithm on the right is also in the class $\mathbb{S}_{sym}$. The queen is indicated by the blue point and the servants by the red points. A red arc indicates points that have been discovered.](image)
6.3 Upper Bound

In the previous section we introduced two evacuation algorithms solving $\text{PE}_n$ with evacuation time $2 + \frac{2\pi}{n}$. We will show that this can be improved:

**Theorem 6.3.1.** There exists an algorithm solving $\text{PE}_n$ for $n \geq 4$ with an evacuation time at most $2 + 4(\sqrt{2} - 1)\frac{\pi}{n} \approx 2 + 1.657\frac{\pi}{n}$.

We will prove Theorem 6.3.1 constructively and present an evacuation algorithm in the class $\mathbb{S}_{\text{sym}}$ achieving the desired upper bound for $n \geq 4$ servants. For ease of presentation we will assume that $n$ is even. Furthermore, as it will greatly simplify the algebra, we will redefine all times (including the evacuation time) to start from the moment the servants first reach the perimeter. To avoid confusion we will use $T_p$ to represent the evacuation time of an algorithm as measured from the moment the servants reach the perimeter. The total evacuation time will thus be $T = T_p + 1$.

As we will describe an algorithm in the class $\mathbb{S}_{\text{sym}}$ we will only need to specify the queen’s trajectory $Q(t)$ and the initial angular positions $\Phi_+$ of the servants lying above the $x$-axis. We start by giving the trajectory for the queen which we parametrize using $\alpha > 0$:

$$Q(t) = \begin{cases} (0, 0), & 0 \leq t < \alpha \\ (\alpha - t, 0), & \alpha \leq t < \alpha + 1 \\ (-1, 0), & t \geq \alpha + 1, \end{cases}$$  

(6.1)

In words, the queen waits at the origin until the time $t = \alpha$ at which moment she begins moving at full speed along the negative $x$-axis stopping when she arrives to the point $(-1, 0)$ at the time $t = \alpha + 1$. The crux of the algorithm will be in specifying the sequence $\Phi_+$. In order to do this we consider the following simple observation:

**Observation 6.3.2.** If the queen is to achieve an evacuation time of $T_p$, then, for all $t < T_p$, all of the undiscovered points of $U$ must remain inside the disk centered on the queen with radius $T_p - t$.

Assume that we have an algorithm with evacuation time $T_p$ and define $C_Q(t)$ as the circle centered on the queen with radius $T_p - t$. Then, in light of Observation 6.3.2, it is not so hard to imagine that the intersection points of the circles $C_Q(t)$ and $U$ will be of importance. Thus, assume that $T_p$ is small enough that at some time $t \geq \alpha$ the circles $C_Q(t)$ and $U$ intersect. Considering the form of the queen’s trajectory, we can conclude that the circles $U$ and $C_Q(t)$ will first intersect at the time $\gamma = \frac{T_p + \alpha - 1}{2}$ at the point $(1, 0)$. For times $t > \gamma$ the circles will intersect at two points $A_\pm$ which are symmetric about the $x$-axis and which move from right to left along the perimeter of $U$ (see Figure 6.3). The importance of the points $A_\pm$ is clear when one considers that $A_\pm$ mark the boundary between those points of $U$ which must be discovered and those which may yet be undiscovered at the time $t$. Intuitively, we will want
to position the servants such that they are searching only when they are to the left of $A_+\pm A_-$. In particular, a servant will stop searching at precisely the moment the intercept $A_+\pm A_-$ catches up to it (with a small caveat to be described shortly). This condition will allow us to specify the sequence $\Phi_+$. 

At this time we will find it useful to re-express the evacuation time as $T_p = 1 + \alpha + \rho$ where $\rho$ is a parameter that will ultimately depend on $\alpha$. Intuitively, $\rho$ represents the radius of $C_Q(t)$ at the moment the queen reaches the perimeter of $\mathcal{U}$ and its inclusion will greatly simplify algebra. Note that, with this definition, the circles $C_Q(t)$ and $\mathcal{U}$ will first intersect at the time $\gamma = \alpha + \frac{\rho}{2}$.

As we only need to specify the sequence $\Phi_+$ we will only consider the intercept $A_+$. The coordinates of $A_+$ for times $\gamma \leq t \leq \alpha + 1$ can be determined by simultaneously solving the implicit equations for $\mathcal{U}$ and $C_Q(t)$, i.e. $\mathcal{U} : x^2 + y^2 = 1$ and $C_Q(t) : (x - \alpha + t)^2 + y^2 = (1 + \alpha + \rho - t)^2$. We find that $A_+(t) = (x_A(t), y_A(t))$ where

$$x_A(t) = \frac{\rho(2 + \rho)}{2(t - \alpha)} - 1 - \rho$$  \hspace{1cm} (6.2)

and

$$y_A(t) = \frac{\sqrt{\rho(\rho + 2)[2(t - \alpha) - \rho][\rho + 2 - 2(t - \alpha)]}}{2(t - \alpha)}$$  \hspace{1cm} (6.3)

The angular position of $A_+$ will be represented as $\phi_A$ and is given by:

$$\phi_A(t) = \tan^{-1}\left(\frac{y_A(t)}{x_A(t)}\right).$$  \hspace{1cm} (6.4)

We define $\nu_A$ as the speed at which $A_+$ moves along the perimeter of $\mathcal{U}$. We can determine
\[ \nu_A \text{ using } \nu_A(t) = \sqrt{\left(\frac{dx_A}{dt}\right)^2 + \left(\frac{dy_A}{dt}\right)^2} \text{ from which we find that:} \]

\[ \nu_A(t) = \frac{1}{t - \alpha} \sqrt{\frac{\rho(\rho + 2)}{[\rho + 2 - 2(t - \alpha)][2(t - \alpha) - \rho]}} \quad (6.5) \]

Consider \( \nu_A(t) \). For times just after \( t = \alpha \) we can see that \( A_+ \) will move with a speed \( \nu_A >> 1 \) and, as such, no single servant will be able to stay to the left of \( A_+ \) for long. What is not so obvious from (6.5) is that \( \nu_A \) continuously decreases until some time \( \tau \) at which \( \nu_A = 1 \). Furthermore, starting at the time \( \tau \) there will be an interval of time during which \( \nu_A \leq 1 \). Thus, if the intercept reaches a servant at exactly the time \( \tau \) that servant does not have to stop searching. We will choose \( \rho \) to ensure that the servant \( S_{n/2} \in S_+ \) satisfies exactly this property.

We can describe the following general overview of our algorithm: the servant \( S_1 \) begins at \( \phi_1 = 0 \) (for even \( n \)) and searches until the time \( t_1 \) at which \( S_1(t_1) = A_+(t_1) \) or when \( t_1 + \phi_1 = \phi_A(t_1) \). The servant \( S_2 \) will begin its search at the position \( \phi_2 = \phi_1 + t_1 \) and it will search for a time \( t_2 \) until \( S_2(t_2) = A_+(t_2) \) or until \( t_2 + \phi_2 = \phi_A(t_2) \). The servant \( S_3 \) will begin at the position \( \phi_3 = \phi_2 + t_2 = \phi_1 + t_1 + t_2 \), and so on. Continuing on like this we can see that the servant \( S_{k+1} \) will begin its search at the position \( \phi_{k+1} = \phi_k + t_k = \phi_1 + \sum_{i=1}^{k} t_i \) with the \( t_k \) satisfying \( t_k = \phi_A(t_k) - \phi_k \) or, equivalently, \( \phi_1 + \sum_{i=1}^{k} t_i = \phi_A(t_k) \). We want the servant \( S_{n/2} \) to be coincident with the intercept \( A_+ \) at exactly the time \( \tau \) (recall that \( \tau \) is the time at which the speed of \( A_+ \) is \( \nu_A = 1 \)) and thus we will choose \( \rho \) to satisfy \( \phi_{n/2} + \tau = \phi_A(\tau) \). In this case the servant \( S_{n/2} \) will search for a total time \( \pi - \phi_{n/2} \) after which all of \( U \) will have been discovered.

To extend this algorithm to the case that \( n \) is odd we will need to split the trajectory of the servant \( S_1 \in S_+ \) between the upper and lower halves of \( U \). We will therefore start the servant \( S_1 \) at the position \( \phi_1 = \frac{-t_1}{2} \). All of the other relevant equations remain unchanged.

We provide links ([162] and [163]) to short animations of the algorithm for \( n = 4, 8 \). In these animations the queen is represented by the blue point, the servants by red points, and the intercepts \( A_\pm \) by green points. A plot of the evacuation time as a function of the time at which the servants find the exit is also shown. Note that the servants stop searching at the exact moment the intercept reaches them (except for the two servants furthest to the left) and at these moments the evacuation time is maximized. The two servants that are last active will be coincident with the intercepts at the moment these intercepts reach a speed of one, and, again, at this moment the evacuation time is maximized. In total there will be \( n \) different locations for the exit (counting the top and bottom of \( U \)) which will maximize the evacuation time. A keen eye will note that the queen reaches the perimeter of \( U \) before the servants have finished searching the perimeter and this would appear to hint that Algorithm 2 can be improved. We will argue in Section 6.5 that this is not the case.

\footnote{It is not guaranteed that for all \( \rho > 0 \) this intercept will reach a speed of one before the queen reaches the perimeter of \( U \). However, we will choose a \( \rho \) such that this does happen.}
Figure 6.4 illustrates an example configuration for the described algorithm when $n = 8$. The algorithm is formally presented in Algorithm 2 where we have left $\alpha$ as a parameter. We claim that Algorithm 2 will always do better than the bound of Theorem 6.3.1 when the evacuation time is minimized over $\alpha$. We will now prove this claim.

**Algorithm 2** IntersectChase($\alpha$), $\mathcal{A}_\alpha \in \mathcal{S}_{sym}$

1: $Q(t) = \begin{cases} (0, 0), & 0 \leq t < \alpha \\ (\alpha - t, 0), & \alpha \leq t < \alpha + 1 \\ (-1, 0), & t \geq \alpha + 1, \end{cases}$

2: $\Phi_+ = \{\phi_k; k = 1, \ldots, \lceil \frac{n}{2} \rceil\}$, where:

\[
\begin{align*}
\phi_1 &= \begin{cases} 0, & n \text{ even} \\ -\frac{t_1}{2}, & n \text{ odd} \end{cases}, \\
\phi_k &= \phi_1 + \sum_{i=1}^{k-1} t_i, \quad \phi_{n/2} + \tau = \phi_A(\tau) \\
\end{align*}
\]

and,

\[
\phi_1 + \sum_{i=1}^{k} t_i = \phi_A(t_k), \quad \nu_A(\tau) = 1
\]

Figure 6.4: Example configuration of Algorithm 2 when $n = 8$. The configuration is only shown for the 4 servants on the upper half of the circle $\mathcal{U}$. In this diagram all servants move counter-clockwise. The servants $S_1$ and $S_2$ have already finished their search and are located at the starting positions of the respective servants $S_2$ and $S_3$. The servant $S_3$ is just about to finish its search. The point $A_+(\tau)$ marks the location where the intercept $A_+(t)$ slows to a speed of 1. The servant $S_4$ will reach the point $A_+(\tau)$ at the exact moment the intercept does.

181
Proof. (Theorem 6.3.1) To simplify the algebra we will assume that \( n \) is even. Algorithm 2 specifies that we choose the \( t_i \) in order to satisfy \( \sum_{i=1}^{k} t_i = \phi_A(t_k) \) where \( \phi_A(t) \) is defined in (6.4). We note that each servant will be able to search for at least a time \( \gamma \) since this marks the first time at which \( C_Q(t) \) and \( \mathcal{U} \) intersect. This motivates us to define the primed time coordinate \( t' = t - \gamma \). In this primed coordinate the defining relation for the \( t'_k \) is \( \sum_{i=1}^{k} t'_k = \phi_A(t'_k) - k\gamma \) (where we assume that \( \phi_A \) is properly redefined for the primed time coordinate). We are interested in an asymptotic limit and thus we make the following claim which is based on the standard definition of Riemann integration:

Claim 6.3.3. Let \( f = \frac{k}{n} \). Then, holding \( f \) constant and taking the limit in large \( n \), the sum \( \sum_{i=1}^{k} t'_i \) becomes a definite integral \( \lim_{n \to \infty} \sum_{i=1}^{k} t'_i = \int_{0}^{\kappa} t'(u) du \) where \( \frac{n}{\kappa} = f \) is to be interpreted as the fractional servant number and \( u \) is a dummy integral variable.

Proof. Consider the sum \( \sum_{i=1}^{k} t'_i \). Write \( t'_k = t'(k) \), define \( f_k = \frac{k}{n} \) as the fractional servant number, and let \( \Delta f = \Delta f_k = \frac{1}{n} \). Finally, defining \( t'(0) = 0 \) allows us to rewrite the sum as \( \sum_{i=1}^{k} t'_i = n \sum_{i=0}^{k} t'(n f_i) \Delta f_i \). Now, for a given constant fraction \( 0 \leq f_k \leq \frac{1}{2} \) the bounds of the sum \( \sum_{i=0}^{k} t'(n f_i) \Delta f_i \) are constant. Furthermore, in the limit \( n \to \infty \) the interval \( \Delta f_i \to 0 \). Thus, the limit, \( \lim_{n \to \infty} \sum_{i=0}^{k} t'(n f_i) \Delta f_i \) is simply the definition of the Riemann integral of \( t'(n f) \) over the domain \( f \in [0, f_k] \), i.e., \( \lim_{n \to \infty} \sum_{i=0}^{k} t'(n f_i) \Delta f_i = \int_{0}^{\kappa} t'(u) du \).

By defining \( \kappa = n f_k \), we get \( \int_{0}^{\kappa} t'(u) du = \frac{1}{n} \int_{0}^{\kappa} t'(u) du \) and this leads us to our desired result \( \lim_{n \to \infty} \sum_{i=1}^{k} t'_i = \int_{0}^{\kappa} t'(u) du \). \( \square \)

Due to the Claim 6.3.3 the asymptotic defining relation for \( t'(\kappa) \) becomes an integral equation \( \int_{0}^{\kappa} t'(u) du = \phi_A(t'(\kappa)) - \kappa \gamma \). Using the fundamental theorem of calculus we can rewrite this as a differential equation: \( t'(\kappa) = \frac{d}{d\kappa} (\phi_A(t'(\kappa)) - \kappa \gamma) = \frac{d\phi_A(t'(\kappa))}{d\kappa} - \gamma \). Applying the chain rule we find that \( \frac{d\phi_A(t'(\kappa))}{d\kappa} = \frac{d\phi_A(t'(\kappa))}{dt'} \cdot \frac{dt'}{d\kappa} \). Observe that \( \frac{dt'}{d\kappa} = \frac{t'+\gamma}{\nu_A(t'(\kappa))} \). This ordinary differential equation can easily be solved for \( \kappa \) in terms of \( t' \) by separation of variables. We find that \( \kappa(t') = \int_{0}^{t'} \frac{\nu_A(u)}{u+\gamma} du \). The equation for the speed \( \nu_A \) is given in (6.5), which, in the primed time coordinate takes the form \( \nu_A(t') = \frac{1}{\left(2u+\rho\right)^2} \sqrt{\frac{\rho(\rho+2)}{u(1-u)}} \). Substituting this into the expression for \( \kappa(t') \) yields \( \kappa(t') = \int_{0}^{t'} \frac{1}{\left(2u+\rho\right)^2} \sqrt{\frac{\rho(\rho+2)}{u(1-u)}} \). This integral has the closed form solution

\[
\kappa(t') = \frac{1}{\alpha} \left[ 2 \tan^{-1} \left( \frac{t'(2t'+\rho)}{\rho} \nu_A(t') \right) \right. \\
- \left. \sqrt{\frac{\rho(\rho+2)}{\gamma(\gamma+1)}} \tan^{-1} \left( \frac{t'(2t'+\rho)}{\sqrt{\frac{\gamma(\gamma+1)}{\gamma(\rho+2)}}} \nu_A(t') \right) \right].
\]

182
We require that the servant \( S_{n/2} \) be coincident with the intercept \( A_+ \) at the time \( \tau' = \tau - \gamma \) and this implies that we need \( \kappa(\tau') = \frac{n}{2} \) or

\[
\frac{n}{2} = \frac{1}{\alpha} \left[ 2 \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \nu_A(\tau') \right) - \frac{\rho(\rho + 2)}{\gamma(\gamma + 1)} \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \frac{\rho(\gamma + 1)}{\gamma(\rho + 2)} \right) \right].
\]

If we set \( \alpha = a \frac{\pi}{n} \) and note that, by definition, \( \nu_A(\tau') = 1 \), we can simplify the above to obtain

\[
\frac{\pi}{2} = \frac{1}{a} \left[ 2 \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \right) - \frac{\rho(\rho + 2)}{\gamma(\gamma + 1)} \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \frac{\rho(\gamma + 1)}{\gamma(\rho + 2)} \right) \right].
\]

Define \( D(a, \rho) \) as the quantity

\[
D(a, \rho) = \frac{\pi}{2} - \frac{1}{a} \left[ 2 \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \right) - \frac{\rho(\rho + 2)}{\gamma(\gamma + 1)} \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \frac{\rho(\gamma + 1)}{\gamma(\rho + 2)} \right) \right]
\]

which we want to be zero. We now make the following claim:

**Claim 6.3.4.** The asymptotic behaviour of \( \tau \) is \( O \left( \rho^{1/3} \right) \).

**Proof.** We want to show that \( \tau \) has asymptotic behaviour \( O \left( \rho^{1/3} \right) \). One way that we can do this is by formally computing a Puiseux series of \( \tau \) from which one would find that the first few terms in the expansion are \( \tau = \left( \frac{\rho}{2} \right)^{1/3} + \frac{1}{2} \left( \frac{\rho}{2} \right)^{2/3} + O \left( \rho \right) \). Alternatively, we note that \( \tau' \) solves the equation \( \nu_A(\tau') = 1 \), i.e. \( \frac{1}{2t' + \rho} \sqrt{\frac{\rho(\rho + 2)}{t'(1-t')}} = 1 \). Expanding the above we arrive at the quartic equation \( t'^4 - (\rho + 1)t'^3 + \frac{\rho(\rho + 2)}{4}(t'^2 + 1) = 0 \) where \( t'' = t' + \frac{\rho}{2} = t - \alpha \). As \( n \to \infty \) we have \( 1 + t'^2 \to 1, 2 + \rho \to 2 \) and \( t'^4 \to 0 \). As \( n \) gets large \( \tau \) approaches the solution to \(-t'^3 + \frac{\rho}{2} = 0 \) which also demonstrates that \( \tau = O \left( \rho^{1/3} \right) \). \( \Box \)

Using Claim 6.3.4 we have that \( \lim_{n \to \infty} \frac{\tau'(2\tau' + \rho)}{\rho} = \lim_{n \to \infty} O \left( \rho^{-1/3} \right) = \infty \) and thus

\[
\frac{\pi}{2} = \lim_{n \to \infty} \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \right) = \lim_{n \to \infty} \tan^{-1} \left( \frac{\tau'(2\tau' + \rho)}{\rho} \frac{\rho(\gamma + 1)}{\gamma(\rho + 2)} \right).
\]

183
We can therefore write \( \lim_{n \to \infty} D(a, \rho) = \frac{\pi}{a} \left( 1 - \frac{a}{2} - \sqrt{\frac{a}{2\pi}} \right) \). Now set \( \rho = q_n^{\pi} \) such that \( \gamma = \alpha + \frac{\rho}{2} = \frac{\pi}{a} (a + \frac{q}{2}) \). Using this notation we have \( \lim_{n \to \infty} D(a, \rho) = \frac{\pi}{a} \left( 1 - \frac{a}{2} - \sqrt{\frac{a}{2a+q}} \right) \). We want this limit to equal zero which implies that we need \( 1 - \frac{a}{2} - \sqrt{\frac{q}{2a+q}} = 0 \) or \( q = \frac{2(2-a)^2}{a(1-a)} \).

Now, to optimize the algorithm we need to minimize the evacuation time \( T_p \). Since \( T_p \) increases with \( a \) we equivalently need to minimize \( a + q = \frac{a^2 - 4a + 8}{4-a} \). Taking the derivative of this with respect to \( a \) and setting the result equal to zero gives us the optimal value of \( a \) and \( q \) to be \( a = 2(2 - \sqrt{2}) \) and \( q = 2(3\sqrt{2} - 4) \). The asymptotic cost of the algorithm is therefore \( T_p = 1 + \alpha + \rho = 1 + 4(\sqrt{2} - 1) \frac{\pi}{n} \). The overall evacuation time is then \( T = 1 + T_p \) which is the bound given in Theorem 6.3.1.

We note that, in the case that \( n \) is odd, the results of the proof will not change due to the fact that, as \( n \to \infty \), we have \( \phi_1 = -\frac{1}{n} \to 0 \).

### 6.4 Lower Bound

In this section we develop a lower bound on the evacuation time of the queen. We first note that we can derive a naive lower bound of \( 2 + \frac{\pi}{n+1} \) since each robot can travel with a maximum speed of one and we have \( n + 1 \) robots in total. We will show that this can be improved:

**Theorem 6.4.1.** In any algorithm with \( n \geq 4 \) the queen cannot be evacuated in time less than \( 1 + 2 \frac{\cos^{-1} \left( \frac{2}{n} \right)}{n} + \sqrt{1 - \frac{4}{n^2}} \). In the limit of large \( n \) this bound approaches \( 2 + \frac{\pi}{n} + \frac{2}{n^2} \).

The outline of the proof is as follows: we first demonstrate that the lower bound holds for any algorithm in which the queen does not participate in searching for the exit before some critical time. We will then show that the queen is not able to participate in the search for the exit before this critical time. We begin with a lemma first given in [57] which is reproduced here for convenience:

**Lemma 6.4.2.** Consider a perimeter of a disk whose subset of total length \( u + \epsilon > 0 \) has not been explored for some \( \epsilon > 0 \) and \( \pi \geq u > 0 \). Then there exist two unexplored boundary points between which the distance along the perimeter is at least \( u \).

In the next two lemmas we demonstrate that the lower bound holds if the queen does not participate in the search.

**Lemma 6.4.3.** For \( n \geq 2 \), any \( x \) satisfying \( \frac{\pi}{n} \leq x < \frac{2\pi}{n} \), and any evacuation algorithm in which the queen does not participate in searching for the exit before the time \( 1 + x \), it takes time at least \( 1 + x + \sin \left( \frac{n\pi}{2} \right) \) to evacuate the queen.
Proof. Consider an algorithm $A$ with evacuation time $T$ and with $n$ servants. Then, at the
time $t = 1 + x$, the total length of perimeter that the robots have explored is at most $n x \geq \pi$
(since each robot may search at a maximum speed of one, the queen does not search by
assumption, and the servants need at least a unit of time to reach the perimeter). Thus, by
Lemma 6.4.2 there exists two unexplored points on the perimeter of $\mathcal{U}$ whose distance along
the perimeter is at least $2 \pi - n x - \epsilon$ for any $\epsilon > 0$. The chord connecting these points has
length at least $2 \sin \left(\frac{\pi - n x}{2} - \frac{\epsilon}{2}\right)$ and an adversary may place the exit at either endpoint of
this chord. Since the queen is at least a distance $\sin \left(\frac{\pi - n x}{2} - \frac{\epsilon}{2}\right)$ from one of the endpoints
of this chord she will take at least $\sin \left(\frac{\pi - n x}{2} - \frac{\epsilon}{2}\right)$ more time to evacuate and the total evacuation time will be at least
$1 + x + \sin \left(\pi - \frac{n x}{2} - \frac{\epsilon}{2}\right)$. As this is true for any $\epsilon > 0$
taking the limit $\epsilon \to 0$ we obtain $T \geq 1 + x + \sin \left(\frac{\pi - n x}{2}\right) = 1 + x + \sin \left(\frac{n x}{2}\right)$. □

Lemma 6.4.4. For any $n \geq 2$ and any evacuation algorithm in which the queen does not
participate in searching for the exit before the time $t = 1 + \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right)$ it takes time at
least $1 + \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right) + \sqrt{1 - \frac{4}{n^2}}$ to evacuate the queen.

Proof. Set $f(x) = 1 + x + \sin \left(\frac{nx}{2}\right)$. The maximum value of $f(x)$ occurs when $\frac{df}{dx} = 0$
or when $x = \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right)$. Since $\frac{\pi}{n} \leq \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right) < \frac{\pi}{n}$ we can invoke Lemma 6.4.3 to
get a lower bound on the evacuation time of $T \geq 1 + \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right) + \sin \left(\cos^{-1} \left(\frac{2}{n}\right)\right) =
1 + \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right) + \sqrt{1 - \frac{4}{n^2}}$, provided that the queen does not search before the time
t $t = 1 + \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right)$. □

We will now demonstrate that the queen is not able to search before the time $1 + \frac{2}{n} \cos^{-1} \left(\frac{2}{n}\right)$. This will be the goal of the next four lemmas and the following simple observation

Observation 6.4.5. If the queen is to achieve an evacuation time of $T$, then, for any time
$t \leq T$, she must remain in the region of intersection of all disks centered on the undiscovered
points of $\mathcal{U}$ with radii $T - t$.

Lemma 6.4.6. Consider any two points $A$ and $B$ on the unit circle connected by a chord of
length $\delta$. Define the circles $C_A$ and $C_B$ as the circles centered on $A$ and $B$ with radii $r$. Then,
if $r > \frac{\delta}{2}$, the circles intersect at two points $C$ and $D$ at distances $\sqrt{r^2 - \frac{1}{4} \delta^2} \pm \sqrt{1 - \frac{4}{3} \delta^2}$ from the origin.

Proof. Assume that $r > \frac{1}{2} \delta$. Set $C$ and $D$ as the intersection points of $C_A$ and $C_B$, and set $E$
 as the midpoint of $A$ and $B$. Refer to Figure 6.5 for a setup of the proof.

Since $C_A$ and $C_B$ have the same radius the points $C$ and $D$ are each separately equidistant
to $A$ and $B$ and they will therefore lie on the perpendicular bisector of $A$ and $B$. Since
A and B lie on the unit circle this bisector will pass through the origin. Referring to Figure 6.5 it is therefore clear that $|EC| = |ED| = \sqrt{r^2 - \frac{1}{4}\delta^2}$ and $|OE| = \sqrt{1 - \frac{1}{4}\delta^2}$. Since $|OC| = |OE| + |EC|$ and $|OD| = |ED| - |OE|$ we get the desired result.

**Lemma 6.4.7.** For a given $r > 0$ define the functions \( f_\pm(x) = \frac{1}{2} \sqrt{4r^2 - x^2} \pm \frac{1}{2} \sqrt{4 - x^2} \). Then, for $0 \leq x \leq \min\{2, 2r\}$ \( f_+ \) is a decreasing function of \( x \) and \( f_- \) is an increasing function of \( x \) if \( r > 1 \) otherwise it is decreasing.

**Proof.** Assume that \( x \) is in the interval $0 \leq x \leq \min\{2, 2r\}$. The rate of change of \( f_\pm \) with \( x \) is \( \frac{df_\pm}{dx} = -\frac{x}{2} \left(\frac{1}{\sqrt{4r^2 - x^2}} \pm \frac{1}{\sqrt{4 - x^2}}\right) \). For \( x \) in the given interval it is clear from this expression that \( \frac{df_+}{dx} < 0 \). On the other hand, \( \frac{df_-}{dx} < 0 \) when \( \frac{1}{\sqrt{4r^2 - x^2}} - \frac{1}{\sqrt{4 - x^2}} > 0 \). It is not hard to see that this occurs when \( r < 1 \).

**Lemma 6.4.8.** Consider any $r > 0$ and assume that the unexplored subset of \( U \) has total length $\phi$. Define \( D_P \) as the disk centered on an undiscovered point \( P \in U \) with radius \( r \) and define \( G \) as the region of intersection of all such disks. Then, if $r \geq \sin \left( \frac{\phi}{2} \right)$, \( G \) is completely contained inside of a disk centered on the origin with radius \( R = \sqrt{r^2 - \sin^2 \left( \frac{\phi}{2} \right) + \cos \left( \frac{\phi}{2} \right)} \). If $r < \sin \left( \frac{\phi}{2} \right)$ then \( G = \emptyset \).

**Proof.** By Lemma 6.4.2 there exists two undiscovered points \( A, B \in U \) such that the length between them along the perimeter of \( U \) is at least \( \phi \). Take these two points and set \( \delta \geq 2 \sin \left( \frac{\phi}{2} \right) \) as the length of the chord connecting them. Since \( A \) and \( B \) are unexplored there must exist a corresponding point \( G \in G \) such that \( G \) is at most a distance \( r \) from both \( A \) and \( B \). Since the chord connecting \( A \) and \( B \) has length \( \delta \), \( G \) is at least a distance \( \frac{\delta}{2} \geq \sin \left( \frac{\phi}{2} \right) \) from one of \( A \) and \( B \). We can therefore conclude that in order for \( G \) to exist we must have $r \geq \sin \left( \frac{\phi}{2} \right)$. This proves the second part of the lemma.

---

\( |AB| \) represents the Euclidean distance between two points \( A \) and \( B \).
Take \( r \geq \sin \left( \frac{\phi}{2} \right) \) and define the functions \( f_\pm(x) \) as in Lemma 6.4.7. Observe that
\[
R = \begin{cases} 
  f_+ \left( 2 \sin \left( \frac{\phi}{2} \right) \right), & 0 \leq \phi \leq \pi \\
  f_- \left( 2 \sin \left( \frac{\phi}{2} \right) \right), & \pi \leq \phi \leq 2\pi.
\end{cases}
\]

We consider the cases \( \phi \leq \pi \) and \( \phi > \pi \) separately.

**Case 1:** Take \( \phi \leq \pi \) and assume that the lemma is false. In this case there must be a point on the boundary of \( G \) that is at a distance \( d_G > f_+(2 \sin \left( \frac{\phi}{2} \right)) \) from the origin. Let \( G \) be such a point. Since \( G \) is on the boundary of \( G \) there are two undiscovered points \( A, B \in U \) that are at a distance \( r \) from \( G \). The point \( G \) is therefore a point of intersection of two circles \( C_A \) and \( C_B \) centered on \( A \) and \( B \) with radii \( r \). If \( \delta \) is the length of the chord connecting \( A \) and \( B \) then by Lemma 6.4.6 the point \( G \) lies a distance \( f_+ (\delta) \) or \( f_- (\delta) \) from the origin. Since \( f_+ (x) \geq f_- (x) \) then \( G \) must in fact lie a distance \( f_+ (\delta) \) from the origin. By assumption \( d_G > f_+ \left( 2 \sin \left( \frac{\phi}{2} \right) \right) \) and thus \( f_+ (\delta) > f_+ \left( 2 \sin \left( \frac{\phi}{2} \right) \right) \). Since \( f_+ \) is a decreasing function we can further say that \( \delta < 2 \sin \left( \frac{\phi}{2} \right) \) and we can thus conclude that there cannot exist two undiscovered points \( A, B \in U \) a distance \( \delta \geq 2 \sin \left( \frac{\phi}{2} \right) \) from each other. However, Lemma 6.4.2 states there must exist two undiscovered points in \( U \) such that the chord joining them has length at least \( 2 \sin \left( \frac{\phi}{2} \right) \). We have arrived at a contradiction and must therefore accept that the lemma is valid in the case that \( \phi \leq \pi \).

**Case 2:** When \( \phi > \pi \) we only have to consider a subset of the unexplored points of \( U \) of length \( \pi \). We can then use the same argument of the previous case to conclude that the lemma must be valid in this case as well.

\[ \square \]

**Lemma 6.4.9.** Consider an algorithm with evacuation time \( T < 3 \). Then if the queen is able to search the perimeter of \( U \) we must have
\[
R(t) = \sqrt{(T - t)^2 - \sin^2 \left( \frac{n(t - 1)}{2} \right) - \cos \left( \frac{n(t - 1)}{2} \right)} > 1.
\]

**Proof.** By Observation 6.4.5 and Lemma 6.4.8 we can immediately conclude that the queen must be within a distance \( R(t) = \sqrt{(T - t)^2 - \sin^2 \left( \frac{\phi(t)}{2} \right) + \cos \left( \frac{\phi(t)}{2} \right)} \) of the origin at any time \( t \). Consider the time \( t = 1 \). At this time the robots have not been able to search any of the perimeter of \( U \) and thus \( \phi(1) = 2\pi \). At this time the queen can be a distance at most \( R(1) = T - 2 \) from the origin and, since we have assumed that \( T < T_0 < 3 \), we have \( R(1) < 1 \). Therefore the queen cannot be on the perimeter of \( U \) at the time \( t = 1 \).
We claim that this case can be ignored since the robots will need to search at a much higher rate if they are to achieve the lower bound of Theorem 6.4.1. Alternatively we can say that the robots must search at a collective rate \( \phi > \pi \). To have the perimeter at a rate of \( \mu \) we need \( \phi > \pi \). To have \( \phi \) increasing we needed \( T - t < 1 \) or \( t > T - 1 \) and thus we must have \( \pi > \mu(T - 1) \) or \( \mu < \frac{\pi}{T - 1} \). A trivial lower bound on \( T \) is 2 and thus \( \mu < \pi \). We claim that this case can be ignored since the robots will need to search at a much higher rate if they are to achieve the lower bound of Theorem 6.4.1.

In the second case both \( \cos \left( \frac{\phi}{2} \right) \geq 0 \) and \( \sin \left( \frac{\phi}{2} \right) \geq 0 \) for \( 0 \leq \phi \leq \pi \). Thus, it is not possible that \( R \) increases with \( \phi \).

We can conclude from the above analysis that \( R \) decreases with \( \phi \) in all reasonable cases and thus we maximize \( R \) when \( \phi(t) \) is minimized. Since the queen cannot be on the perimeter of \( U \) at the beginning of the algorithm the robots can search at most at a rate \( n \) and therefore \( \phi(t) \) is minimized when \( \phi(t) = 2\pi - n(t - 1) \). Thus, up until the time the queen reaches the perimeter of \( U \), the queen must be located within a distance

\[
R(t) = \sqrt{(T - t)^2 - \sin^2 \left( \frac{n(t-1)}{2} \right) - \cos \left( \frac{n(t-1)}{2} \right)} \]

of the origin. We can finally conclude that in order for the queen to search we must have \( R(t) = \sqrt{(T - t)^2 - \sin^2 \left( \frac{n(t-1)}{2} \right) - \cos \left( \frac{n(t-1)}{2} \right)} \geq 1. \)

\[\square\]

Armed with these lemmas we are now able to tackle our main result.

**Proof.** (Theorem 6.4.1) Set \( T_0 = 1 + \frac{2}{n} \cos^{-1} \left( \frac{2}{n} \right) + \sqrt{1 - \frac{4}{n^2}} \) and assume we have an algorithm with an evacuation time \( T < T_0 \). By Lemma 6.4.4 this implies that the queen must search the perimeter of \( U \) before the time \( t_c = 1 + \frac{2}{n} \cos^{-1} \left( \frac{2}{n} \right) \). Assume that at the time \( t_c \) the robots have collectively searched the perimeter of \( U \) at a rate \( \mu \) satisfying \( n < \mu \leq n + 1 \). Then at the time \( t_c \) the unexplored subset of \( U \) has length \( \phi(t) = 2\pi - \mu(t_c - 1) = 2\pi - 2\mu \cos^{-1} \left( \frac{2}{n} \right) < \pi \). Since \( \phi(t_c) \leq \pi \) we can use Lemma 6.4.8 to say that the queen must be located within a distance of \( R(t_c) \) of the origin at the time \( t_c \). Furthermore, in order for the queen to have searched the perimeter of \( U \) at the time \( t_c \), we

\[\text{Alternatively we can say that the robots must search at a collective rate} > n \text{ by the time} t_c. \text{This is why we were able to ignore the "unreasonable case" in Lemma 6.4.9.}\]
must have $R(t_c) \geq 1$. However, observe that

$$R(t_c) = \sqrt{(T - t_c)^2 - \sin^2 \left( \frac{n(t_c - 1)}{2} \right) - \cos \left( \frac{n(t_c - 1)}{2} \right)}$$

$$\leq \sqrt{(T_0 - t_c)^2 - \sin^2 \left( \frac{n(t_c - 1)}{2} \right) - \cos \left( \frac{n(t_c - 1)}{2} \right)}$$

$$= \sqrt{1 - \frac{4}{n^2} - \sin^2 \left( \cos^{-1} \left( \frac{-2}{n} \right) \right) - \cos \left( \cos^{-1} \left( \frac{-2}{n} \right) \right)}$$

$$= \frac{2}{n}$$

which is clearly less than one for $n \geq 4$. We have therefore arrived to a contradiction and must conclude that the lower bound holds.

To determine the asymptotic behaviour of $T_0$ we can compute a Taylor series of $T_0$ about $n = \infty$. We find that the first few terms in the series are $2 + \frac{4}{n} + 2 \frac{n}{n^2}$.

### 6.5 Conclusions

We studied an evacuation problem concerning priority search on the perimeter of a unit disk where only one robot (the queen) needs to exit from an unknown location. We focused on the case of $n \geq 4$ servants and showed in Section 6.3 that for any $n \geq 4$ the queen can be evacuated in time at most $2 + 4(\sqrt{2} - 1) \frac{n}{n}$. Furthermore, in Section 6.4, we demonstrated that the queen cannot be evacuated in time less than $1 + \frac{2}{n} \cos^{-1} \left( \frac{-2}{n} \right) + \sqrt{1 - \frac{4}{n^2}} > 2 + \frac{\pi}{n} + \frac{2}{n^2}$.

Thus, in the limit of large $n$, we are left with a gap of $(4\sqrt{2} - 5) \frac{n}{n} \approx 0.657 \frac{n}{n}$ between the best upper and lower bounds. We conjecture that Algorithm 2 is in fact optimal. We will now justify this conjecture.

We have already mentioned that in Algorithm 2, the queen is able to reach the perimeter of $U$ before the servants have finished their search. This would seem to indicate that it is possible to improve on our algorithm. We will now argue why we do not think this is true. Similar to the proof of Theorem 6.4.1, there are critical times ($\frac{n}{2}$ of them) that occur before the queen reaches the perimeter and anything she does after these critical times cannot improve the evacuation time. These critical times result from a tradeoff between maximizing the rate at which the servants search – which implies that the servants are spread out on the perimeter and thus the queen must be near the origin so as to minimize her distance from potential exits – and minimizing the distance of the queen from possible exits near the end of the algorithm – for which the queen would like to position herself closer to the perimeter. In order to achieve the best tradeoff, the queen should travel as fast as she can from the origin to the perimeter without getting too far away from possible exits found earlier in the algorithm. In other words, between these critical times, the queen should maximize her
radial velocity. If we could prove that the queen does not need to participate in searching then it would not be difficult to conclude why Algorithm 2 is optimal. Any other trajectory of the queen between the critical search times will result in the same or a reduced radial velocity of the queen. It therefore does not seem likely that, with a reduced radial velocity, we can reduce the evacuation time.

In addition to improving the bounds obtained in this paper there are several interesting open problems related to priority search and evacuation. In particular, we may define a weighted evacuation problem (for a given group of agents) as a generalization of the priority evacuation problem studied here. One can differentiate on agent preferences by assigning a weight $w_i$ to each agent $i$ and require to evacuate a subset of agents of total weight $\geq W$ in minimum time. With this formulation in mind, the regular evacuation problem (see [57]) is the case where $w_i = 1$ for all agents and $W = n$, while for the problem considered in this work $w_i = 0$ for all agents except the queen for which $w_{\text{queen}} = 1$ and $W = 1$.

References


Part II

Rendezvous and gathering
Chapter 7

Linear Rendezvous with Asymmetric Clocks

In this chapter we move away from the problems of search and begin investigating problems of rendezvous and gathering. In the first two chapters of this part of the thesis, the problem of symmetric rendezvous by two agents on the infinite line and plane is considered. In both cases, the novel idea is to make use of asymmetries relating to the capabilities of the robots in order to achieve the rendezvous – even when the robots do not know about these asymmetries. In this chapter the problem on the infinite line is considered and in the next chapter the same problem is considered on the infinite plane. The paper “Linear Rendezvous with Asymmetric Clocks” included here is a conference publication appearing in the proceedings of OPODIS2018.
Linear Rendezvous with Asymmetric Clocks

Jurek Czyzowicz$^1$, Ryan Killick$^2$, Evangelos Kranakis$^2$

$^1$Université du Québec en Outaouais, Gatineau, Québec, Canada
$^2$School of Computer Science, Carleton University, Ottawa, Ontario, Canada

Abstract

Two anonymous robots placed at different positions on an infinite line need to rendezvous. Each robot possesses a clock which it uses to time its movement. However, the robot’s individual parameters in the form of their walking speed and time unit may or may not be the same for both robots. We study the feasibility of rendezvous in different scenarios, in which some subsets of these parameters are not the same. As the robots are anonymous, they execute the same algorithm and when both parameters are identical the rendezvous is infeasible. We propose a universal algorithm, such that the robots are assured of meeting in finite time, in any case when at least one of the parameters is not equal for both robots.

7.1 Introduction

Rendezvous is concerned with two robots arbitrarily placed in a known search region and moving about until they meet each other. In this paper we will study symmetric rendezvous in which the two robots are instructed to employ the same algorithm. In our setting the robots’ environment is an infinite line on which each robot may move at a constant speed. Each robot is equipped with its own clock which is used to time its movements and this clock is not necessarily consistent between the robots.

The rendezvous problem was studied for numerous models and various types of environments in randomized as well as deterministic settings. The fundamental question related to deterministic rendezvous concerns feasibility, or, more exactly, to identify the parameters of the model for which the rendezvous is possible to achieve (in finite time). The main concern related to the feasibility of rendezvous is that of symmetry breaking. Typical example of symmetry breaking is the use of the robot label in which the robot is aware of its label and may use its value as a parameter.

In the present paper the symmetry is broken yet another way. If the robots differ according to their speeds or private time units, we have a universal algorithm which guarantees rendezvous. Furthermore, and contrary to the case of labeled robots, knowledge as to which of the parameters is different is not necessary. Our robot is completely unaware of the
value(s) of its individual parameters and it does not use them in the computations needed to run the algorithm.

When the rendezvous is feasible, research is concerned with the efficiency of the algorithm, which is usually measured by the time required until the meeting of the two robots takes place. The objective is to design algorithms that achieve good competitive ratios for the time spent by the robots to rendezvous divided by the time spent by the robots if they were running an optimal algorithm.

7.1.1 Model

We consider the symmetric rendezvous problem of two mobile robots \( R \) and \( R' \) modeled as points on the infinite line. The robots are initially located an unknown distance \( d \) from each other and the rendezvous problem is solved if it ever happens that the robots occupy the same position on the line at the same time (i.e. their trajectories intersect). The robots cannot see each other and must employ the same algorithm in order to rendezvous. We assume that robots can store and compute real numbers with arbitrary precision.

We consider a model in which each robot has its own constant speed and in which each is equipped with a clock allowing them to measure their travel time. Each robot will consider itself as the origin of its own coordinate system and it will use its clock to fix the distance unit for this coordinate system as the product of its maximum speed and local time unit. We explicitly consider the possibility that the robots have different speeds and/or clocks. We study algorithms which progress in a synchronous and continuous time model (i.e. robots are always active).

Without loss of generality, we will present our analysis from the viewpoint of the robot \( R \) and thus assume that this robot has maximum unit speed, and that its clock is “correct” in the sense that it agrees with some predefined global coordinate system. On the other hand, we set the speed of \( R' \) as \( v > 0 \), and set its time unit as \( \tau > 0 \) with the result that one time unit as measured by the clock of \( R' \) will actually be \( \tau \) time units as measured by the clock of \( R \). The robots will determine their progress/distance traveled in an algorithm as the product of their travel time and maximum speed.

We specifically focus on three sub-models obtained from the preceding general model by fixing one of \( v, \tau, \) or the product \( v\tau \) to one. In the equal time-unit model (or \( T \)-Model) \( \tau = 1 \), in the equal distance-unit model (or \( D \)-Model) \( v\tau = 1 \), and in the equal speeds model (or \( V \)-Model) \( v = 1 \). Since only one of \( v \) or \( \tau \) is independent in these models we will assume without loss of generality that \( 0 < v < 1 \) in the \( T \)- and \( D \)-Models and take \( 0 < \tau < 1 \) in the \( V \)-Model.

In analyzing the time complexity of our rendezvous algorithms we will employ an adversarial argument in which we assume that an adversary is able to choose the values of \( d \), \( v \), and/or \( \tau \) in order to maximize the competitive ratio of a given algorithm (i.e. we employ a worst-case analysis). Loosely defined, the competitive ratio of a rendezvous algorithm \( A \)
is the maximum ratio of the time it takes the robots to rendezvous using $A$ divided by the
time it would take them given that they are running an optimal algorithm. We will give a
more precise definition of the competitive ratio at a later time.

To end this section we observe that to be most general one should consider the possibility
that the robots also differ in their orientations (i.e. sense of positive direction). However, in
all cases for rendezvous on the line studied here, having different orientations will only help
the robots in achieving their goal. As a result, all derived upper bounds will not be affected
by a difference in orientation and, since including this only serves to complicate notation,
we will not explicitly consider it in this work.

### 7.1.2 Related Work

Search may be viewed as a game between two players having divergent goals, one trying to
hide for as long as possible and the other one attempting to minimize its search time. As
a contrast, in rendezvous the two involved players have converging goals in that they are
aiming to find one another as quickly as possible. The duality of the two problems has been
presented and investigated in the beautiful book [8].

The search problem for one robot on an infinite line was initiated independently by
Bellman [25] and Beck [22]. There have been numerous studies on search even for an
environment as simple as the infinite line, emphasizing various aspects arising from the
capabilities of one or more robots and the status of the search domain. These include
randomized [128], group search [40], linear terrains [77], faulty searchers [78], and turn
costs [84]. The efficiency of linear search is most often measured by the competitive ratio,
which is the time spent by the robot to complete its search divided by the time needed by an
omniscient robot that knows the location of the target. The competitive ratio of 9 is obtained
by the cow-path algorithm (cf. [25] and Beck [22]) and was first proved to be optimal for
stochastic linear search in [23] and deterministically in [13, 15].

The approach to solving the rendezvous problem is most often fundamentally different
from the techniques applied for search (although, somewhat surprisingly, this will not
always be the case in the present paper). The solution of the symmetric rendezvous problem
requires that the two robots are somehow equipped to break symmetry [146]. There has
been extensive research literature concerned with taking advantage of “innate” asymmetries
of the studied model. For example, [83, 154, 86, 89] focus on robots having distinct labels,
[135, 161] on robots equipped with identical tokens that can be placed on selected nodes,
[49] on a robot’s awareness of its GPS position in the environment.

The rendezvous (and its more general version of gathering) problem has been also studied
for robots of different speeds [33, 95], inconsistent compasses [45, 125] and chirality or
sense of direction [18, 34]. However, in the studies previously mentioned, these differences
were obstacles that needed to be circumvented by the suggested algorithms, rather than
used for the benefit of the proposed approach, which is the case of the present paper. To the
best of our knowledge, linear rendezvous for robots with asymmetric clocks has never been studied before.

7.1.3 Results and outline

In this paper we study the rendezvous problem when the robots are equipped with asymmetric clocks. In addition to exploring novel ways of defining rendezvous algorithms, we demonstrate the feasibility of rendezvous and give two algorithms solving rendezvous in finite time provided that at least one of a robot’s maximum speed or time unit differs from the other (see end of Section 7.5). In addition, we study rendezvous in the three restricted models, T-model, D-model, and V-model.

In Section 7.3 we analyze rendezvous in the T-Model and show that possessing equal time-units reduces rendezvous into the problem of search, allowing one to optimally solve the problem with a competitive ratio of 9. In Section 7.4 we analyze rendezvous under the assumption that the robots have equal distance-units, i.e. $v\tau = 1$. Here we get our first taste of the difficulties involved with asymmetric clocks. We show that rendezvous is solved with a competitive ratio of $\frac{105}{11} \approx 9.55$. Furthermore, in the limit of large $d$, the competitive ratio is 9.

In Section 7.5 we analyze rendezvous when the robots’ speeds are equal and $0 < \tau < 1$. This is the most difficult model to analyze and we will observe large differences in the algorithms employed and resulting upper bounds, as compared to the T- and D-Models. We give two algorithms that solve rendezvous, the first with a competitive ratio of $O\left(\frac{\tau \log^2 (d)}{\log (\tau)}\right)$ and the second with a slightly tunable competitive ratio of $O\left(\frac{\tau \log^{1+c} (d)}{c \log (\tau)}\right)$, where $c > 0$ is a parameter of the algorithm. We also offer arguments as to why it may not be possible to achieve a constant competitive ratio in this model.

7.2 Preliminaries and Notation

In this section we introduce some preliminary ideas and notations used throughout our analysis. We begin with some notation.

As usual, for any real number we use $|\cdot|$ to indicate its absolute value, $\log(\cdot)$ to indicate the base-2 logarithm, and $\ln(\cdot)$ to indicate the natural logarithm with base $e$.

7.2.1 Time and position space

It will be useful to consider rendezvous algorithms as specifying a continuous trajectory in a two dimensional space where the horizontal axis represents a robot’s position on the line and the vertical axis represents the flow of time. In this representation one can view the effect of $v$, $\tau$, and $d$ as a scaling and translation of the coordinate axes of the local
xt-space of $\mathcal{R}'$ as compared to $\mathcal{R}$. As a result, if we represent the trajectory $t(x)$ of $\mathcal{R}$ as a (possibly multivalued) function of position on the line, then the actual trajectory of $\mathcal{R}'$ will be $\tau \cdot t \left( \frac{x \pm d}{v \tau} \right)$. Equivalently, if the trajectory of $\mathcal{R}$ is represented as a function of time $x(t)$, then the trajectory of $\mathcal{R}'$ is $v \tau x \left( \frac{t}{\tau} \right) \pm d$. It would help to remember these transformations as they will be used repeatedly throughout our analysis.

7.2.2 Rendezvous algorithms

In what follows assume that we are speaking about the robot $\mathcal{R}$.

We consider algorithms defined by an infinite sequence of turning-points $P_k = (X_k, T_k)$ which specify the time and location on the line at which a robot reverses its direction. On the round $k$ of such an algorithm a robot will move from its initial position to the turning-point $P_k$ and then return to its initial position. The time necessary to finish a round will be $2|X_k|$ and the time at which a robot begins the round $k$ is $2 \sum_{i=0}^{k-1} |X_i|$. The time $T_k$ at which a robot is at the turning-point $P_k$ can thus be written as

$$T_k = 2 \sum_{i=0}^{k-1} |X_i| + |X_k|. \quad (7.1)$$

Since $T_k$ is dependent on $X_k$ we will often refer to $X_k$ as the $k^{th}$ turning-point. We will also adopt the convention that $P_{-1}$ indicates a robot’s starting location and assume without loss of generality that $X_k > 0$ when $k$ is even (i.e. a robot initially moves to the right). Finally, we will always explicitly state whether or not $\mathcal{R}'$ begins on the left or right of $\mathcal{R}$ and assume that $d > 0$.

Using the same terminology as [8] we call an infinite sequence of turning-points periodic and monotonic if, for all $k \geq 0$, it satisfies

$$X_{2k-1} < X_{2k+1} < \cdots < 0 < \cdots < X_{2k} < X_{2k+2}. \quad (7.2)$$

As discussed in [8], an adversary may achieve an arbitrarily large competitive ratio if an algorithm ever has a first turning-point. This results from the ability of an adversary to choose $d$ arbitrarily small with respect to this first turning-point and force $\mathcal{R}$ to initially move away from the initial position of $\mathcal{R}'$. As a result, one either needs to consider doubly infinite sequences of turning-points or, more practically, make the additional assumption that the robots possess knowledge of a lower bound on $d$ (which might be indirectly due to the robot having a finite size or visibility range). In this work we are primarily interested in large values of $d$ and / or $v$ and $\tau$ close to one, we will take the latter approach and assume that the adversary is restricted to values of $d$ that are comparable in size to the first turning-point of a rendezvous algorithm.

\footnote{Technically this was discussed for the case of search but the argument also applies here.}
We observe that any periodic and monotonic algorithm will have turning-points that span a cone-shaped curve $\mathcal{T}(x)$ in $xt$-space satisfying the identity $T_k = \mathcal{T}(X_k)$ (i.e. the turning points lie on the boundary of a cone-shaped curve). This observation allows us to give an alternate, and particularly useful representation of a rendezvous algorithm – we specify the curve $\mathcal{T}(x)$ and have the robots infer where their turning-points are located. Using this method a robot will be instructed to move in one direction until its trajectory in $xt$-space intersects the curve $\mathcal{T}(x)$. It will then reverse its direction and move at full speed until its trajectory again intersects $\mathcal{T}(x)$, and so on. The robots can compute their turning-points for this type of algorithm using the relation $T_k = \mathcal{T}(X_k)$.

If an algorithm is given by its turning-points $X_k$ then we say that $X_k$ induces the curve $\mathcal{T}(x)$. If an algorithm is specified by $\mathcal{T}(x)$ then we say that $\mathcal{T}(x)$ induces the turning-points $X_k$. We call an algorithm symmetric if the curve $\mathcal{T}(x)$ is an even function of $x$. We will be interested in algorithms that are symmetric, periodic, and monotonic and we call $SPM$ the class of all such algorithms. We will use the following formal definition, based on $\mathcal{T}(x)$, for an algorithm to be in the class $SPM$:

**Definition 7.2.1.** An algorithm with turning-points $X_k$ is in the class $SPM$ if the curve $\mathcal{T}(x)$ which induces these turning-points satisfies:

1. $\mathcal{T}(x)$ is an even function, i.e. $\mathcal{T}(x) = \mathcal{T}(-x)$.
2. There exists an $X_0 > 0$ such that $X_0 = \mathcal{T}(X_0)$.
3. For all $|x| > X_0$, $\mathcal{T}(x) > |x|$.
4. $\mathcal{T}(x)$ is continuously differentiable for all $x > X_0$.

One can easily confirm that if $\mathcal{T}(x)$ satisfies the above definition, the induced turning-points will be periodic and monotonic (as per the condition (7.2)). In the sequel we will construct algorithms which belong to the class $SPM$.

We formally define Algorithm 3 and Algorithm 4 which respectively take $X_k$ and $\mathcal{T}(x)$ as parameters.

**Algorithm 3 Rendezvous[$X_k$]**

1: $k = 0$
2: **repeat**
3: Move to the position $X_k$ and return to initial position. $k = k + 1$.
4: **until** Meeting occurs

**Algorithm 4 Rendezvous[$\mathcal{T}(x)$]**

1: Run Algorithm 3 with $X_k$ defined by $T_k = \mathcal{T}(X_k)$. 


So far we have only discussed rendezvous algorithms from the perspective of $\mathcal{R}$. When referring to the turning-points etc. of $\mathcal{R}'$ we will indicate this using a prime. So, for example, the turning-points of $\mathcal{R}'$ will be $P'_k = (X'_k, T'_k)$ and the curve induced by these turning-points will be $T'(x)$. We note that $X'_k = v_\tau X_k \pm d$, $T'_k = \tau T_k$, and $T'(x) = \tau T\left(\frac{x \pm d}{v_\tau}\right)$.

### 7.2.3 Robot trajectories and rendezvous points

Viewed as a whole, a rendezvous algorithm will specify a trajectory in $xt$-space composed of a series of line segments connected at their endpoints. For an algorithm defined by the sequence $X_k$ we can explicitly write the equations of the lines traversed by $\mathcal{R}$ as
\[ t_k(x) = (-1)^k x + T_k - |X_k|. \]  
(7.3)

where $t_k(x)$ is the segment beginning at $P_{k-1}$ and ending at $P_k$. Likewise, the lines traversed by $\mathcal{R}'$ can be written as
\[ t'_k(x) = (-1)^k \frac{x \mp d}{v} + \tau T_k - \tau |X_k|. \]  
(7.4)

We claim the following:

**Lemma 7.2.2.** Assume that we have chosen $|X_k|$ such that Algorithm 3 solves rendezvous. Then, if the robots meet when $\mathcal{R}$ is approaching its $k^{th}$ turning-point and $\mathcal{R}'$ is approaching its $j^{th}$ turning-point then the time of rendezvous can be written as
\[ t_{k,j} = \pm (-1)^k d - (-1)^{k+j} v_\tau (T_j - |X_j|) + T_k - |X_k| \]

where $t_{k,j}$ is the segment starting at $P'_{k-1}$ and ending at $P'_{k}$. The lemma follows from solving the equation $t_k(x) = t'_j(x)$.

### 7.2.4 Competitive ratios

We are interested in algorithms that achieve small competitive ratios where we have defined the competitive ratio of an algorithm $\mathcal{A}$ as the supremum ratio of the time it takes to rendezvous using $\mathcal{A}$ to the time taken to rendezvous if the robots employ an optimal algorithm. We now give more precise definitions.

**Definition 7.2.3.** Let $\mathcal{A}$ be an algorithm solving rendezvous in the $T$- or $D$-Models and let the time of rendezvous be $T_*(v,d)$. Then the competitive ratio of $\mathcal{A}$ is $\text{CR} = \sup_{v,d} \frac{(1-v)T_*(v,d)}{d}$.

**Definition 7.2.4.** Let $\mathcal{A}$ be an algorithm solving rendezvous in the $V$-Model and let the time of rendezvous be $T_*(\tau,d)$. Then the competitive ratio of $\mathcal{A}$ is $\text{CR} = \sup_{\tau,d} \frac{(1-\tau)T_*(\tau,d)}{d}$.

We will justify these definitions as we consider each model in turn.
7.2.5 Feasibility of rendezvous

Here we establish the feasibility of rendezvous. To this end consider an algorithm in the class $SPM$ with the curve $T(x)$. We say that $T(x)$ contains $T'(x)$ (resp. $T'(x)$ contains $T(x)$) if there exists an $x_0$ such that for all $x$ satisfying $|x| > |x_0|$ we have $T(x) \leq T'(x)$ (resp. $T(x) \geq T'(x)$). If such an $x_0$ exists we call $x_0$ and $t_0 = T(x_0)$ the containment point and containment time of $T(x)$ and $T'(x)$. Intuitively, $x_0$ will be an intersection point of $T(x)$ and $T'(x)$. Figure 7.1 illustrates these definitions. We can now claim the following:

**Lemma 7.2.5.** If either of $T(x)$ or $T'(x)$ contains the other, then rendezvous is guaranteed.

*Proof.* Assume that $T(x)$ contains $T'(x)$ and let $t_0$ be the containment time. Let $X_k$ be the first turning-point that $R$ reaches after the time $t_0$ and assume that $X_k > 0$. In this situation $R$ will be on the right of $R'$ once it reaches $X_k$ and will be on the left of $R'$ once it reaches $X_{k+1}$. Likewise, if $X_k < 0$, then $R$ will be on the left of $R'$ at $X_k$ and on the right of $R'$ at $X_{k+1}$. In either case the robots must rendezvous between the turning-points $X_k$ and $X_{k+1}$ of $R$. In a similar manner one can confirm that the robots will rendezvous between the turning-points $X_j$ and $X_{j+1}$ of $R'$ where $X_j$ is the first turning-point $R'$ reaches after the time $t_0$.

In both the $T$- and $D$-models there are a wide range of algorithms in the class $SPM$ with curves $T(x)$ such that $T(x)$ contains $T'(x)$. In particular, if $v \neq 1$ and $T(x)$ is a linear function, then $T'(x) = \tau T \left( \frac{x+d}{v\tau} \right) = \frac{1}{v} T(x \mp d)$ and $T(x)$ will clearly contain this. If $v = 1$ then there are still a variety of curves one can choose (for example, $T(x) = x^2$) and thus rendezvous is guaranteed in general.

![Figure 7.1: Illustration of the idea of containment. On the left $T(x)$ contains $T'(x)$ and on the right $T'(x)$ contains $T(x)$. In both cases the containment point is indicated.](image-url)
7.3 T-Model

In this section we analyze rendezvous under the assumption that the robots have the same time units, i.e. \( \tau = 1 \). This assumption turns out to be rather powerful as it allows us to reduce the problem into the problem of search for a stationary target.

**Theorem 7.3.1.** Rendezvous in T-Model is equivalent to search for a stationary target at distance \( d^* = \frac{d}{1-v} \).

**Proof.** Assume that we have a rendezvous algorithm that specifies the trajectory \( R(t) \) for \( R \). The robot \( R' \) will then follow the actual trajectory \( R'(t) = vR(t) \pm d \). Now consider a coordinate system moving with the robot \( R' \) and scaled by a factor of \( \frac{1}{1-v} \). In this coordinate system \( R' \) will appear to be stationary at the position \( \pm \frac{d}{1-v} \) and \( R \) will appear to move along the trajectory \( R_*(t) = R(t) \). We can thus view this problem as a search problem for a target at distance \( \frac{d}{1-v} \) and any algorithm solving search will also solve rendezvous in the same amount of time.

This clearly justifies our definition of the competitive ratio for this model. Furthermore, this result allows us to draw on many of the results known about search and, in particular, allows one to optimally solve the rendezvous problem.

**Theorem 7.3.2.** Rendezvous in T-Model is optimally solved with a competitive ratio of 9 using Algorithm 3 with \(|X_k| = 2^k \).

**Proof.** Observe that Algorithm 3 with \(|X_k| = 2^k \) is the familiar cow-path algorithm which optimally solves search for a target at distance \( d \) in time \( 9d \). Since search is equivalent to rendezvous in T-Model, this algorithm will optimally solve rendezvous as well.

Note that the optimal solution in this case lies within the class \( SPM \).

7.4 D-Model

In this section we analyze rendezvous under the assumption that the robots have the same distant-units, i.e. \( v\tau = 1 \). Unlike the T-model, we cannot transform the problem into the search problem and thus we will approach our analysis slightly different. We begin with a lemma that justifies our definition of the competitive ratio for this model.

**Theorem 7.4.1.** Rendezvous in D-model takes time at least \( \frac{d}{1-v}\).}

**Proof.** Consider any rendezvous algorithm with turning points \( X_k \). Assume that the robots rendezvous between the \((k - 1)^{th}\) and \( k^{th}\) turning points of \( R \) and the \((j - 1)^{th}\) and \( j^{th}\) turning points of \( R' \). Furthermore, assume that \( R' \) begins to the right of \( R \). In this case \( R \)
must be moving to the right when the robots rendezvous and thus \( k \) will be even. Assume that \( d \) and \( \tau \) are chosen such that the robots rendezvous when \( j \) is also even. Then, by Lemma 7.2.2, the robots will rendezvous at the time

\[
t_{k,j} = \frac{d - (T_j - |X_j|) + (T_k - |X_k|)}{1 - v} = \frac{d - 2 \sum_{i=0}^{j-1} |X_i| + 2 \sum_{i=0}^{k-1} |X_i|}{1 - v}.
\]

Since \( v < 1 \) and \( v \tau = 1 \) we have \( \tau > 1 \) and thus \( R' \) will always take longer to finish a round. We must therefore have \( j \leq k \) and the rendezvous time will be at least \( \frac{d}{1-v} \).

We note that the above lower bound is general and applies to any algorithm (not just those in the class SPM). We now claim the following:

**Theorem 7.4.2.** Rendezvous in D-Model is solved with a competitive ratio of \( \frac{105}{11} \) using Algorithm 3 with \( |X_k| = 2^k \). Furthermore, in the limit of large \( d \), the competitive ratio is 9.

**Proof.** Assume that \( R \) is approaching its \( k^{th} \) turning-point and when \( R' \) is approaching its \( j^{th} \) turning-point the robots rendezvous. Then, by Lemma 7.2.2 they will rendezvous at the time

\[
t_{k,j} = \frac{\pm(-1)^k d - (-1)^{k+j} (T_j - |X_j|) + T_k - X_k}{1 - (-1)^{k+j} v}.
\]

Observe that the robots will never rendezvous when \( R \) is traveling away from \( R' \). Thus, if \( R' \) begins to the right of \( R \), \( k \) must be even. Likewise, if \( R' \) begins to the left of \( R \), \( k \) must be odd. On the other hand, \( R' \) may be moving towards or away from \( R \) when they rendezvous and thus \( j \) may be even or odd. This gives two possible rendezvous times depending on the parity of \( k + j \). If \( k + j \) is even then \( t_{k,j} = \frac{d + 2^{k+1} - 2^{j+1}}{1 + v} \) and if \( k + j \) is odd then \( t_{k,j} = \frac{d + 2^{k+1} + 2^{j+1} - 4}{1 + v} \). The competitive ratios for these two cases are \( CR_+ = 1 + \frac{2}{3}(2^k - 2^j) \) and \( CR_- = \frac{1 + v}{1 + v} + \frac{2}{3}(2^k + 2^j - 2) \) where \( CR_+ \) (resp. \( CR_- \)) corresponds to \( k + j \) even (resp. \( k + j \) odd).

We note that, in order for \( R \) and \( R' \) to meet while traveling towards their \( k^{th} \) and \( j^{th} \) turning-points, the time \( t_{k,j} \) must satisfy \( T_{k-1} \leq t_{k,j} \leq T_k \) and \( T_{j-1} \leq t_{k,j} \leq T'_j \). Furthermore, since \( R' \) will take longer to finish a round, there can be at most one turning-point of \( R' \) between any two turning-points of \( R \).

Now assume that \( k + j \) is even. In this case an adversary will get the best payoff if they choose \( d \) as small as they can without causing the robots to rendezvous on an earlier round. The smallest value of \( d \) is achieved if \( d \) and \( v \) can be chosen such that the \( (k - 2)^{nd} \) turning-point of \( R \) is arbitrarily close to the \( (j - 2)^{nd} \) turning-point of \( R' \) (see the left side of Figure 7.2). We claim that the adversary can choose \( d \) and \( v \) to achieve this. To see why assume that the adversary has chosen \( d \) and \( v \) such that \( T_{k-2} = T'_{j-2} \). In order to rendezvous at the time \( t_{k,j} \) then we need to satisfy \( T_{j-1} \leq t_{k,j} \leq T'_{j} \). Since the robots will not rendezvous when \( R \) is moving away from \( R' \), we can easily conclude that \( T'_{j-1} \leq t_{k,j} \).
Also, since \( t_{k,j} \) must be smaller than \( T_k \), we can also conclude that \( t_{k,j} \leq T'_j \) due to the fact that \( T'_{j-2} = T_{k-2} \) and because \( \mathcal{R}' \) takes longer to finish a round. Thus, in the worst case, we may take \( X_{k-2} = X'_{j-2} \) which, after simplification, tells us that we should take \( d = 2^{k-2} - 2^j - 2^k \). Using this result in the expression for \( CR_+ \) yields a competitive ratio of 9.

Now assume that \( k + j \) is odd. In this case the competitive ratio will depend on both \( d \) and \( v \). However, we can similarly conclude that the adversary would like to minimize \( d \) and thus they will try to choose \( d \) and \( v \) in order to make the \((j - 1)^{st}\) turning-point of \( \mathcal{R}' \) equal (or arbitrarily close) to the \((k - 2)^{nd}\) turning-point of \( \mathcal{R} \). However, in this case, we claim that the adversary cannot always do this. Indeed, if \( T_{k-2} = T'_j \) and \( X_{k-2} = X'_{j-1} \) then, after some manipulation, we find that,

\[
2^{j-1} = \frac{dv}{1-v} + \frac{2}{3} \quad (7.5)
\]

and

\[
2^{k-2} = \frac{d}{1-v} + \frac{2}{3}. \quad (7.6)
\]

Since we are assuming that \( \mathcal{R}' \) meets \( \mathcal{R} \) as it is traveling towards its \( j^{th} \) turning-point we need \( t_{k,j} \leq T'_j \). Using (7.6), (7.5), and the expression for \( t_{k,j} \) we find that \( d \) must satisfy \( d \leq \frac{2}{3} \left( \frac{1}{v} - 1 \right) \) and thus \( j \) must satisfy \( 2^{j-1} = \frac{de}{1-v} + \frac{2}{3} \leq \frac{4}{3} \). Clearly, the only \( j \) that satisfies this is \( j = 1 \) and we can conclude that, in the case that \( k + j \) is odd, the worst-case situations occur when \( j = 1 \) (and \( k \) is even). When \( j = 1 \), we have from (7.5) that \( d = \frac{1-v^3}{3v} \) and from (7.6) that \( 2^k = \frac{4}{3} \left( \frac{1}{v} + 2 \right) \). Using these results in the expression for \( CR_- \) gives us \( CR_- = \frac{15v^9}{1+v} \) and this can be seen to increase with \( v \). However, we cannot simply maximize this over \( v \). If we take \( k = 2 \) then we need \( v = 1 \) and \( d \leq 0 \) and this is clearly not possible. We can thus conclude that \( k \geq 4 \). If \( k = 4 \) then one can confirm that \( v = \frac{1}{10} \), and if \( k > 4 \) then \( v < \frac{1}{10} \). Thus, we find the maximum competitive ratio when \( v = \frac{1}{10} \). Substituting this result in for \( CR_- \) we find that \( CR_- \leq \frac{105}{15} \).

Finally, since we found that \( 2^k = \frac{4}{3} \left( \frac{1}{v} + 2 \right) \) in the worst case, we can see that for very large \( k \) we will have \( v \) very small. Since \( k \) will be large for large \( d \) and since \( CR_- = \frac{15v^9}{1+v} = 9 \) when \( v = 0 \), we can conclude that the competitive ratio approaches 9 as \( d \) gets large.

We note that the upper-bound of Theorem 7.4.2 is tight for the algorithm considered since it is easily confirmed that the competitive ratio is exactly 9.55 when \( v = \frac{1}{10} \) and \( d = 3 \).

What is somewhat surprising about this result is that the competitive ratio is nearly identical to that of the \( T \)-Model when \( d \) is large, and this is achieved using the same algorithm. We suspect that this algorithm is optimal here since, in the limit that \( v \) goes to zero, this model reduces to search for which a competitive ratio of 9 is optimal. We do not have a formal proof of this, however, and thus it is still an open question whether or not this algorithm is optimal.
Figure 7.2: The two worst case possibilities in $D$-Model using Algorithm 3 with $|X_k| = 2^k$. The point of rendezvous is indicated in green and in both cases the robots rendezvous as $R$ is approaching its $j_k^{th}$ turning-point and when $R'$ is approaching its $j^{th}$ turning-point. Left: The case that $k + j$ is even. Right: The case that $k + j$ is odd. The absolute worst-case is depicted and occurs when $d = 3$ and $v = \frac{1}{10}$ such that the competitive ratio is 9.55. Note that the scales of the depicted trajectories are not the same for the two cases.

### 7.5 $V$-Model

In this section we analyze upper bounds on rendezvous when the robots’ speeds are equal and $0 < \tau < 1$. In both the $T$-Model and $D$-Model the robots employed Algorithm 3 with a geometric sequence of turning-points ($|X_k| = 2^k$) in order to rendezvous with a small constant competitive ratio. We will see that the $V$-Model is rather more complicated, and, in particular, a geometric sequence of turning-points will not work (see Lemma 7.5.9 at the end of this section). We will therefore have to employ a different type of algorithm. We begin, however, with a lower-bound to justify our definition of the competitive ratio:

**Theorem 7.5.1.** Rendezvous in $V$-Model takes time at least $\frac{d}{1-\tau}$.

**Proof.** For concreteness assume that $R'$ begins to the right of $R$. Then, the robots will rendezvous as $R$ is traveling to the right and $R'$ is traveling to the left. Consider the $j^{th}$ turning-point of $R'$ and assume that $j$ is even such that $R'$ moves left after this turning-point. Let $X_k$ be the first even turning-point of $R'$ after the $j^{th}$ turning-point of $R'$. Observe that the robots will rendezvous before the $(j + 1)^{st}$ turning-point of $R'$ provided that $X'_j - X_k \leq T_k - T'_j$. Now assume that $\tau$ is close enough to one such that we may assume that $j = k$. In this case we can rewrite the condition $X'_j - X_k \leq T_k - T'_j$ as $T_k + X_k \geq \frac{d}{1-\tau}$.

Since we are free to choose $d$ and $\tau$ let us choose these parameters such that $T_k + X_k = \frac{d}{1-\tau} - \epsilon$ for an arbitrarily small $\epsilon > 0$. Then the robots will not rendezvous until some
time after the \((k + 2)\text{nd}\) turning-point of \(R'\). Since \(T_{k+2} = T_{k+1} + |X_{k+1}| + |X_{k+2}| = T_k + |X_k| + 2|X_{k+1}| + |X_{k+2}| = \frac{d}{1 - \epsilon} - \epsilon + 2|X_{k+1}| + |X_{k+2}|\) we can conclude that the robots will take at least \(\frac{d}{1 - \epsilon}\) time to rendezvous. \hfill \(\square\)

Note that this lower bound applies to any algorithm. Now for an upper bound. We claim the following:

**Theorem 7.5.2.** Rendezvous in V-Model is solved with a competitive ratio of \(\frac{18 \log^2(d) \tau}{|\log(\tau)|} + O\left(\frac{\log(d) \tau}{|\log(\tau)|}\right)\) using Algorithm 3 with \(|X_k| = (k + 2)2^k\).

An overview of the proof of this theorem is as follows. We first show that Algorithm 3 with \(|X_k| = (k + 2)2^k\) does indeed solve rendezvous and we will do this by demonstrating that \(T(x)\) contains \(T'(x)\). Since \(T(x)\) contains \(T'(x)\), the robots will be guaranteed to rendezvous by the second turning-point reached by \(R\) after the containment time and we will use this fact to bound the rendezvous time. Throughout the proof we will need to make use of the Lambert-W function (or simply Lambert function) \(W(x)\) which is defined as the inverse function of \(f(x) = xe^x\) (we consider the real valued branches only and thus \(x\) is restricted to the range \(x \geq -\frac{1}{e}\)). Since \(W(x)\) is multivalued on the open interval \((-e^{-1}, 0)\) it is usual to define \(W_{-1}(x)\) as the branch which attains values \(\leq -1\) and reserve the use of \(W(x)\) to refer to the principal branch which attains values \(\geq -1\). We will need the following properties of \(W(x)\) which are found in, or trivially derived from, the results in [119] and [53]:

**Lemma 7.5.3.** The two real valued branches \(W(x)\) and \(W_{-1}(x)\) satisfy:

\[
W(x) \leq \ln(x) - \ln(\ln(x)) + \frac{e}{e - 1} \cdot \frac{\ln(\ln(x))}{\ln(x)}, \quad x \geq e \tag{7.7}
\]

\[
W(x) \geq \ln(x) - \ln(\ln(x)) + \frac{\ln(\ln(x))}{2 \ln(x)}, \quad x \geq e \tag{7.8}
\]

\[
W_{-1}(x) < \ln(-x), \quad x \geq e \tag{7.9}
\]

\[
\frac{d}{dx} W(x) = \frac{1}{1 + W(x)}, \quad x \geq \frac{1}{e} \tag{7.10}
\]

\[
\frac{d^2}{dx^2} W(x) < 0, \quad x \geq \frac{1}{e} \tag{7.11}
\]

Before we can demonstrate that the algorithm solves rendezvous we need to first determine the curve \(T(x)\) induced by the turning-points \(|X_k| = (k + 2)2^k\).

**Lemma 7.5.4.** Let \(T(x)\) be the curve induced by Algorithm 3 with \(|X_k| = (k + 2)2^k\). Then

\[
T(x) = 3|x| - \frac{4 \ln(2)|x|}{W(4 \ln(2)|x|)}.
\]

207
Proof. One can observe that the turning-points \(|X_k| = (k + 2)2^k\) form an arithmetico-geometric sequence and its sum has the closed form expression \(\sum_{i=0}^{k-1} |X_i| = k2^k\). We therefore have \(T_k = 2\sum_{i=0}^{k-1} |X_i| + |X_k| = (3k + 2)2^k\). We may rewrite \(T_k\) as \(T_k = 3(k + 2)2^k - 4 \cdot 2^k = 3|X_k| - \frac{4|X_k|}{k+2}\).

To express \(T_k\) fully in terms of \(|X_k|\) we need to invert \(|X_k| = (k + 2)2^k\). We can do this using the Lambert function. First rewrite the equation \(|X_k| = (k + 2)2^k\) as \(4\ln(2)|X_k| = \ln(2)(k + 2)e^{\ln(2)(k+2)}\). In this form we can directly apply the definition of the Lambert function to get the solution \(\ln(2)(k + 2) = W(4\ln(2)|X_k|)\). We can therefore express \(T_k\) as \(T_k = 3|X_k| - \frac{4\ln(2)|X_k|}{W(4\ln(2)|X_k|)}\). Since \(\mathcal{T}(X_k) = T_k\) the lemma follows.

Now that we have determined the curve \(\mathcal{T}(x)\) induced by the turning points \(|X_k| = (k + 2)2^k\), we can show that \(\mathcal{T}(x)\) will contain \(\mathcal{T}'(x)\) and thus the algorithm will solve rendezvous.

Lemma 7.5.5. Consider Algorithm 3 with \(|X_k| = (k + 2)2^k\). Then \(\mathcal{T}(x)\) contains \(\mathcal{T}'(x)\).

Proof. We need to show that, for all \(d\) and \(\tau < 1\), there exists an \(x_0\) such that for all \(|x| > x_0\) we have \(\mathcal{T}'(x) > \mathcal{T}(x)\). To do this we will assume that \(d = 0\) and show that the difference \(D(x) = \mathcal{T}'(x) - \mathcal{T}(x)\) grows without bound for all \(\tau\) satisfying \(0 < \tau < 1\). If this is the case, then, no matter the value of \(d\), there will eventually be an \(x_0\) such that for all \(x > x_0\) we have \(\mathcal{T}'(x) > \mathcal{T}(x)\).

To this end consider the rate of change of \(\mathcal{T}(x)\) for \(x > 0\). Using (7.10) we find that \(\frac{d\mathcal{T}(x)}{dx} = 3 - \frac{4\ln(2)}{1+W(4\ln(2)x)}\). Since \(W(0) = 0\) and \(W(x)\) is an increasing function, \(\mathcal{T}(x)\) is also increasing for all \(x > 0\). Furthermore, if \(d = 0\) we have \(\frac{d\mathcal{T}(x)}{dx} = 3 - \frac{4\ln(2)}{1+W(4\ln(2)x)}\), and this is clearly larger than \(\frac{d\mathcal{T}(x)}{dx}\) for \(0 < \tau < 1\). The rate of change of \(D(x)\) must therefore always be positive and thus \(D(x)\) does indeed grow without bound.

We need one more simple lemma before tackling the proof of Theorem 7.5.2

Lemma 7.5.6. Consider Algorithm 3 with \(|X_k| = (k + 2)2^k\) and let \(\mathcal{T}(x)\), \(x_0\), and \(t_0\) be the induced curve, and containment time and position. Then \(t_0 < 3x_0\) and \(\tau\mathcal{T}(x - d) > \mathcal{T}(x) - 3d\).

Proof. The first part of the lemma follows easily from the fact that \(\frac{d\mathcal{T}(x)}{dx} < 3\). The second part also follows easily from the facts that \(\frac{d\mathcal{T}(x)}{dx} < 3\) and \(\mathcal{T}(x)\) is a convex function (see (7.11)). As a result, \(\mathcal{T}(x)\) always lies below any secant line, and every secant line will have a slope less than three.

Proof. (Theorem 7.5.2) We would like to bound the rendezvous time and to do this we will first bound the containment time. Since the robots must rendezvous by the second turning-point of \(\mathcal{R}\) after the containment time, and since \(\frac{|X_{k+2}|}{|X_k|} \leq 8\), the rendezvous time
will be bounded by eight times the containment time. By Lemma 7.5.6 the containment time is itself bounded by three times the containment position and thus we will actually determine a bound on the containment position. Thus, if \( T_s, t_0, \) and \( x_0 \) are respectively the rendezvous time, containment time, and containment position, then \( T_s < 8t_0 < 24|x_0| \). We will assume that \( x_0 > 0 \) and note that, since \( \mathcal{T}(x) \) contains \( \mathcal{T}'(x) \), we must have \( d > 0 \) in order for \( x_0 > 0 \).

To begin, we note that \( x_0 \) is the solution to the equation \( \mathcal{T}'(x) - \mathcal{T}(x) = 0 \), and, since this difference is increasing, any \( x \) satisfying \( \mathcal{T}'(x) - \mathcal{T}(x) > 0 \) will suffice for a bound. In particular, since \( \mathcal{T}'(x) = \tau \mathcal{T}(\frac{x-d}{d}) > \tau \mathcal{T}(\frac{x}{d}) - 3d, \) we have \( \mathcal{T}'(x) - \mathcal{T}(x) > \tau \mathcal{T}(\frac{x}{d}) - \mathcal{T}(x) - 3d \) and we will thus bound \( x_0 \) by an \( x \) satisfying \( \tau \mathcal{T}(\frac{x}{d}) - \mathcal{T}(x) > 3d \).

By (7.8) and (7.7) we can write \( D(y) = y \cdot f(y) \). We therefore wish to find a \( y \) satisfying \( D(y) = y \cdot f(y) > 3d \). By (7.8) and (7.7) we can write \( f(y) \geq \frac{1}{\ln(y) - \ln(\ln(y))} - 1 \)
\[
\ln(y) - \ln(\ln(y)) + \frac{\ln(\ln(y))}{\ln(y)}
\]
Now set \( z = \ln(y) \). The right hand side of the above inequality then becomes
\[
th(z) = \frac{1}{z - \ln(z) + \frac{\ln(z)}{e - 1}} - \frac{1}{z - \ln(\tau) + \ln(z - \ln(\tau)) + \frac{\ln(\ln(y))}{2(z - \ln(\tau))}}.
\]
One can confirm that \( h(z) \) admits a generalized Puiseaux series in the limit \( z \to \infty \). Keeping only the leading term of this series we find that \( h(z) = \frac{\ln(z)}{z^2} + O \left( \frac{\ln^2(z)}{z^3} \right) \). We can therefore conclude that \( f(y) > \frac{\ln(z)}{\ln^2(y)} \), \( D(y) > \frac{\ln(\tau)}{\ln^2(y)} \), and we now wish to find a \( y \) satisfying \( \frac{\ln(\tau)}{\ln^2(y)} \geq 3d \). Let \( y_+ \) be the solution to \( \frac{\ln(\tau)}{\ln^2(y)} = 3d \). We can use the Lambert W function to solve this equation. We find that \( y_+ = \frac{12d}{\ln(\frac{1}{\tau})} W_2 \left( -\frac{\ln(\frac{1}{\tau})}{12d} \right) \). Since the rendezvous time satisfies \( T_0 < 8t_0 < 24x_0 < \frac{6y_+}{\ln(2)} \), we find that \( T_s < \frac{72d}{\ln(2) \ln(\frac{1}{\tau})} \frac{1}{W_2 \left( -\frac{\ln(\frac{1}{\tau})}{12d} \right)} \). To express this in terms of more familiar functions we can use (7.9) to write
\[
T_s < \frac{72d}{\ln(2) \ln(\frac{1}{\tau})} \ln^2 \left( \sqrt{\frac{\ln(\frac{1}{\tau})}{12d}} \right) = \frac{18d \ln^2(d)}{\ln(2) \ln(\frac{1}{\tau})} + O \left( \frac{d \ln(d)}{\ln(\frac{1}{\tau})} \right)
\]
Expressing \( T_s \) with base-2 logarithms and dividing by \( \frac{d}{\ln(\frac{1}{\tau})} \) gives the desired bound on the competitive ratio.

If we abandon the use of algorithms with turning-points that are easily defined, then we can get an algorithm with a competitive ratio which is slightly “tunable”. We claim the following:
Theorem 7.5.7. Rendezvous in $V$-Model is solved with a competitive ratio of \( \frac{72 \ln^c (2) d \log(d)^{1+c}}{c | \log(\frac{1}{t})|} \) + \( O \left( \frac{d \log^c(d)}{\epsilon \log(\frac{1}{\epsilon})} \right) \) using Algorithm 4 with \( T(x) = 3|x| - \frac{|x|}{\ln^c(|x|)} \) where \( c > 0 \) is a parameter of the algorithm.

As the proof of Theorem 7.5.7 is essentially identical to that for Theorem 7.5.2 we do not provide it here.

There are a couple of things to note about this upper bound. First, although we can reduce the exponent of the \( \log(d) \) term by making \( c \) small, we cannot make it arbitrarily small without suffering a large multiplicative constant due to the \( \frac{1}{c} \) term in the competitive ratio. Thus, the algorithm of Theorem 7.5.7 will be most useful if one knows a lower bound on \( d \) as this will allow one to compare the bounds of Theorem 7.5.2 and 7.5.7 and choose an appropriate \( c \). Without this knowledge it may be better to just stick with the algorithm of Theorem 7.5.2 as it has the benefit of having simple turning-points.

The upper-bounds of Theorem 7.5.2 and 7.5.7 are clearly much worse than the competitive ratios found for both the \( T \)- and \( D \)-Models. In those cases we had a constant competitive ratio and in these cases the competitive ratio is unbounded. One might then expect that we can do better. This, however, does not seem to be the case. We provide two arguments for this. First off, if one tries to use an algorithm in which the leading term of \( T(x) \) is \( \omega(x) \) then we arrive to a similar result – the competitive ratio is unbounded.

Lemma 7.5.8. If the leading term of \( T(x) \) is \( \omega(x) \) then the competitive ratio is \( \omega(1) \).

Proof. We observe that before the robots can rendezvous, there must be, at the very least, an intersection point \((x_*, t_*)\) of the curves \( T(x) \) and \( T'(x) \) satisfying \( 0 < |x_*| < d \). We will show that we can always choose \( d \) and \( \tau \) to make the time \( t_* = \omega \left( \frac{d}{1-\tau} \right) \) if the leading term of \( T(x) \) is \( \omega(x) \). For concreteness, assume that \( R' \) begins to the right of \( R \) such that \( x_* \) is the intersection point of the right arm of \( T(x) \) and the left arm of \( T'(x) \). Furthermore, since \( T(x) = \omega(x) \), we will express the leading order term of \( T(x) \) as \( x \cdot f(x) \) for some positive function \( f(x) = \omega(1) \).

We claim that \( x_* \) is bigger than \( \frac{d}{2} \). Indeed, the right arm of \( T(x) \) is bounded from below by \( x \) and the left arm of \( T'(x) \) is bounded from below by \( d - x \). Since \( x = d - x \) when \( x = \frac{d}{2} \), we must have \( x_* > \frac{d}{2} \). However, if \( x_* > \frac{d}{2} \) then \( t_* > T(\frac{d}{2}) > \frac{d}{2} f(\frac{d}{2}) \). For any fixed \( \tau \neq 1 \) we can take \( d \) large enough that \( \frac{d}{2} f(\frac{d}{2}) > \frac{d}{1-\tau} g(\frac{d}{1-\tau}) \) for an appropriately chosen function \( g(x) = \omega(1) \). Thus, \( t_* = \omega \left( \frac{d}{1-\tau} \right) \) and, since the rendezvous time is larger than \( t_* \), the lemma follows.

Thus, the two algorithms analyzed in this section do seem to be from the “right” class of algorithms one should consider if one is to hope for a constant competitive ratio. The next lemma – which demonstrates why a geometric sequence of turning-points cannot be used – also supports this conclusion:

210
Lemma 7.5.9. If $\mathcal{T}(x)$ has the form $\mathcal{T}(x) = ax + G(x)$ with $a > 1$ and $|G(x)| = O(1)$ then there are choices of $\tau \neq 1$ such that the robots will never rendezvous.

Proof. Let us first determine what the turning-points of $\mathcal{T}(x)$ look like. Since the turning-points satisfy the relation $\mathcal{T}(X_k) = T_k$ and $T_k = 2\sum_{i=0}^{k-1} |X_i| + |X_k|$ we have $a|X_k| + G(X_k) = 2\sum_{i=0}^{k-1} |X_i| + |X_k|$ and this simplifies to $|X_k| = \frac{2}{a-1} \sum_{i=0}^{k-1} |X_i| - G(X_k)$. Set $G_k = G(X_k)$ and observe that

$$|X_{k+1}| = \frac{2}{a-1} \sum_{i=0}^{k} |X_i| - G_{k+1} = \frac{2}{a-1} |X_k| + \left( \frac{2}{a-1} \sum_{i=0}^{k-1} |X_i| - G_k \right) + G_k - G_{k+1}$$

$$= \frac{a+1}{a-1} |X_k| + G_k - G_{k+1}$$

and we can therefore see that $X_{k+1}$ is nearly a geometric sequence with common ratio $\frac{a+1}{a-1}$.

Let $b = \frac{a+1}{a-1}$, and assume that $\tau = b^{-2}$ and that $\mathcal{R}'$ begins to the right of $\mathcal{R}$. In this case we can write

$$|X_{k+2}| = b|X_{k+1}| + G_{k+1} - G_{k+2} = b(b|X_k| + G_k - G_{k+1}) + G_{k+1} - G_{k+2}$$

$$= b^2 |X_k| + bG_k - (b-1)G_{k+1} - G_{k+2}.$$  

Combining this with the fact that $|X'_{k+2}| = \tau|X_{k+2}| + d$ gives us

$$|X'_{k+2}| - |X_k| = d + \frac{1}{b^2} [bG_k - (b-1)G_{k+1} - G_{k+2}] = d + \Delta_k.$$ 

implying that the robot $\mathcal{R}'$ will be a distance $d + \Delta_k$ from its $(k+2)^{th}$ turning-point when $\mathcal{R}$ reaches its $k^{th}$ turning-point. In order to rendezvous, $d + \Delta_k$ must eventually decrease to zero. However, since $G(x) = O(1)$, we can always take $d$ sufficiently large such that this difference is bounded from below by a positive constant. Thus, with an appropriate choice of $\tau$ and $d$, the robots will never rendezvous.

We are thus left with a rather small class of functions that $\mathcal{T}(x)$ can belong to -- if $\mathcal{T}(x) = \omega(x)$ then the algorithm will take too much time, and if $\mathcal{T}(x) = a \cdot x \pm O(1)$ then rendezvous cannot be solved. Thus, the only possibility left is if $\mathcal{T}(x) = a \cdot x + f(x)$ with $|f(x)| = o(x)$ and $|f(x)| = \omega(1)$. The two algorithms analyzed in this section each used curves of this form.

It is interesting to note that simulations of the two algorithms in this section show that there are choices of $d$ and $\tau$ to (nearly) match the upper-bounds derived here. Even more interesting is that, in order to achieve these worst-case situations, one chooses $d$ and $\tau$ precisely so that the two robots have turning-points that are arbitrarily close to the containment point of the curves $\mathcal{T}(x)$ and $\mathcal{T}'(x)$. This reflects a similar argument we made when we derived an upper-bound on rendezvous in the $D$-Model. In that case it also turned
out that there were choices of $d$ and $\tau$ in order to achieve the upper bound. If one could prove that it is always possible to choose $d$ and $\tau$ such that the robots do have turning-points arbitrarily close to the containment point for all algorithms with $T(x) = a \cdot x + f(x)$, $|f(x)| = o(x)$, and $|f(x)| = \omega(1)$ then a lower-bound that grows with $d$ would easily follow. This is easier said than done, however, and we leave it as an open problem whether or not one can achieve a constant competitive ratio in the $V$-Model.

Finally, we note that both algorithms provided in this section are universal in the sense that they will also solve the problem if both $v$ and $\tau$ are different than one (it is trivial to see that one of $T(x)$ or $T'(x)$ will contain the other if $v \neq 1$). Since a robot does not need to know the values of its parameters in order to employ these algorithms we can conclude that it is sufficient to rendezvous if at least one of $v$ or $\tau$ is different than one.

**Theorem 7.5.10.** Both of Algorithm 3 with $X_k = (k + 2)2^k$ and Algorithm 4 with $T(x) = 3|x| - \frac{|x|}{\ln(c|x|)}$ solve rendezvous in general if at least one of $v$ or $\tau$ is different than one.

The time complexity of this more general model does not turn out to be all that interesting to study since, in the worst cases, an adversary chooses $v = 1$. Thus, the general model reduces to the $V$-Model and all of the results derived here still apply.

### 7.6 Discussion and conclusion

The focus of our paper was on symmetric rendezvous on an infinite line for two robots endowed with asymmetric clocks. After introducing the new concept of asymmetric clocks, we gave a universal algorithm which ensures feasibility of rendezvous if at least one of the robots’ maximal speeds or time units differ. We analyzed the impact of equal time-unit, distance-unit, and equal speeds of the robots on the competitive ratio of the cost of rendezvous. The problem considered not only provides a surprising twist to the well-known rendezvous problem on an infinite line, it also creates interesting avenues for future research. These may include improving the algorithms, tightening bounds, employing robots that may have alternative capabilities (visibility and variable speed), as well as extensions to gathering for multiple robots.

### References


Chapter 8

Symmetry Breaking in the Plane: Rendezvous by Robots with Unknown Attributes

This chapter presents the paper “Symmetry Breaking in the Plane: Rendezvous by Robots with Unknown Attributes” which appeared in the conference proceedings of PODC2019. This is a continuation of the problem studied in Chapter 7 now concerning the case that the agents are located on the infinite plane.
Symmetry Breaking in the Plane: Rendezvous by Robots with Unknown Attributes

Jurek Czyzowicz\textsuperscript{1}, Ryan Killick\textsuperscript{2}, Evangelos Kranakis\textsuperscript{2}, Leszek Gąsieniec\textsuperscript{3}

\textsuperscript{1}Université du Québec en Outaouais, Gatineau, Québec, Canada
\textsuperscript{2}School of Computer Science, Carleton University, Ottawa, Ontario, Canada
\textsuperscript{3}Department of Computer Science, University of Liverpool, Liverpool, UK

Abstract

We study a fundamental question related to the feasibility of deterministic symmetry breaking in the infinite Euclidean plane for two robots that have minimal or no knowledge of the respective capabilities and “measuring instruments” of themselves and each other. Assume that two anonymous mobile robots are placed at different locations at unknown distance \(d\) from each other on the infinite Euclidean plane. Each robot knows neither the location of itself nor of the other robot. The robots cannot communicate wirelessly, but have a certain nonzero visibility radius \(r\) (with range \(r\) unknown to the robots). By rendezvous we mean that they are brought at distance at most \(r\) of each other by executing symmetric (identical) mobility algorithms. The robots are moving with unknown and constant but not necessarily identical speeds, their clocks and pedometers may be asymmetric, and their chirality inconsistent.

We demonstrate that rendezvous for two robots is feasible under the studied model iff the robots have either: different speeds; or different clocks; or different orientations but equal chiralities. When the rendezvous is feasible, we provide a universal algorithm which always solves rendezvous despite the fact that the robots have no knowledge of which among their respective parameters may be different.

\section{8.1 Introduction}

Two anonymous robots (represented as points) are placed at unknown but different locations on the infinite Euclidean plane. They can move with constant speeds (which are not guaranteed to be different), their clocks may be asymmetric (they do not run on the same time unit), the distance units they may be able to measure may be different, their orientation and chirality possibly inconsistent, and their initial Euclidean distance unknown to them. To make matters worse, the robots not only cannot communicate wirelessly, but also all of the above parameters, namely moving-speed, clock-speed, distance-unit, orientation, and
chirality, are unknown to them. The only ability the robots have is limited visibility so they can see each other only if they are within a given range, albeit this is also unknown to them and its magnitude may not be related in any way to their initial distance.

The question arising in this setting is: can the two robots rendezvous? I.e., can they meet at the same point in the plane? It is not at all obvious that a solution could exist in the general setting described above, where knowledge of their moving-speeds, and the consistency of their clock-speeds, distance-units, orientations, and chiralities cannot be assured. In fact, it might even seem counter-intuitive that the problem could be solvable. But if it were, one would want to know what strategies should the robots employ so as to minimize their rendezvous time. Note that the robots operate in the infinite Euclidean plane on which each robot may move with its constant speed. We are interested in symmetric rendezvous so that the robots must execute the same strategy as opposed to the corresponding asymmetric rendezvous problem which may have an optimal solution if one robot waits at its original location while the other is searching for it.

The fundamental problem in the question posed above is related to the feasibility of deterministic rendezvous. More broadly, one is interested to identify the parameters of the given model under which rendezvous is possible to achieve in finite time. Evidently, the overall concern of any strategy is how to break symmetry. However, unlike traditional ways of symmetry breaking where a robot can make use of unconcealed parameters, such as tokens, markers, white-boards, and labels, in our setting we are interested in the possibility of rendezvous under unknown built-in attributes, which include parameters important to the operation of the robots, such as moving-speed, clock-speed, distance-unit, orientation, and chirality, which may not have to be revealed by the robots during the execution of the rendezvous algorithm. Our overall objective is to design algorithms that prove the feasibility of rendezvous in such a constrained environment and whenever possible achieve good performance for the time spent by the robots to rendezvous.

In the present paper we show how two robots placed at unknown distance in the Euclidean plane can break symmetry when some of their unknown built-in attributes, including moving-speed, clock-speed, distance-unit, orientation, and chirality are different, and in some instances leading to a universal algorithm which guarantees rendezvous. Furthermore, and contrary to the case of rendezvous for robots with unconcealed parameters studied in the past, knowledge as to which of the parameters is different may be unnecessary and in any case may not even be given as input to the algorithm. Moreover our robots are completely unaware of the value(s) of their individual hidden parameters and do not make use of them in the computations needed to run the algorithm.

When no parameter of our scenario permits to break symmetry, the robots walk indefinitely and the rendezvous never takes place. However, as the robots have no knowledge of their parameters, it is obvious that they can never stop their walks and conclude that the rendezvous is indeed infeasible.
8.1.1 Model

We consider the symmetric rendezvous problem of two mobile robots \( \mathcal{R} \) and \( \mathcal{R}' \) modeled as points on the infinite Euclidean plane. The robots are initially located an unknown distance \( d \) from each other and each has a non-zero radius of visibility \( r \). The rendezvous problem is solved the first time that the robots see each other, i.e. the first time they are a distance at most \( r \) from each other. The robots must employ the same algorithm in order to rendezvous. We assume that robots can store and compute real numbers with arbitrary precision.

We consider a model in which each robot has its own constant speed and in which each is equipped with a clock and compass allowing them to respectively measure their travel time and travel direction. We assume that robots can use their clocks to measure arbitrarily small time intervals. Each robot will consider itself as the origin of its own coordinate system and it will use its clock and compass to fix the distance unit (the product of its speed and local time unit) and orientation of its coordinate axes. We explicitly consider the possibility that the robots have different speeds, clocks, and/or compasses. We study algorithms which progress in a synchronous and continuous time model (i.e. robots are always active) and where robots can be instructed to move to any real position on the plane.

Without loss of generality, we will present our analysis from the viewpoint of the robot \( \mathcal{R} \) and thus assume that this robot has maximum unit speed, and that its clock and compass are “correct” in the sense that they agree with some predefined global coordinate system. On the other hand, we set the speed of \( \mathcal{R}' \) as \( v > 0 \); its time unit as \( \tau > 0 \); its orientation as \( \phi \in [0, 2\pi) \); and its chirality as \( \chi = \pm 1 \). The overall effect of these differences in reference frames is that the robots will follow different trajectories despite them using the same algorithm. More specifically, \( v \neq 1 \) implies the robots have different speeds and will therefore travel different distances in the same unit of time; \( \tau \neq 1 \) implies that one time unit as measured by the clock of \( \mathcal{R}' \) will in fact be \( \tau \) time units as measured by that of \( \mathcal{R} \); \( \phi \in [0, 2\pi) \) implies that the coordinate axes of \( \mathcal{R}' \) have been rotated (counter-clockwise) by an angle \( \phi \) with respect to those of \( \mathcal{R} \); and \( \chi = -1 \) implies that \( \mathcal{R}' \) and \( \mathcal{R} \) disagree on the \(+y\) direction.

8.1.2 Related work

Rendezvous problems are well-known in the scientific literature. They are not only of theoretical interest due to the fundamental challenges one encounters to provide adequate mathematical solutions. They are also encountered in numerous applications which include the fields of operations research, search and rescue operations and planning in the mathematical sciences, as well as process synchronization, operating system design, and message sharing in computer science.

The rendezvous problem was first introduced informally in 1976 by Steve Alpern in [5] who later also formulated and formalized the continuous time version of the problem in [7].

218
A further impetus to the problem was given by the seminal book treatment \cite{8} where rendezvous was viewed as a search-game between two players having the converging goal in that they are aiming to find one another as quickly as possible.

Numerous papers on rendezvous followed, covering several cases depending on various parameters of the model: type of environment (graph or geometric), robot's knowledge about the environment (partial or complete), anonymity of the robot (labeled or not), robot movement mode (synchronous, semi-synchronous or asynchronous), algorithm type (deterministic or randomized), reliability issues related to robot instruments, etc. A survey covering deterministic rendezvous algorithms is presented in \cite{145} while the monograph \cite{133} is dedicated to the ring and torus.

Many papers on rendezvous adopt the discrete model, e.g., where the robots may meet only at graph nodes (e.g., \cite{152}). In the continuous model for graphs, it is possible to consider the agents' meeting in the interior of graph edges (e.g., \cite{89}). However, when the continuous environment is a two-dimensional plane it is necessary to equip robots with devices permitting non-zero visibility, i.e. the rendezvous arises when the robots belong to each other's visibility range (e.g., see \cite{10}).

Our formulation and analysis of rendezvous is based on the continuous time model in which robots are always active. Recent studies in this model include \cite{131, 138} in which the authors focus on distributed local protocols concerning swarms that result in formations like “gathering at one point”. In \cite{75} gathering at a point in the plane is analyzed when some of the robots may be faulty. They design algorithms which achieve gathering of all reliable robots within the smallest possible time. In fact, they minimize the competitive ratio, defined as the ratio of the time required to achieve gathering of the reliable robots, to the time required for such gathering to occur under the assumption that the reliable robots were known in advance.

The rendezvous (and its more general version of gathering) problem has also been studied for robots with different speeds \cite{33, 95}, inconsistent compasses \cite{45, 125} and chirality or sense of direction \cite{18, 34}. In \cite{124} the authors study the feasibility of gathering by mobile robots that have $\phi$-absolute error dynamic compasses, which allows the angle difference between a local coordinate system and the global coordinate system to vary with time in the range of $[0, \phi]$. In \cite{130} a gathering problem is discussed for robots equipped with inaccurate (incorrect) compasses which may point a different direction from other robots’ compasses. However, in the studies previously mentioned, these differences were obstacles that needed to be circumvented by the suggested algorithms, rather than used for the benefit of the proposed approach, which is the case of the present paper. To the best of our knowledge, rendezvous in the plane for two robots with unknown attributes in continuous time has never been studied before.

The fundamental issue of the rendezvous problem is symmetry breaking, for example by exploiting some specific parameter(s) of the model that permit robots to act differently and not to be trapped in the same relative position to one another (cf. \cite{145}). If there is
no such parameter available it may be shown that the rendezvous is infeasible (e.g., \cite{87}). For rendezvous in graph environments, the symmetry may be broken using asymmetry in the graph topology or the robots’ positions within it (see \cite{76}). In the case of rendezvous in the two-dimensional plane this is not possible and a symmetry-breaking procedure may exploit, for example, the difference of the robots labels \cite{81,89} or robot’s knowledge of its own position in the Cartesian plane \cite{50}. In all these cases the designer of the rendezvous algorithm needs to know what is this parameter of the studied scenario that permits the symmetry breaking. In the present paper it is shown that the knowledge, which of the parameters of the studied scenario makes the rendezvous possible, is not necessary.

Closely related to our current work is \cite{74} which considers rendezvous of two anonymous robots on an infinite line when their walking speed and time units (which are unknown to the robots) may or may not be the same. The authors introduced the new concept of asymmetric clock and proposed a universal algorithm, so that the robots rendezvous in finite time, in any case when at least one of the parameters is not identical for the two robots.

8.1.3 Results and outline

The structure and results of the paper are as follows. In Section 8.2 we consider the problem of search and provide algorithms solving this problem nearly optimally. These algorithms will then form the basis for our rendezvous algorithms. In Section 8.3 we study the rendezvous problem under the assumption that the robots’ time-units are equal. We demonstrate in this setting that rendezvous is always feasible if the robots differ in their speeds, and is only feasible when \( v = 1 \) if also \( \chi = 1 \) but \( 0 < \phi < 2\pi \). When the rendezvous is feasible we provide an algorithm which solves rendezvous in nearly optimal time.

In Section 8.4 we study the rendezvous problem when the robots’ time units differ. In this case we demonstrate that rendezvous is always feasible when \( \tau \neq 1 \). Moreover, we provide an algorithm which universally solves rendezvous whenever the parameters of the robots are such that rendezvous is feasible. This algorithm does not require the robots to know which of their parameters differ. In Section 8.5 we conclude with a short discussion.

8.2 Search

The problem of search is as follows. We have a single robot \( R \) with radius of visibility \( r > 0 \) which needs to find a stationary target (exit, treasure, etc.) that is initially at an unknown distance \( d \) from the robot. This problem was solved in \cite{148} where it was shown that the search time was in \( O \left( \log \left( \frac{d^3}{r} \right) \frac{d^2}{r} \right) \) and that this search time is optimal. We give a slightly different algorithm to the one in \cite{148} which solves the problem in \( O \left( \log \left( \frac{d^3}{r} \right) \frac{d^2}{r} \right) \). We will use this search algorithm to build our rendezvous algorithms.
We construct our search algorithm using a number of smaller procedures/algorithms. The first two of these procedures are called \( \text{SearchCircle}(\delta) \) and \( \text{SearchAnnulus}(\delta_1, \delta_2, \rho) \). The procedure \( \text{SearchCircle}(\delta) \) takes as input a positive real \( \delta \) and instructs a robot to move from its initial position to a radius \( \delta \), move along the circle of radius \( \delta \), and return to its initial position. The procedure \( \text{SearchAnnulus}(\delta_1, \delta_2, \rho) \) takes three positive real parameters \( \delta_1, \delta_2, \) and \( \rho \). It repeatedly calls \( \text{SearchCircle}(\cdot) \) with the end result that all points within the annulus of inner and outer radii \( \delta_1 \) and \( \delta_2 \) respectively have been within a distance at most \( \rho \) from the robot at some time during the algorithm. These procedures are formally described as Algorithms 5 and 6 respectively.

**Algorithm 5 SearchCircle(\( \delta \))**

**Input:** \( \delta > 0 \) (real);

**Begin:**
1: Move along \( x \)-axis to radial position \( \delta \).
2: Traverse circle with radius \( \delta \).
3: Return to initial position.
**End**

**Algorithm 6 SearchAnnulus(\( \delta_1, \delta_2, \rho \))**

**Input:** \( \delta_1 \geq 0 \) (real); \( \delta_2 > \delta_1 \) (real); \( \rho > 0 \) (real);

**Begin:**
1: for \( i = 0 \) to \( \left\lceil \frac{\delta_2 - \delta_1}{2\rho} \right\rceil \) do \( \text{SearchCircle}(\delta_1 + 2i\rho) \).
**End**

The next procedure \( \text{Search}(k) \) takes a positive integer \( k \geq 1 \) as input and instructs a robot to search the set of \( 2^k - 1 \) annuli in such a way that: a) the \( j^{th} \), \( 0 \leq j \leq 2k - 1 \), annulus has inner and outer radii \( \delta_{j,k} = 2^{-k+j} \) and \( \delta_{j,k+1} = 2^{-k+j+1} \) respectively, and, b) all points of this annulus are approached within a distance of \( \rho_{j,k} = 2^{-3k+2j-1} \). The idea is that the robot will search successive annuli of increasing inner and outer radii \( \delta_{j,k} \) and \( \delta_{j,k+1} \) under the assumption that its visibility radius is \( \rho_{j,k} \). The specific values of \( \delta_{j,k} \) and \( \rho_{j,k} \) are chosen such that the ratio \( \delta_{j,k}^2/\rho_{j,k} = 2^{k+1} \). At the end of the algorithm a robot is instructed to wait a rather specific amount of time only in order to simplify algebra later on. This procedure is formally described as Algorithm 7.
Algorithm 7 Search \((k)\)

Input: \(k > 0\) (integer);

Begin:
1:  for \(j = 0\) to \(2k - 1\) do
2:    SearchAnnulus \((2^{-k+j}, 2^{-k+j+1}, 2^{-3k+2j-1})\)
3:  Wait at initial position for a time \(3(\pi + 1)(2^k + 2^{-k})\).
:End

We are now ready to introduce our main search algorithm. This algorithm is formally presented as Algorithm 8. The algorithm repeatedly runs Search \((k)\) until the target is discovered. The idea is that the robot will search regions of the plane in such a way that in each round \(k\) a robot spends time proportional to the ratio \(d^2/r\) under the assumption that the values of \(d\) and \(r\) were such that the target was discovered on the round \(k\).

Algorithm 8

Begin:
1:  \(k \leftarrow 1\);
2:  repeat
3:    Perform Search \((k)\);
4:    \(k \leftarrow k + 1\);
5:  until Target found.
:End

We now claim the following:

**Theorem 8.2.1.** Algorithm 8 solves the search problem in time

\[
T(d, r) < 6(\pi + 1) \log \left( \frac{d^2}{r} \right) \frac{d^2}{r}.
\]

We will prove this theorem using a series of lemmas. We begin by demonstrating that the algorithm is correct.

**Lemma 8.2.2.** Algorithm 8 solves the search problem.

**Proof.** Assume that the target is at distance \(d\) and the robot has visibility \(r\). In the round \(k\) and sub-round \(j\) of Algorithm 7 a robot will search the entire disk of radius at least \(2^{-k+j+1}\) with a granularity of at most \(2^{-3k+2j-1}\). The robot must therefore find the target by the end of the round \(k\) of Algorithm 8 for which there exists an integer \(j\) such that \(0 \leq j \leq 2k - 1\), \(2^{-k+j+1} \geq d\), and \(2^{-3k+2j-1} \leq r\). It is not hard to confirm, for example, that \(k = \left\lfloor \log \frac{d^2}{r} \right\rfloor\) and \(j = \left\lfloor \log(d) \right\rfloor + \left\lfloor \log \frac{d^2}{r} \right\rfloor\) satisfy these constraints. \(\square\)
We now compute the running times of Algorithms 5 - 8.

**Lemma 8.2.3.** It takes time:

- \( 2(\pi + 1)\delta \) to complete SearchCircle(\( \delta \)).
- \( 2(\pi + 1) \left( 1 + \left\lceil \frac{\delta_2 - \delta_1}{2\rho} \right\rceil \right) \left( \delta_1 + \rho \left\lceil \frac{\delta_2 - \delta_1}{2\rho} \right\rceil \right) \) to complete SearchAnnulus(\( \delta_1, \delta_2, \rho \)).
- \( 3(\pi + 1)(k + 1) \cdot 2^{k+1} \) to complete Search(k).
- \( 3(\pi + 1)k \cdot 2^{k+2} \) to complete the first \( k \) rounds of Algorithm 8.

**Proof.** SearchCircle(\( \delta \)) clearly takes time \( 2(\pi + 1)\delta \). Let \( m = \left\lceil \frac{\delta_2 - \delta_1}{2\rho} \right\rceil \). The time to complete SearchAnnulus(\( \delta_1, \delta_2, \rho \)) is then

\[
\sum_{i=0}^{m} 2(\pi + 1)(\delta_1 + 2i\rho) = 2(\pi + 1) [(m + 1)\delta_1 + \rho(m + 1)m]
= 2(\pi + 1) (1 + m) (\delta_1 + \rho m).
\]

Now consider the algorithm Search(k). The time to complete the round \( j \) of this algorithm is just the time required to complete SearchAnnulus(\( \delta_{j,k}, \delta_{j,k+1}, \rho_{j,k} \)) where \( \delta_{j,k} = 2^{-k+j} \) and \( \rho_{j,k} = 2^{-3k+2j-1} \). Observe that \( \frac{\delta_{j+1,k} - \delta_{j,k}}{2\rho_{j,k}} = 2^{-k+j+1}2^{-2k-j} = 2^{2k-j} \). The time to complete one round of the algorithm is then \( 2(\pi + 1) \left( 1 + 2^{2k-j} \right) \left( 2^{-k+j} + 2^{-3k+2j-1} \cdot 2^{2k-j} \right) \) which simplifies to \( 3(\pi + 1)(2^{j-k} + 2^k) \). Since

\[
3(\pi + 1) \sum_{j=0}^{2k-1} (2^{j-k} + 2^k) = 3(\pi + 1) \left[ k2^{k+1} + 2^{-k}(2^{2k} - 1) \right]
\]

we can conclude that the time required to complete Search(k) is

\[
3(\pi + 1)(2^k + 2^{-k}) + 3(\pi + 1) \left[ k2^{k+1} + 2^{-k}(2^{2k} - 1) \right]
= 3(\pi + 1)(k + 1) \cdot 2^{k+1}.
\]

Finally, the time to complete the first \( k \) rounds of Algorithm 8 is \( 3(\pi + 1) \sum_{j=1}^{k} (j + 1)2^{j+1} = 3(\pi + 1)k \cdot 2^{k+2} \).

The next lemma gives a lower bound on the value of \( \frac{d^2}{r} \) assuming that the target was found on round \( k \) of Algorithm 8.

**Lemma 8.2.4.** If a robot finds the target on the round \( k \) of Algorithm 8 then \( \frac{d^2}{r} \geq 2^{k+1} \).
Proof. Assume that the target is found on round $k$ and sub-round $j$, $0 \leq j \leq 2k - 1$. In this case we know that $d \geq \delta_{j,k} = 2^{-k+j}$ and $r \leq \rho_{j,k} = 2^{-3k+2j-1}$. We therefore have that $\frac{d^2}{r} \geq \frac{\delta_{j,k}^2}{\rho_{j,k}} = 2^{k+1}$. 

The proof of Theorem 8.2.1 is now simple:

Proof. (Theorem 8.2.1) By Lemma 8.2.2 there must exist a round during which the robot finds the target. Assume that the target is found on the round $k$. Then, by Lemma 8.2.4, we have $\frac{d^2}{r} \geq 2^{k+1}$ and the time to find the target is upper-bounded by the time to complete the first $k$ rounds of Algorithm 8. Thus, if $T(d, r)$ is the rendezvous time, then by Lemma 8.2.3 we have that

$$T(d, r) \leq 3(\pi + 1)k \cdot 2^{k+2} \leq 6(\pi + 1) \left[ \log \left( \frac{d^2}{r} \right) - 1 \right] \frac{d^2}{r}.$$ 

8.3 Rendezvous with symmetric clocks

In this section we consider the rendezvous problem when the time units of the two robots are equal, i.e. $\tau = 1$. We will see that this problem is intimately related to search. Indeed, our goal is to demonstrate that the same Algorithm 8 used for search also solves the rendezvous problem (whenever a solution exists).

Theorem 8.3.1. Suppose that $\tau = 1$, and $v \neq 1$ or $\phi \neq 0$. Algorithm 8 solves the rendezvous problem in time

$$T(d, r, v, \phi, \chi) < \begin{cases} 6(\pi + 1) \log \left( \frac{d^2}{\mu r} \right) \frac{d^2}{\mu r}, & \chi = 1 \\ 6(\pi + 1) \log \left( \frac{d^2}{(1-v)r} \right) \frac{d^2}{(1-v)r}, & \chi = -1 \end{cases}$$

where $\mu = \sqrt{v^2 - 2v \cos(\phi) + 1}$.

To prove this theorem we will demonstrate that any algorithm which solves rendezvous for two robots with visibility $r$ and initial distance $d$ necessarily generates a corresponding algorithm which solves an instance of the search problem for a single robot with visibility $r$ and initial distance $d$ from an unknown target. The theorem will follow by analyzing the corresponding search algorithm when we use Algorithm 8 as our rendezvous algorithm.

We observe that we can interpret an algorithm for either the search or rendezvous problems as a single parametric trajectory $S(t)$ which specifies how the robots should move. In the case of rendezvous things are somewhat complicated by the fact that we have two robots which may not necessarily agree on their reference frames, i.e. they may have
different speeds, orientations, and/or chiralities. Thus, although only a single trajectory is specified by an algorithm, the two robots will actually follow two different trajectories during the execution of that algorithm – \( R \) will follow the trajectory \( S(t) \) and \( R' \) will follow a modified trajectory \( S'(t) \) which depends on the specific values of \( v, \phi, \) and \( \chi \). Nevertheless, the single trajectory \( S(t) \) suffices to completely describe a rendezvous algorithm. In the sequel we will refer to rendezvous and/or search algorithms by the trajectory \( S(t) \) they correspond to.

In the case of search, an algorithm \( S(t) \) solves the problem if there exists a time \( t \geq 0 \) for which \( |S(t) - \vec{d}| \leq r \) with \( \vec{d}, |\vec{d}| = d \), a vector pointing from the initial position of the robot to the target. In the case of rendezvous, \( S(t) \) solves the problem if there exists a time \( t \geq 0 \) for which \( |S(t) - S'(t) - \vec{d}| \leq r \), where, in this case, \( \vec{d}, |\vec{d}| = d \), represents a vector pointing from the initial position of \( R \) to that of \( R' \). If we reinterpret the trajectory \( S_c(t) = S(t) - S'(t) \) as a search algorithm, then it is easy to see that \( S(t) \) solves rendezvous for a given \( d \) and \( r \) if (and only if) \( S_c(t) \) solves search for the same \( d \) and \( r \). Moreover, if \( S(t) \) solves rendezvous at the time \( T \), then \( S_c(t) \) solves search at the same time \( T \). Thus, we say that the rendezvous algorithm \( S(t) \) induces an equivalent search algorithm \( S(t) - S'(t) \). This observation provides our strategy to prove Theorem 8.3.1 – we show that Algorithm \( 8 \) taken as a rendezvous algorithm, induces an equivalent search algorithm which still solves the search problem.

To begin we need to express the trajectory \( S'(t) \) of the robot \( R' \) in terms of \( v, \phi, \chi, \) and \( S(t) \).

**Lemma 8.3.2.** Consider a rendezvous algorithm \( S(t) \). Then the robot \( R \) will follow the trajectory \( S(t) \) and the robot \( R' \) will follow the trajectory \( S'(t) + \vec{d} \) where

\[
S'(t) = \begin{bmatrix}
v \cos(\phi) & -v \chi \sin(\phi) \\
v \sin(\phi) & v \chi \cos(\phi)
\end{bmatrix} S(t)
\]

and \( \vec{d} \) points from the initial position of \( R \) to the initial position of \( R' \).

**Proof.** If the robots have different chiralities (\( \chi = -1 \)) then they will disagree on the \(+y\) direction. This implies that \( R' \) will move along a trajectory that is a reflection about the \( x \)-axis of the trajectory followed by \( R \). If the orientation of \( R' \) is \( \phi \), then its trajectory will be rotated by an angle \( \phi \) with respect to that of \( R \). If the speed of \( R' \) is \( v < 1 \) then it will travel a smaller distance in the same time interval as compared to \( v \) and its trajectory will thus be scaled by a factor of \( v \). Combining all of these transformations together we find that

\[
S'(t) = v \begin{bmatrix}
\cos(\phi) & -\sin(\phi) \\
\sin(\phi) & \cos(\phi)
\end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \chi \end{bmatrix} S(t).
\]

Now that we know the trajectory of \( R' \) we can compute the equivalent search trajectory.
\[ S_\circ(t) = S(t) - S'(t) \]

as

\[ S_\circ(t) = S(t) - S'(t) = \begin{bmatrix}
1 - v \cos(\phi) & v \chi \sin(\phi) \\
-v \sin(\phi) & 1 - v \chi \cos(\phi)
\end{bmatrix} S(t). \]

Let \( T_\circ = \begin{bmatrix}
1 - v \cos(\phi) & v \chi \sin(\phi) \\
-v \sin(\phi) & 1 - v \chi \cos(\phi)
\end{bmatrix} \). In the next lemma we rewrite \( T_\circ \) in a more convenient form.

**Lemma 8.3.3.** The matrix \( T_\circ \) can be factored as \( T_\circ = \Phi T'_\circ \) where \( \Phi \) is a rotation matrix, \( T'_\circ = \begin{bmatrix}
\mu & \frac{-1-\chi v \sin(\phi)}{\mu} \\
0 & \frac{1+\chi v^2-(1+\chi) v \cos(\phi)+1}{\mu}
\end{bmatrix} \) and \( \mu = \sqrt{v^2 - 2v \cos(\phi)} + 1 \).

**Proof.** What we want amounts to a QR-factorization. This is a well known operation and so we just quote the result:

\[
T_\circ = \frac{1}{\mu} \begin{bmatrix}
1 - v \cos(\phi) & v \sin(\phi) \\
-v \sin(\phi) & 1 - v \cos(\phi)
\end{bmatrix} \begin{bmatrix}
\mu & -\frac{(1-\chi)v \sin(\phi)}{\mu} \\
0 & \frac{1+\chi v^2-(1+\chi) v \cos(\phi)}{\mu}
\end{bmatrix}
\]

where we have set \( \mu = \sqrt{v^2 - 2v \cos(\phi)} + 1 \). It is easy to confirm that

\[
\frac{1}{\mu} \begin{bmatrix}
1 - v \cos(\phi) & v \sin(\phi) \\
-v \sin(\phi) & 1 - v \cos(\phi)
\end{bmatrix}
\]

is an orthogonal matrix with determinant 1. \( \square \)

Since rotations do not scale distances, the condition \( |T_\circ S(t) - \vec{d}| \leq r \) is equivalent to the condition that \( |T'_\circ S(t) - \vec{d}| \leq r \). It will be easier to analyze this rotated version of the problem and thus, in the sequel, we will use the following definition for an equivalent search trajectory:

**Definition 8.3.4.** The equivalent search trajectory induced by the rendezvous trajectory \( S(t) \) is the trajectory \( S_\circ(t) \) where

\[
S_\circ(t) = T_\circ S(t) = \begin{bmatrix}
\mu & \frac{-1-\chi v \sin(\phi)}{\mu} \\
0 & \frac{1+\chi v^2-(1+\chi) v \cos(\phi)}{\mu}
\end{bmatrix} S(t)
\]

and \( \mu = \sqrt{v^2 - 2v \cos(\phi)} + 1 \).

At this point it will be easier to consider the cases that \( \chi = \pm 1 \) separately. We begin with the case that \( \chi = 1 \):
Lemma 8.3.5. Suppose that $\chi = 1$, and that $v \neq 1$ or $\phi \neq 0$. Algorithm 8 solves rendezvous in time

$$T(d, r, v, \phi) < 6(\pi + 1) \log \left( \frac{d^2}{\mu r} \right) \frac{d^2}{\mu r},$$

where $\mu = \sqrt{v^2 - 2v \cos(\phi) + 1}$.

Proof. We first observe that $T_\circ$ takes on a particularly simple form when $\chi = 1$. We find that

$$T_\circ = \begin{bmatrix} \mu & -(1-\chi)v \sin(\phi) \\ 0 & 1+\chi v^2-(1+\chi)\nu \cos(\phi) \end{bmatrix} = \begin{bmatrix} \mu & 0 \\ 0 & \mu \end{bmatrix}$$

with the result that $S_\circ(t) = \mu S(t)$. Thus $S_\circ(t)$ is the trajectory of a robot with speed $\mu$ performing Algorithm 8. We need to demonstrate that there exists a time $t \geq 0$ for which $|S(t) - \ddot{d}| \leq r$. Multiplying this inequality by $\frac{1}{\mu}$ gives the equivalent condition $|\ddot{S}(t) - \frac{d}{\mu}| \leq \frac{r}{\mu}$, and we already know that there exists a time for which this is satisfied when $S(t)$ is given by Algorithm 8. We can therefore conclude that Algorithm 8 solves rendezvous. To bound the rendezvous time we can directly use the results of Theorem 8.2.1 with $d$ and $r$ replaced with $\frac{d}{\mu}$ and $\frac{r}{\mu}$ respectively.

Now we consider the case that the robots have opposite chiralities. We begin by revisiting what it means for an algorithm to solve the search problem. To this end, assume that we have an algorithm $S(t)$ that solves search in time at most $T$ when the robot has visibility $r$ and the target is at a distance $d$. What this tells us is that all possible positions of the target at distance $d$ from the initial position of the robot can be reached in time at most $T$. In particular, if the target is located at $\ddot{d}$, then there exists a time $0 \leq t \leq T$ for which $|S(t) - \ddot{d}| \leq r$. Since the distance of closest approach of the robot and the target occurs when the robot is located on the line defined by the unit vector $\hat{d} = \ddot{d}/d$, we can conclude that there exists a time $0 \leq t \leq T$ for which the length of the projection of $S(t) - \ddot{d}$ onto the line defined by $\hat{d} = \ddot{d}/d$ is at most $r$. Thus, if $S(t)$ solves search, then there exists a time $0 \leq t \leq T$ for which $(S(t) - \ddot{d}) \cdot \hat{d} \leq r$. With this observation in hand, we can proceed to prove the following:

Lemma 8.3.6. When $\chi = -1$ Algorithm 8 solves rendezvous in time

$$T(d, r, v) < 6(\pi + 1) \log \left( \frac{d^2}{(1-v)r} \right) \frac{d^2}{(1-v)r}.$$ 

Proof. When $\chi = -1$ the matrix $T_\circ$ simplifies to

$$T_\circ = \begin{bmatrix} \mu & -2v \sin(\phi) \\ 0 & \frac{1-v^2}{\mu} \end{bmatrix}.$$ 

227
We want to show that there exists a time \( t \geq 0 \) for which \((S_o(t) - \vec{d}) \cdot \hat{d} \leq r\) is satisfied. Observe that

\[
(S_o(t) - \vec{d}) \cdot \hat{d} = (T_o S(t)) \cdot \hat{d} - d = S^T(t)T_o \hat{d} - d
\]

\[
= S(t) \cdot (T_o^T \hat{d}) - d = |T_o^T \hat{d}| \left[ S(t) \cdot \frac{T_o \hat{d}}{|T_o \hat{d}|} - \frac{d}{|T_o \hat{d}|} \right]
\]

where \( \hat{d}' = \frac{T_o \hat{d}}{|T_o \hat{d}|} \) and \( \vec{d}' = \frac{d}{|T_o \hat{d}|} \cdot \hat{d}' \). Thus, the condition that \((S_o(t) - \vec{d}) \cdot \hat{d} \leq r\) is satisfied for some \( t \geq 0 \) is equivalent to the condition that \((S(t) - \vec{d}' \cdot \hat{d}') \leq r' \) with \( r' = \frac{r}{|T_o \hat{d}|} \). As we know that Algorithm 8 indeed does satisfy this equality for some \( t \geq 0 \) we can conclude that Algorithm 8 solves rendezvous. To bound the time of rendezvous we need to replace \( d \) and \( r \) with \( \frac{d}{|T_o \hat{d}|} \) and \( \frac{r}{|T_o \hat{d}|} \) in the bound of Theorem 8.2.1. To complete the proof we first need to compute the value of \( |T_o^T \hat{d}| \) and then maximize the resulting time bound over all possible positions of the target. As we only need to compute the worst case position of \( \hat{d} \) in order to bound the rendezvous time, this justifies that we set \( \vec{d}' = (0, 1) \) and instead maximize the result over all possible values of \( \phi \). In this case it is easy to confirm that \( |T_o^T \hat{d}| = \frac{1-v^2}{\mu} \).

Since the search time scales with \( \frac{d^2}{r} \) we want to maximize the quantity \( \frac{\mu}{1-v^2} \) with respect to \( \phi \). This is equivalent to maximizing \( \mu = \sqrt{v^2 - 2v \cos(\phi) + 1} \). Clearly, the maximum value of \( \mu \) is \( \sqrt{v^2 + 2v + 1} = 1 + v \). The bound in the lemma then follows from Theorem 8.2.1 and the fact that \( \frac{1+v}{1-v^2} = \frac{1}{1-v} \).

### 8.4 Rendezvous with asymmetric clocks

We now consider rendezvous for the case that the robots have different time units (\( \tau \neq 1 \)). Without loss of generality we will take \( \tau < 1 \). We will begin with the assumption that the robots have the same speeds (\( v = 1 \)) and then extend our results to cover the case that the speeds of the robots may also be different. Our goal is to demonstrate that rendezvous is always feasible. The main result of this section follows:

**Theorem 8.4.1.** For any fixed \( \tau < 1 \) there exists an algorithm which solves rendezvous in finite time.

We will prove this theorem constructively and give an algorithm which solves the rendezvous problem in finite time provided that \( \tau < 1 \). This algorithm will be constructed using the algorithms of Section 8.2. In the sequel we describe this algorithm and then analyze its running time.
An outline and the idea of our rendezvous algorithm is as follows. The algorithm proceeds in a series of rounds with each round composed of equal length inactive and active phases. In an inactive phase a robot will remain stationary at its initial position. In the active phase of round $n$ a robot will (essentially) perform the first $n$ rounds of Algorithm 8. We observe that this implies that there will be a round $n_*$ of the algorithm for which, say, the robot $R$ would find $R'$ if $R'$ were stationary at its initial position. Moreover, this will be true for every round $n \geq n_*$. We will show that, due to the differing time units of the robots, it will happen that the active and inactive phases of the robots overlap and that the length of the overlap interval grows without bound. Thus, there will exist a round $n \geq n_*$ during which the active and inactive phases of the robots overlap long enough that $R$ is able to find $R'$ waiting at its initial location.

Our rendezvous algorithm is formally described as Algorithm 11. In each active phase a robot will run the procedures $\text{SearchAll}(n)$ and $\text{SearchAllRev}(n)$ which are formally described as Algorithms 9 and 10 respectively. The procedure $\text{SearchAll}(n)$ is identical to Algorithm 8 except that it always terminates after $n$ rounds. The procedure $\text{SearchAllRev}(n)$ is identical to $\text{SearchAll}(n)$ except that it is run in reverse, i.e. it begins on the $n^{th}$ round and ends on the round 1. Figures 8.1 and 8.2 illustrate the structure of each round of the algorithm.

---

**Algorithm 9 SearchAll$(n)$**

**Input:** $n > 0$ (integer);

**Begin:**

1. for $k = 1$ to $n$ do Perform Search$(k)$.

:End

---

**Algorithm 10 SearchAllRev$(n)$**

**Input:** $n > 0$ (integer);

**Begin:**

1. for $k = n$ to 1 do Perform Search$(k)$.

:End
Algorithm 11

Begin:
1: \( n \leftarrow 1 \)
2: repeat
3: \( \text{Wait at initial position for a time } 2S(n) \).
4: \( \text{Perform SearchAll}(n) \) then SearchAllRev(n).
5: \( n = n + 1 \).
6: until Rendezvous occurs

:End

We begin our proof of Theorem 8.4.1 with the following lemma which gives the time at which the inactive and active phases of Algorithm 11 begin.

Lemma 8.4.2. The \( n^{th} \) inactive phase of Algorithm 11 begins at the time \( I(n) = 24(\pi + 1)[(2n - 4) \cdot 2^n + 4] \) and the \( n^{th} \) active phase begins at the time \( A(n) = 24(\pi + 1)[(3n - 4) \cdot 2^n + 4] \).

Proof. It is easy to see that each round of Algorithm 11 lasts time \( 4S(n) \) where \( S(n) \) is the time it takes to complete SearchAll(\( n \)). Using the results of Lemma 8.2.3 we have

\[ S(n) = 12(\pi + 1)n \cdot 2^n. \tag{8.1} \]

The beginning of the \( n^{th} \) inactive phase is just the total time required to complete the first \( n - 1 \) rounds of the algorithm. Thus

\[ I(n) = 4 \sum_{k=1}^{n-1} S(k) = 48(\pi + 1) \sum_{k=1}^{n-1} k2^k = 24(\pi + 1)[(2n - 4)2^n + 4]. \]

The \( n^{th} \) active phase begins when the \( n^{th} \) inactive phase ends. We therefore have \( A(n) = I(n) + 2S(n) = 24(\pi + 1)[(3n - 4) \cdot 2^n + 4] \).

\[ \square \]

230
After some manipulation we arrive to the equivalent condition

\[ \text{SearchAllRev} \]

Since the robots run both of the procedures \text{SearchAll} and \text{SearchAllRev} during an inactive phase of \( R' \) where \( k \) is the first round of \text{SearchAll} \( \cdot \) during which \( R \) would find \( R' \) if \( R' \) were stationary at its initial location. Since the robots run both of the procedures \text{SearchAll} \( n_* \) and \text{SearchAllRev} \( n_* \) the inactive and active phases of the robots can overlap in the two ways illustrated in Figure 8.3.

In the next two lemmas we show under what circumstances the robots will find themselves inactive and active phases of the robots can overlap in the two ways illustrated in Figure 8.3.

![Figure 8.3](image)

Figure 8.3: Illustration of how the active phase of \( R \) can overlap with the inactive phase of \( R' \). The intervals of overlap are shaded, the active/inactive phases of \( R \) are indicated by solid lines, and those of \( R' \) are indicated by dashed lines.

To reiterate, we need to show that the overlap between the active and inactive phases of the algorithm for which \( R \) is able to finish the first (resp. last) \( k \) rounds of \text{SearchAll} \( n_* \) (resp. \text{SearchAllRev} \( n_* \)) during an inactive phase of \( R' \) where \( k \) is the first round of \text{SearchAll} \( \cdot \) during which \( R \) would find \( R' \) if \( R' \) were stationary at its initial location.

Lemma 8.4.3. If \[ \frac{k}{(k+1+a)2^{a+1}} \leq \tau \leq \frac{3}{2} \cdot \frac{k}{(k+1+a)2^{a+1}} \] and \( k \geq 2(a + 1) \) for some integer \( a \geq 0 \) then the \( k^{th} \) active phase of \( R \) overlaps with the \((k + 1 + a)^{th}\) inactive phase of \( R' \) by an amount \( \tau \cdot A(k + 1 + a) - A(k) \).

**Proof.** Referring to Figure 8.3(a), one can see that in order for the \( k^{th} \) active phase of \( R \) to overlap with the \((k + a)^{th}\) inactive phase of \( R' \) by an amount \( \tau \cdot A(k+1+a) - A(k) \) we need to satisfy \( \tau \cdot I(k+1+a) \leq A(k) \leq \tau \cdot A(k+a) \). Rearranging these inequalities leads to the equivalent condition \[ \frac{A(k)}{A(k+1+a)} \leq \frac{ \tau \cdot I(k+1+a) }{A(k+1+a)} \leq \frac{A(k)}{I(k+1+a)} \leq \frac{3}{2} \cdot \frac{k}{(k+1+a)2^{a+1}} \]. Consider the latter inequality first. A simple rearrangement of this inequality yields \( (k+1+a)2^{a+1}A(k) \leq kA(k+1+a) \). Substitution of the expression derived in Lemma 8.4.2 for \( A(k) \) gives

\[ (k+1+a)2^{a+1}[(3k - 4)2^k + 4] \leq k[3(k + 1 + a) - 4]2^{k+1+a} + 4 \].

After some manipulation we arrive to the equivalent condition \( \frac{a+1}{k} > \frac{1 - 2^{-a-1}}{2^{a-1}} \). It is easy to confirm that this inequality is satisfied for all \( a \geq 0 \) and \( k \geq 2(a + 1) \).

Now consider the inequality \[ \frac{A(k)}{I(k+1+a)} \geq \frac{3}{2} \cdot \frac{k}{(k+1+a)2^{a+1}} \]. Rearranging this inequality and substitution of the expressions for \( A(k) \) and \( I(k) \) from Lemma 8.4.2 yields

\[ 2(k+1+a)2^{a+1}[(3k - 4)2^k + 4] \leq 3k[2(k + 1 + a) - 4]2^{k+1+a} + 4 \].

231
After some manipulation this reduces to \( \frac{k}{a+1} \geq \frac{2}{1+3 \cdot 2^{\frac{n-a}{a+1}}} \). It is easy to see that this inequality is also satisfied whenever \( a \geq 0 \) and \( k \geq 2(a+1) \).

**Lemma 8.4.4.** If \( \frac{2}{3} \cdot \frac{k}{(k+a)2^a} \leq \tau \leq \frac{k}{(k+1+a)2^a} \) and \( k \geq 2(a+1) \) for some integer \( a \geq 0 \) then the \((k-1)\)st active phase of \( \mathcal{R} \) overlaps with the \((k+a)\)th inactive phase of \( \mathcal{R}' \) by an amount \( I(k) - \tau \cdot I(k+a) \).

**Proof.** Referring to Figure 8.3b, in order for the \((k-1)\)st active phase of \( \mathcal{R} \) to overlap with the \((k+a)\)th inactive phase of \( \mathcal{R}' \) by an amount \( I(k) - \tau \cdot I(k+a) \) we need to satisfy \( \tau \cdot I(k+a) \leq I(k) \leq \tau \cdot A(k+a) \). Rearranging these inequalities leads to the equivalent condition \( \frac{I(k)}{A(k+a)} \leq \tau \leq \frac{I(k)}{I(k+a)} \). Thus, in order to prove the lemma we need to demonstrate that \( \frac{I(k)}{A(k+a)} \leq \frac{2}{3} \cdot \frac{k}{(k+a)2^a} \) and \( \frac{I(k)}{I(k+a)} \geq \frac{k}{(k+1+a)2^a} \). Consider the latter inequality first. Rearrangement and substitution of the expressions for \( A(k) \) and \( I(k) \) gives

\[
3(k+a)2^a[(2k-4)2^k + 4] \leq 2k\{3(k+a) - 4\}2^{k+a} + 4\}.
\]

After some manipulation we get \( 3a(2k-1) + k(2^k - 3 + 2^{1-a}) \geq 0 \). It is easy to confirm that this inequality is satisfied for all \( a \geq 0 \) and \( k \geq 2(a+1) \).

Now consider the inequality \( \frac{I(k)}{I(k+a)} \geq \frac{k}{(k+1+a)2^a} \). Rearrangement and substitution of the expression for \( I(k) \) yields

\[
(k+1+a)2^a[(2k-4)2^k + 4] \leq k\{2(k+a) - 4\}2^{k+a} + 4\}\]

After some manipulation this reduces to \( \frac{k}{a+1} \geq \frac{2}{1+3 \cdot 2^{\frac{n-a}{a+1}}} \). It is easy to see that this inequality is also satisfied whenever \( a \geq 0 \) and \( k \geq 2(a+1) \). \(\)

In the next two lemmas we determine in which round the robots will rendezvous under the assumption that the conditions of the previous two lemmas are met.

**Lemma 8.4.5.** Assume that \( \mathcal{R} \) would find a stationary target located at the initial position of \( \mathcal{R}' \) on round \( n \) of Algorithm 11. If there exists two integers \( a \geq 0 \) and \( k_0 \geq 2(a+1) \) such that \( \tau \) satisfies \( \frac{k}{(k+1+a)2^a} \leq \tau \leq \frac{k}{(k+1+a)2^a} \) for all rounds \( k \geq k_0 \) then the robots will rendezvous by the end of the round \( k_* = n + \left\lceil \log \left( \frac{a}{a+1} \right) \right\rceil \) of Algorithm 11.

**Proof.** Assume that integers \( a \geq 0 \) and \( k_0 \geq 2(a+1) \) exist such that \( \tau \) satisfies the condition of the lemma. This implies that \( \tau \) must satisfy

\[
\frac{1}{2} \cdot 2^{-a} \leq \tau \leq \frac{3}{4} \cdot \frac{k_0}{k_0 + 1 + a} \cdot 2^{-a}
\]

since \( \frac{k}{k+1+a} \) is increasing with \( k \) and \( \frac{k}{k+1+a} < 1 \).
By Lemma 8.4.3 the $k^{th}$ active phase of $R$ overlaps with the $(k+1+a)^{th}$ inactive phase of $R'$ by an amount $\tau \cdot A(k+1+a) - A(k)$ for all $k \geq k_0$. The robots are guaranteed to rendezvous on or before the first round $k_\ast \geq k_0$ for which this overlap is larger than $S(n)$. We observe that

$$\tau A(k+1+a) - A(k) = 24(\pi + 1) \left[ \frac{3(a+1)}{3(a+1)} \left( \frac{n}{2} \cdot 2^n + 4 \right) \right] - \frac{3(a+1)}{3(a+1)}2k - 4$$

For $\tau$ satisfying (8.2) we can write

$$\tau \cdot A(k+1+a) - A(k) \geq 24(\pi + 1) \left[ 3(a+1)2^k - 4 \right] \geq 24(\pi + 1) \left[ 3(a+1)2^k - 4 \right].$$

Thus we will have $\tau \cdot A(k+1+a) - A(k) \geq S(n)$ when $24(\pi + 1) \left[ 3(a+1) \cdot 2^k - 4 \right] \geq S(n)$. Using the expression (8.1) for $S(n)$ we can rewrite the above inequality as $3(a+1) \cdot 2^k - 4 \geq \frac{n}{2} \cdot 2^n$ which simplifies to $k \geq \log \left[ \frac{1}{3(a+1)} \left( \frac{n}{2} \cdot 2^n + 4 \right) \right]$. The robots will therefore rendezvous by the end of the round

$$k_\ast = \left\lceil \log \left[ \frac{1}{3(a+1)} \left( \frac{n}{2} \cdot 2^n + 4 \right) \right] \right\rceil < n + \left\lceil \log \left( \frac{n}{a+1} \right) \right\rceil.$$

Lemma 8.4.6. Assume that $R$ would find a stationary target located at the initial position of $R'$ on round $n$ of Algorithm 11. If there exists two integers $a \geq 0$ and $k_0 \geq 2(a+1)$ such that $\tau$ satisfies $\frac{2}{3} \cdot \frac{k}{(k+1+a)^2} \leq \tau \leq \frac{k}{(k+1+a)^2}$ for all $k \geq k_0$ then the robots will rendezvous by the end of the round $k_\ast = n + \left\lceil \log(n) + \log \left( 1 + \frac{k_0}{a+1} \right) \right\rceil$ of Algorithm 11.

Proof. Assume that integers $a \geq 0$ and $k_0 \geq 2(a+1)$ exist such that $\tau$ satisfies the condition of the lemma. This implies that $\tau$ must satisfy

$$\frac{2}{3} \cdot 2^{-a} \leq \tau \leq \frac{k_0}{k_0 + 1 + a} \cdot 2^{-a} \quad (8.3)$$

since $\frac{k}{k+1+a} < 1$ and is increasing with $k$.

Assume $d$ and $r$ are chosen such that $R$ would find a stationary target at the initial position of $R'$ on round $n$ of Algorithm 11. By Lemma 8.4.4 the $(k-1)^{st}$ active phase of $R$ overlaps with the $(k+a)^{th}$ inactive phase of $R'$ by an amount $I(k) - \tau \cdot I(k+a)$ for all
\( k \geq k_0 \). The robots are guaranteed to rendezvous on or before the first round \( k_a \geq k_0 \) for which this overlap is larger than \( S(n) \). We observe that
\[
I(k) - \tau \cdot I(k + a) \\
= 24(\pi + 1) \left[ (2k - 4)2^k + 4 - \tau[(2(k + a) - 4)2^{k+a} + 4] \right] \\
= 24(\pi + 1) \left[ (2k - 4)2^k[1 - \tau 2^a] + 4(1 - \tau) - 2a\tau 2^{k+a} \right].
\]

Let \( \gamma = \frac{k_0}{k_0 + 1+a} \). Then, for \( \tau \) satisfying (8.3), we can write
\[
I(k) - \tau \cdot I(k + a) \\
\geq 24(\pi + 1) \left[ (2k - 4)2^k(1 - \gamma) + 4(1 - \gamma 2^{-a}) - 2a\gamma 2^k \right] \\
> 24(\pi + 1) \left[ (2k - 4)2^k(1 - \gamma) - 2a\gamma 2^k \right] \\
= 48(\pi + 1)[(k - 2)(1 - \gamma) - a\gamma]2^k
\]

Thus we will have \( I(k) - \tau \cdot I(k + a) \geq S(n) \) when \( [(k - 2)(1 - \gamma) - a\gamma]2^k \geq \frac{n}{4} \cdot 2^n \). Let \( x = (k - 2)(1 - \gamma) - a\gamma \) such that \( k = \frac{x + a\gamma}{1 - \gamma} + 2 = \frac{x + (a-2)\gamma + 2}{1 - \gamma} \). We may write
\[
[(k - 2)(1 - \gamma) - a\gamma]2^k = x \cdot \left( \frac{2}{1 - \gamma} \right)^x \cdot \left( \frac{2}{1 - \gamma} \right)^{(a-2)\gamma + 2} \\
= x \cdot e^{\ln(2)x} \cdot \left( \frac{2}{1 - \gamma} \right)^{(a-2)\gamma + 2}.
\]

Then, in terms of \( x \), we need to satisfy,
\[
x \cdot e^{\ln(2)x} \cdot \left( \frac{2}{1 - \gamma} \right)^{(a-2)\gamma + 2} \geq \frac{n}{4} \cdot 2^n
\]

or
\[
\ln(2)x \cdot e^{\ln(2)x} \geq \frac{\ln(2)n}{4(1 - \gamma)} \cdot 2^n \cdot \left( \frac{2}{1 - \gamma} \right)^{(a-2)\gamma - 2}.
\]

In the case of equality, the above has the form \( ze^z = y \) and this has the solution \( z = W(y) \) where \( W(\cdot) \) is the W-Lambert function. We therefore have that
\[
\frac{x \ln(2)}{1 - \gamma} = W \left[ \frac{\ln(2)n}{4(1 - \gamma)} \cdot 2^n \cdot \left( \frac{2}{1 - \gamma} \right)^{(a-2)\gamma - 2} \right].
\]

Expressing this again in terms of \( k \) we find that \( k \) must satisfy
\[
k \geq 2 + \frac{\alpha \gamma}{1 - \gamma} + \frac{1}{\ln(2)} \left[ \frac{\ln(2)n}{4(1 - \gamma)} \cdot 2^n \cdot \left( \frac{2}{1 - \gamma} \right)^{(a-2)\gamma - 2} \right].
\]
which implies that
\[
k_\ast = 2 + \left\lfloor \frac{a\gamma}{1 - \gamma} + \frac{1}{\ln(2)} W \left[ \frac{\ln(2)n}{4(1 - \gamma)} \cdot 2^n \cdot (2^{1 - \gamma})^{-(a - 2)\gamma - 2} \right] \right\rfloor.
\]

Now, \( W(x) \) behaves asymptotically as \( \ln(x) - \ln(\ln(x)) \) \([119]\) and we may therefore simplify this expression to
\[
k_\ast \leq 2 + \left\lfloor \frac{a\gamma}{1 - \gamma} + \frac{1}{\ln(2)} \ln \left[ \frac{\ln(2)n}{4(1 - \gamma)} \cdot 2^n \cdot (2^{1 - \gamma})^{-(a - 2)\gamma - 2} \right] \right\rfloor < n + \left\lfloor \frac{1}{\ln 2} \ln \left( \frac{n}{1 - \gamma} \right) \right\rfloor = n + \left\lfloor \log \left( \frac{n}{1 - \gamma} \right) \right\rfloor.
\]

Finally, rewriting the above in terms of \( k_0 \) leads us to the desired result.

We now determine explicitly an upper bound on the round in which the robots rendezvous.

**Lemma 8.4.7.** Parameterize \( \tau \) as \( \tau = t \cdot 2^{-a} \) for an integer \( a \geq 0 \) and a real \( t \in \left( \frac{1}{2}, 1 \right) \). Then, if \( \frac{1}{2} \leq t \leq \frac{2}{3} \) the robots will rendezvous before the end of the round \( k_\ast = \max \left\{ 8(a + 1), n + \left\lfloor \log \left( \frac{n}{a + 1} \right) \right\rfloor \right\} \) and otherwise, for \( \frac{2}{3} < t < 1 \), the robots will rendezvous before the end of the round \( k_\ast = \max \left\{ (a + 1)\frac{t}{1 - t}, n + \left\lfloor \log \left( \frac{n}{1 - t} \right) \right\rfloor \right\} \).

**Proof.** First note that we may always write \( \tau \) uniquely as \( t \cdot 2^{-a} \) by taking \( a = \lfloor -\log_2(\tau) \rfloor - 1 \) and \( t = \frac{1}{2} \) if \( \tau \) is a power of two, and otherwise taking \( a = \lfloor -\log_2(\tau) \rfloor \) and \( t = \tau \cdot 2^a \).

First assume that \( \frac{1}{2} \leq t \leq \frac{2}{3} \). Then the first part of the lemma will follow from Lemma 8.4.5 if we can find a \( k_0 \) such that \( \tau = t \cdot 2^{-a} \leq \frac{3}{4} \cdot \frac{k_0}{k_0 + 1 + a} \cdot 2^{-a} \). Solving this inequality we find that \( k_0 \) is given by \( k_0 \geq 4(a + 1)\frac{t}{3 - 4t} \). For \( t \in \left[ \frac{1}{2}, \frac{2}{3} \right] \) one can easily confirm that \( k_0 \) must be at least \( 8(a + 1) \) in order to guarantee that \( k_0 \geq 4(a + 1)\frac{t}{3 - 4t} \).

Thus, if \( n + \left\lfloor \log \left( \frac{n}{a + 1} \right) \right\rfloor \geq 8(a + 1) \) the robots will rendezvous before the end of the round \( n + \left\lfloor \log \left( \frac{n}{a + 1} \right) \right\rfloor \) otherwise the robots will rendezvous at the end of the round \( 8(a + 1) \).

Now assume that \( \frac{2}{3} < t < 1 \). Then the second part of the lemma will follow from Lemma 8.4.6 if we can determine a \( k_0 \) such that \( \tau \leq \frac{k_0}{k_0 + 1 + a} \cdot 2^{-a} \), or, equivalently, \( t \leq \frac{k_0}{k_0 + 1 + a} \). Solving this inequality we find that \( k_0 = \frac{(a + 1)\frac{t}{1 - t}}{1 - \frac{k_0}{a + 1}} \). Thus, if \( n + \left\lfloor \log \left( 1 + \frac{k_0}{a + 1} \right) \right\rfloor \geq \frac{t}{1 - t}(a + 1) \) the robots will rendezvous before the end of the round \( n + \left\lfloor \log \left( 1 + \frac{k_0}{a + 1} \right) \right\rfloor = n + \left\lfloor \log \left( \frac{1}{1 - t} \right) \right\rfloor \), and otherwise the robots will rendezvous by the end of the round \( (a + 1)\frac{t}{1 - t} \).

At this point we can almost prove Theorem 8.4.1 For now we will prove the following weaker statement:
Lemma 8.4.8. For any fixed $\tau < 1$ there exists an algorithm which solves rendezvous in finite time if $v = 1$.

Proof. Write $\tau = t \cdot 2^{-a}$ where $t$ and $a$ are defined as in the previous lemma. Assume that $R$ would find a stationary target at the initial position of on round $n$ of Algorithm 11. Then $\frac{d^2}{r} \geq 2^{n+1}$ and also $n \leq \log \left( \frac{d^2}{r} \right) - 1$ (see Lemma 8.2.4). The robots will rendezvous by the round $k_\ast$ of Algorithm 11 where $k_\ast$ is given in Lemma 8.4.7. The total time to complete $k_\ast$ rounds is $I(k_\ast) = 24(\pi + 1)[(2k_\ast - 4)2^{k_\ast} + 4]$. Thus, the rendezvous time of the algorithm is $T(d, r, \tau) < 24(\pi + 1)[(2k_\ast - 4)2^{k_\ast} + 4]$. We claim that, for any fixed value of $\tau < 1$, this bound is finite. Indeed, observe that $k_\ast$ is only infinite if $t = 1$. However, $t$ is strictly smaller than one if $\tau < 1$. We have thus demonstrated that Algorithm 11 solves rendezvous in finite time if $\tau < 1$ and $v = 1$.

We now extend this result to cover the case that the robots have different speeds. This will conclude the proof of Theorem 8.4.1.

Proof. (Theorem 8.4.1) Observe that the speed of a robot does not affect the times at which its active and inactive phases begin and/or end. This implies that everything up to and including Lemma 8.4.7 applies directly to the case that $v \neq 1$. This already implies that there will be a finite number of rounds until the robots rendezvous. A time bound can be derived in a similar manner to the derivation of the previous lemma.

We note that we have not needed to consider the chirality and/or orientations of the robots in our analysis due to the fact that our proof relied on one robot finding the other while the other was stationary. Thus Theorem 8.4.1 applies regardless of the robots’ relative orientations and/or chiralities. Moreover, one can use the same techniques of Section 8.3 to show that Algorithm 11 will also solve the rendezvous problem if the robots have different speeds but equal time units. We can therefore conclude this section with the following theorem:

Theorem 8.4.9. Algorithm 11 solves the rendezvous problem in finite time if $\tau \neq 1$ or $v \neq 1$, or $\chi = 1$ and $0 < \phi < 2\pi$.

8.5 Conclusions

In this paper we addressed the fundamental problem of feasibility of deterministic rendezvous in the infinite Euclidean plane for two robots that have minimal or no knowledge of the respective capabilities and “measuring instruments” of themselves and each other. We examined the impact on feasibility of rendezvous that the four parameters of speed, clock, distance, chirality have and presented and analyzed specific algorithms with good performance guarantees on the rendezvous time.
Our approach not only provides a surprising twist to the well-known rendezvous problem on the infinite Euclidean plane, but possibly it also creates interesting avenues for future research. In addition to tightening our bounds, there are several interesting questions that could be considered concerning the rendezvous problem. These include rendezvous for robots that may have alternative capabilities (e.g., variable speed), more general terrains with and without obstacles, and rendezvous in higher dimensional space. In addition, it would be challenging to solve deterministic gathering for multiple robots in this setting of “minimal knowledge”.

References


Chapter 9

Gathering with faults

The previous two chapters concerned the symmetric rendezvous of two agents. In this chapter an asymmetric rendezvous/gathering problem is studied wherein a large group of robots want to converge on a single location in the plane in the presence of a group of faulty agents that want to prevent the gathering from happening. The included paper “Gathering in the Plane of Location-Aware Robots in the Presence of Spies” appears in Theoretical Computer Science and is a journal version of a conference paper appearing in the proceedings of SIROCCO2018.
Gathering in the Plane of Location-Aware Robots in the Presence of Spies

Jurek Czyzowicz\textsuperscript{1}, Ryan Killick\textsuperscript{2}, Evangelos Kranakis\textsuperscript{2}, Danny Krizanc\textsuperscript{3}, Oscar Morales-Ponce\textsuperscript{4}

\textsuperscript{1}Université du Québec en Outaouais, Gatineau, Québec, Canada
\textsuperscript{2}School of Computer Science, Carleton University, Ottawa, Ontario, Canada
\textsuperscript{3}Department of Mathematics & Comp. Sci., Wesleyan University, Middletown, CT, USA
\textsuperscript{4}Dep. of Computer Engineering and Computer Science, California State University Long Beach, CA, USA.

Abstract

A set of mobile robots (represented as points) is distributed in the Cartesian plane. The collection contains an unknown subset of byzantine robots which are indistinguishable from the reliable ones. The reliable robots need to gather, i.e., arrive to a configuration in which at the same time, all of them occupy the same point on the plane. The robots are equipped with GPS devices and at the beginning of the gathering process they communicate the Cartesian coordinates of their respective positions to a central authority. On the basis of this information, and without the knowledge of which robots are faulty, the central authority designs a trajectory for every robot. The central authority aims to provide the trajectories which result in the shortest possible gathering time of the reliable robots. The efficiency of a gathering strategy is measured by its competitive ratio, i.e., the maximal ratio between the time required for gathering achieved by the given trajectories and the optimal time required for gathering in the offline case, i.e., when the faulty robots are known to the central authority in advance. The role of the byzantine robots, controlled by the adversary, is to act so that the gathering is delayed and the resulting competitive ratio is maximized.

The objective of our paper is to propose efficient algorithms when the central authority is aware of an upper bound on the number of byzantine robots. We give optimal algorithms for collections of robots known to contain at most one faulty robot. When the proportion of byzantine robots is known to be less than one half or one third, we provide algorithms with small constant competitive ratios. We also propose algorithms with bounded competitive ratio in the case where the proportion of faulty robots is arbitrary.
9.1 Introduction

9.1.1 The background

A collection of mobile robots need to meet at some point of the geometric environment. This task, known as gathering or rendezvous, has been extensively investigated in the past. The gathering may be necessary, e.g., to coordinate a future task or to exchange previously acquired information.

In most formerly studied cases, robots have limited knowledge about the environment and they do not know the positions of the other robots. In the present paper, the robots are distributed in the two-dimensional Cartesian plane. They are equipped with GPS devices and they can wirelessly communicate their positions to the central authority. The central authority then informs each individual robot of the trajectory it is to follow in order to meet. However, the team of reliable robots has been contaminated with “spies” - a subset of byzantine robots, indistinguishable from the original ones, controlled by an omnipotent adversary. The role of the faulty robots is simple – delay the gathering of the reliable ones for as long as possible. A byzantine robot may report a wrong position, fail to report any, or fail to follow its assigned route. As the central authority does not recognize which robots are byzantine, it sends the travel instructions to all of them.

Our goal is to design a strategy resulting in gathering of all reliable robots within the smallest possible time. We attempt to minimize the competitive ratio – the ratio of the time required to achieve gathering of the reliable robots, to the time required for such gathering to occur under the assumption that the reliable robots were known in advance.

9.1.2 The model and the problem

A collection $S$ of $n$ mobile robots (represented as points, and so $S$ is effectively a set of points) move at maximum unit speed within the two-dimensional plane. It is assumed that each robot in $S$ is equipped with a GPS device so it is aware of a pair of Cartesian coordinates representing its current location in the plane. We will assume that the robots are initially located in general position (no three points are collinear and no four points lie on a circle).

We consider the problem of gathering an unknown subset $N \subseteq S$ of robots. The robots of $N$ need to arrive at a same point on the plane eventually in order to complete some given task. We refer to this set $N$ of at least $n - F$ robots as the set of reliable robots and define $F = S \setminus N$ of $f \leq F$ robots as the set of byzantine robots. We call this problem of gathering all reliable robots from a collection containing at most $F$ byzantine robots the $\text{GATHER}(n, F)$ problem.

At the beginning, all robots in $S$ send a single message recording their starting positions to a central authority. In turn, this central authority computes a set of trajectories instructing
each robot how to time their respective movements in order to achieve gathering. At this point the robots follow the trajectories provided.

The movement continues until all reliable robots meet for the first time. We imagine a successful gathering as a meeting of robots possessing pieces of information allowing them to solve some puzzle. As long as all pieces are disassembled, the puzzle remains unsolved, and the identification of useful or invalid information is not possible.

The byzantine robots may report incorrect initial locations, which can potentially adversely affect the robots’ trajectories. Clearly, this results in byzantine robots not being able to follow the assigned trajectories. However, as long as all reliable robots complete their trajectories, the schedule must lead to their gathering.

The trajectories designed by the central authority are computed uniquely on the basis of the reported set of robot positions and possibly using the knowledge of the upper bound on the number of byzantine robots. Once the robots start their movements they will continue until all reliable robots have gathered. No adaptation to our algorithm is ever possible as no extra information may be obtained. We assume that the adversary knows in advance our algorithm and it will put the byzantine robots in the positions which result in the worst possible competitive ratio.

9.1.3 Our results

We provide algorithms with constant competitive ratio for all but a small bounded region in the space of possible \( n \) and \( F \) pairs. In doing so we demonstrate that having knowledge of the upper bound of the number of byzantine robots in the subset (represented by the parameter \( F \)) permits fine-tuning of the gathering algorithm, resulting in better competitive ratio.
ratios. In Section 2 we consider the gathering problem for collections involving only a single byzantine robot. After developing insight into the problem we give a gathering algorithm that is optimal for any number of robots, at most one of which is byzantine. For the boundary case of three robots, one of which is byzantine, we give a closed form expression for the competitive ratio. Section 3 presents two algorithms with small constant competitive ratio when the number of byzantine robots is bounded by a small fraction of $n$. Specifically, we give algorithms with competitive ratios of 2 and $1 + \sqrt{2}$ when $F < \lceil n/3 \rceil$ and $F < \lceil n/2 \rceil$ respectively.

Table 9.1: Summary of competitive ratio bounds for various algorithms.

<table>
<thead>
<tr>
<th>$F$</th>
<th>Upper-bound</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>optimal</td>
<td>Alg. 15</td>
</tr>
<tr>
<td>$\leq \lceil n/3 \rceil$</td>
<td>2</td>
<td>Alg. 16</td>
</tr>
<tr>
<td>$\leq \lceil n/2 \rceil$</td>
<td>$1 + \sqrt{2}$</td>
<td>Alg. 17</td>
</tr>
<tr>
<td>$&gt; [32\sqrt{2}] - 2$</td>
<td>$32\sqrt{2}$</td>
<td>Alg. 18</td>
</tr>
<tr>
<td>$\leq [32\sqrt{2}] - 2$</td>
<td>$F + 2$</td>
<td>Alg. 19</td>
</tr>
</tbody>
</table>

Finally, in Section 4, we give two gathering algorithms solving the problem for any $n$ and any $F$. The competitive ratio of one of these algorithms is constant, while the other is bounded by $F + 2$. We summarize the results of the paper in Table 9.1 and Figure 9.1.

9.1.4 Related work

The gathering problem was originally introduced in [157] as a version of pattern formation (see also [82]). In operations research, Alpern [7] considers the gathering of two robots,
referred to as the rendezvous problem (cf. [147]). Both problems are central in theoretical computer science. The rich related literature is due to the large variety of studied settings: deterministic and randomized, synchronous and asynchronous, for labeled and anonymous agents, in graphs and geometric environments, for same-speed or distinct maximal speed agents, etc. (cf. [8, 44, 49, 83, 102, 134, 149, 161]). More recently, efficient solutions were proposed for the plane [54] and for grids [52].

In many papers on gathering the agents are a priori assumed to have limited knowledge of the environment. Moreover, most papers supposed that an agent is not aware of the positions in the environment of other agents. In the deterministic settings, one of the central studied questions was feasibility of gathering or rendezvous, cf. [82, 83, 149], which most often led to some form of the symmetry breaking problem, see [134, 147]. Surprisingly, when agents were equipped with GPS devices, knowledge of the agent’s own position in the environment permitted executing very efficient rendezvous algorithms (see [48, 49]).

Fault tolerance in mobile agent algorithms has also been extensively studied in the past, but the failures were more often related to the static elements of the environment (network nodes or links), cf. [120, 140]. The faults of the mobile agents were studied for the problems of convergence [47], flocking [160], searching [70, 77] or patrolling [56]. Faults or imperfections concerning mobile agents performing gathering were investigated in [1, 46, 88, 125, 153]. Research in [46], [125] and [153] considered the gathering problem in the presence of inaccurate or faulty robot perception components. In [1] the initial positions of the collection is known to all robots, which operate in so called look-compute-move cycle. The feasibility of the problem, as a function of faulty robots, is investigated in [1] for crash and byzantine faults. In [88], the gathering problem is studied in an unknown graph environment and the feasibility question for byzantine faults in the strong and weak sense are investigated. The results of [88] depend on the knowledge of the upper bound on the size of the graph environment (or the absence of such knowledge).

In [42] the authors studied, similar to ours, the online rendezvous problem using GPS-equipped robots on a line, where some robots may turn out to be byzantine. However the robot movements along the line are much easier to analyze than the setting studied in the present paper. Indeed, in the case of a line, the robots move inside a corridor forcing robots to meet.

9.1.5 Notation

We will use $S$ to refer to a general collection of any robots (reliable and/or byzantine) and use $N \left( F \right)$ to represent a set of reliable (byzantine) robots only. We will represent the cardinality of a set $S$ as $|S|$ and will always use $n = |S|$, and $f = |F|$. We reserve the use of $F$ for the upper bound on the number of byzantine robots in $S$ (and, as such, it may be that $f \leq F$).

As we are dealing with robots in the plane we will use the term robot and point inter-
changeably. When it is required to refer to a particular robot/robots in a set we will use the capital letters \(A, B,\) and \(C\). We use the capital letter \(D\) to refer to meeting points of robots.

We let the distance between any two points \(A\) and \(B\) be \(|AB|\), and use \(\overrightarrow{AB}\) to represent the directed line segment joining \(A\) and \(B\). We will refer to the individual coordinates of a point using the subscripts \(x\) and \(y\), e.g., \(A = (A_x, A_y)\).

We define \(MC(S)\) as the minimum enclosing circle (MEC) of a set of points \(S\), and let \(\text{Sup}[S]\) be the supporting set of \(MC(S)\). It is a well known property that for any set of points in general position we have \(2 \leq |\text{Sup}[S]| \leq 3\)\(^{[41]}\). We further define the radius \(\text{Radius}[S]\) and center \(\text{Center}[S]\) of \(S\) to be the radius and center of the MEC of \(S\) respectively.

Finally, we let \(FVD(S)\) represent the furthest-point Voronoi diagram (FVD) of the point set \(S\), and, for a point \(A\) in \(S\), we let \(FVR(A)\) be the cell/region in \(FVD(S)\) belonging to the point \(A\). See \([27]\) for a description of the properties of the FVD.

### 9.2 One byzantine robot

In this section we develop optimal algorithms to solve the \(GATHER(n, 1)\) problem. However, for simplicity in presentation, we will assume for now that there is exactly one (instead of at most one) byzantine robot in \(S\). To make a distinction between this and our original problem we refer to this modified problem as the \(GATHER^*(n, 1)\) problem. At the end of the section we will consider what happens when there is at most 1 byzantine robot in \(S\).

We will need to consider subsets of \(S\) containing \(n-1\) robots and we therefore introduce some convenient notation. We let \(S_i \subset S, i \in [0, n-1]\) represent the \(n\) subsets of \(n-1\) robots that can be formed from \(S\) and we define an ordering for the \(S_i\) in such a way that \(\text{Radius}[S_i] \leq \text{Radius}[S_j] \forall j \geq i\). For the sake of brevity, we use \(r_S = \text{Radius}[S]\) and \(r_i = \text{Radius}[S_i]\) for the remainder of the section.

We start with the following (trivial) lemma concerning the optimal meeting time of any set of robots in the plane,

**Lemma 9.2.1.** The minimal time needed to gather a set \(S\) of robots is \(T^*(S) = r_S\).

An immediate consequence of the above lemma is the following optimal algorithm for gathering a group of \(n\) reliable robots.

**Algorithm 12 (Optimal \(GATHER(n, 0)\))**

**Initialize:**
1: \(D \leftarrow \text{Center}[S]\);

**Begin:**
2: All robots in \(S\) move at full speed towards \(D\);
3: The algorithm terminates when the last robot in \(S\) reaches \(D\);
To get an idea of how different the problem is when we consider the presence of even a single byzantine robot, let us run the above algorithm on the two inputs depicted in Figure 9.2.

Figure 9.2: Inputs for example analysis of competitive ratio. In both cases the robots $A$, $B$, and $C$ move directly towards the center of the minimum enclosing circle of $S = \{A, B, C\}$.

For a given input $S = \{A, B, C\}$ the adversary can choose at most one of the robots $A$, $B$, and $C$ to be byzantine. We assume that they will do so in such a way as to maximize the competitive ratio of our algorithm. Which robot would they choose?

In the case $a)$ the choice is not so obvious, and, indeed, the competitive ratios for all three possibilities are not very different. In the case $b)$, however, there is an obvious choice: the adversary would make $C$ byzantine since the robots $A$ and $B$ were initially very close but travelled far before meeting.

Therefore, Algorithm 12 is not optimal when at least one robot is byzantine. Observe that after $A$ and $B$ meet we can deduce that $C$ is byzantine. Similarly, if either $A$ or $B$, say $A$, does not reach the meeting point, we can deduce that $A$ is byzantine.

This exercise, although simple, highlights an important observation – the “closest” robots should meet first. It turns out that, when $F = 1$, we can formalize this statement\footnote{When $F > 1$ there are cases when this is not true.}.

**Lemma 9.2.2.** Consider an optimal algorithm $\overline{A}$ solving the GATHER∗($n$, 1) problem for the input $S$. Let $S_i$ be the first group of $n - 1$ robots to meet. Then $S_i = S_0$, i.e. $S_i$ is the group of $n - 1$ robots in $S$ with the smallest enclosing circle.

**Proof.** (Lemma 9.2.2) Assume we have an optimal algorithm $\overline{A}$ such that $S_i$ – the first group to gather using $\overline{A}$ – is not the group with the smallest enclosing circle, i.e. $S_i \neq S_0$. In this case an adversary chooses $\mathcal{N} = S_0$. Since the minimal time at which all robots can gather is $r_S$, the competitive ratio of $\overline{A}$ is $\geq r_S/r_0$.

Now apply Algorithm 12 to solve this GATHER∗($n$, 1) problem and observe that the competitive ratio of this algorithm is equal to $r_S/r_0$. This implies that $\overline{A}$ is, at best, as good
as Algorithm 12. However, it is not difficult to show that Algorithm 12 is not an optimal algorithm solving the $\text{GATHER}^\ast(n, 1)$ problem. Thus, we must conclude that $\overline{A}$ is not an optimal algorithm either – a contradiction.

So, we now know that we have to make the smallest group of $n - 1$ robots meet first. What choice does this leave the adversary? Well, naturally, they would choose the byzantine robots in such a way that the second-smallest group of $n - 1$ robots should have gathered. This observation leads us to the following:

**Theorem 9.2.3.** The competitive ratio of any algorithm solving the $\text{GATHER}^\ast(n, 1)$ problem with input $S$ is at least $r_S / r_1$.

**Proof.** (Theorem 9.2.3) Consider an algorithm $A$ solving the $\text{GATHER}^\ast(n, 1)$ problem with input $S$. Let $S_i$ be the first group of $n - 1$ robots to meet using $A$, and let $S_j = N$ be the group of $n - 1$ reliable robots. Observe that an adversary can always choose to make $S_i \neq S_j$ such that, effectively, all $n$ robots must meet before $A$ terminates. Let the time at which this happens be $T$. Lemma 9.2.1 tells us that $r_S$ is the minimum time necessary to gather all $n$ robots and we thus have $T \geq r_S$. The competitive ratio of $A$ is therefore at least $\hat{\text{CR}} \geq r_S / r_j$, $j \neq i$. There are two cases to consider: $i = 0$ and the adversary chooses $j = 1$ such that $\hat{\text{CR}} \geq r_0 / r_1$, or, $i \neq 0$ and the adversary chooses $j = 0$ such that $\hat{\text{CR}} \geq r_i / r_0 \geq r_i / r_1$.

At this point we can make a useful observation: an optimal gathering algorithm ends either at the moment the first group of robots meet or the moment all robots meet. Furthermore, at the moment of the first meeting, all robots are located at either one of only two positions. Thus, in an optimal algorithm, we must send these remaining two groups of robots directly towards each other. We can claim the following:

**Lemma 9.2.4.** An optimal algorithm $\overline{A}$ solving the $\text{GATHER}^\ast(n, 1)$ problem can be completely described by the single point $D$ at which the first $n - 1$ robots gather.

**Corollary 9.2.5.** (Lemma 9.2.4) There is an optimal algorithm solving the $\text{GATHER}^\ast(n, 1)$ problem following the strategy given in Algorithm 13.
Algorithm 13 (General GATHER\(^*\)(n, 1))

**Input:** Set of initial positions of robots, \(S\). Gathering point, \(D\).

**Initialize:** (Compute the midpoint \(D'\) between \(D\) and the position of the single robot that will not have arrived at \(D\) at the time the first group of robots gather at \(D\)).

1: \(A \leftarrow \) position of (one of) the robot(s) furthest from \(D\);
2: \(B \leftarrow \) position of (one of) the robot(s) second furthest from \(D\);
3: \(D' \leftarrow D + \frac{1}{2} \left(1 - \frac{|BD|}{|AD|}\right) (A - D)\);

**Begin:**

4: for each robot \(R \in S\) do
5: \(d \leftarrow \min\{|RD|, |BD|\}\);
6: move at full speed distance \(d\) towards \(D\); wait time \(|BD| - d\);
7: if all reliable robots have gathered then the algorithm terminates;
8: else all robots move at full speed towards \(D'\). The algorithm terminates once they meet;

Corollary 9.2.5 reduces the task of searching for an optimal algorithm to the conceptually simpler task of searching for some optimal meeting point \(D\). The following lemma tells us how to find this point:

**Lemma 9.2.6.** Consider an optimal algorithm \(\overline{A}\) solving the GATHER\(^*\)(n, 1) problem for the input \(S\) parameterized by the point \(D\). Let the group \(S_i\) represent the first group of \(n - 1\) robots to gather at the point \(D\). Then the point \(D\) lies on the perpendicular bisector of the two robots in \(S_i\) furthest from \(D\).

**Proof.** (Lemma 9.2.6) Let \(A\) and \(B\) be the last two robots in \(S_i\) that reach \(D\) and let \(C\) be the single robot in \(S\) that is not contained in \(S_i\). We argue by contradiction and assume that the point \(D\) does not lie on the perpendicular bisector \(AB_\perp\) of \(A\) and \(B\). Without loss of generality assume that \(A\) reaches \(D\) before \(B\). This situation is depicted in Figure 9.3.

If we let the group of \(n - 1\) reliable robots be \(S_j = N\) then the competitive ratio of \(\overline{A}\) is

\[
\overline{\text{CR}}(\overline{A}, S) = \max_{S_j} \begin{cases} \frac{1}{r_i} |BD|, & S_i = S_j \\ \frac{1}{2r_j} (|BD| + |CD|), & \text{otherwise} \end{cases}
\]

Now consider the algorithm \(A'\) which replaces the point \(D\) in \(\overline{A}\) with the point \(D'\) on the segment \(BD\) located some small distance \(\epsilon\) in the direction of \(AB_\perp\) such that \(A\) and \(B\) are still the last two robots to arrive at \(D'\) (and \(A\) arrives before \(B\). The competitive ratio of this new algorithm is,

\[
\overline{\text{CR}}(A', S) = \max_{S_j} \begin{cases} \frac{1}{r_i} |BD'|, & S_i = S_j \\ \frac{1}{2r_j} (|BD'| + |CD'|), & \text{otherwise} \end{cases}
\]

249
We claim that \( \hat{\text{CR}}(A', S) < \hat{\text{CR}}(\overline{A}, S) \). It is obvious that \( |BD'| < |BD| \). Thus we need to show that \( |BD'| + |CD'| < |BD| + |CD| \). Indeed, observe that
\[
|BD| + |CD| = |BD'| + |DD'| + |CD| > |BD'| + |CD'| \quad \text{(triangle inequality)}.
\]

We can thus conclude that \( \hat{\text{CR}}(A', S) < \hat{\text{CR}}(\overline{A}, S) \) which is in contradiction to our assumption that \( \overline{A} \) is an optimal algorithm.

As a last step we derive an expression for the competitive ratio of an optimal \( \text{GATHER}^*(n, 1) \) algorithm.

**Lemma 9.2.7.** An optimal algorithm following the strategy in Algorithm 13 solves the \( \text{GATHER}^*(n, 1) \) problem for the input \( S \) with competitive ratio
\[
\hat{\text{CR}} = \max \left\{ \frac{|AD|}{r_0}, \frac{|AD| + |CD|}{2r_1} \right\} \quad (9.2)
\]
where \( A \) is one of the two points in \( S_0 \) furthest from \( D \) and \( C \) is the point in \( S \) that is not in \( S_0 \).

**Proof.** (Lemma 9.2.7) Lemma 9.2.2 tells us that the first group of \( n - 1 \) robots to gather is the group \( S_0 \). Thus, if \( A \) is the point in \( S_0 \) that is furthest from \( D \), then, in the case that \( S_0 = N \), the competitive ratio of the algorithm is \( |AD|/r_0 \).

If \( S_0 \neq N \), then, if \( C \) is the single point in \( S \) that is not in \( S_0 \), the algorithm terminates after time \( (|AD| + |CD|)/2 \). Thus, the overall competitive ratio of the algorithm is \( \hat{\text{CR}}(D) = \max \left\{ \frac{|AD|}{r_0}, \frac{|AD| + |CD|}{2r_1} \right\} \) as required.
We are now ready to present our main result:

**Algorithm 14 (Optimal GATHER∗(n, 1) point)**

**Input:** Set of initial positions of robots, \(S\).

**Begin:**
1. \(C \leftarrow \text{position of the single robot in } S \text{ that is not in } S_0\);
2. Compute \(FVD(S_0)\) of the point set \(S_0\);
3. \(CR_{min} \leftarrow \infty; D_{min} \leftarrow NULL\);
4. for each edge \(E\) in \(FVD[S_0]\) do
5. \((A, B) \leftarrow \text{two points such that } E \text{ separates } FVR(A) \text{ and } FVR(B)\);
6. \(D' \leftarrow \text{point on } E \text{ minimizing } CR(D') = \max \left\{ \frac{|AD'|}{r_0}, \frac{|AD'|+|CD'|}{2r_1} \right\}\).
7. if \(CR(D') < CR_{min}\) then
8. \(D_{min} \leftarrow D'; CR_{min} \leftarrow CR(D')\)
9. return \(D_{min}\).

**Algorithm 15 (Optimal GATHER∗(n, 1))**

**Input:** Set of initial positions of robots, \(S\)

**Initialize:** \(D \leftarrow \text{output of Algorithm } 14\)

**Begin:** The robots perform Algorithm 13 with input \(D\);

**Theorem 9.2.8.** Algorithm 15 is an optimal algorithm solving the GATHER∗(n, 1) problem with input \(S\). The computational complexity of the algorithm is \(O(n \log n)\).

**Proof.** (Theorem 9.2.8) By Lemmas 9.2.2 and 9.2.6 we know that the optimal point \(D\) must lie on the perpendicular bisector of two points in \(S_0\) that are furthest from \(D\). In Algorithm 14 we are choosing \(D\) to be on one of the edges of the FVD of \(S_0\). Thus, by construction, \(D\) does in fact lie on the perpendicular bisector of two points in \(S_0\) that are furthest from \(D\).

The fact that the algorithm is optimal follows from Lemma 9.2.7 and the definition of an optimal algorithm given in Eq. (9.1).

The complexity of the algorithm is \(O(n \log n)\) since we need to determine the FVD of \(S_0\) which can be found optimally in \(O(n \log n)\) time [27]. Minimizing the quadratic equation given in Lemma 9.2.7 on a line segment can be done in constant time, and this needs to be done only once for each of the \(O(n)\) edges of the FVD.

It does not seem likely that a closed form expression can be derived for the competitive ratio of Algorithm 15 for arbitrary \(n\). However, in the boundary case that \(n = 3\) and \(F = 1\) this is possible.
9.2.1 Gathering three robots

Observe that we can describe an instance of the GATHER∗(3, 1) problem by the triangle \( \triangle ABC \) whose vertices specify the initial positions of the three robots for which gathering should occur. Thus, throughout this section, we let \( a = |BC|, b = |AC|, \) and \( c = |AB| \) be the side lengths of \( \triangle ABC \), and set the angles \( \alpha = \angle CAB, \beta = \angle ABC, \) and \( \gamma = \angle BCA \). Without loss of generality we assume that \( a \leq b \leq c \).

With this description of the problem, Lemmas 9.2.2, 9.2.6, 9.2.7, and Theorem 9.2.3 take the following simple forms:

**Corollary 9.2.9.** *(Lemma 9.2.2 and Lemma 9.2.6)* An optimal algorithm solving the GATHER∗(3, 1) problem with input \( \triangle ABC \) has the robots \( B \) and \( C \) meet first at some point on their perpendicular bisector.

**Corollary 9.2.10.** *(Theorem 9.2.3 and Lemma 9.2.7)* An optimal algorithm solving the GATHER∗(3, 1) problem with input \( \triangle ABC \) has competitive ratio

\[
\hat{\text{CR}} = \max \left\{ \frac{2|BD|}{a}, \frac{|AD| + |BD|}{b} \right\}
\]

(9.3)

and this is at least \( c/b \).

At this point we could simply apply Algorithm 15 to determine the optimal point for this problem, however, in order to derive a closed form expression for the point \( D \) we take a different approach. We claim the following:

**Theorem 9.2.11.** Algorithm 13 optimally solves the GATHER∗(3, 1) problem with input \( \triangle ABC \) of side lengths \( a \leq b \leq c \) and respective angles \( \alpha \leq \beta \leq \gamma \) if the point \( D \) is chosen such that

\[
D_x = \frac{1}{2} \left[ (B_x + C_x) + a(B_y - C_y) \tan \phi \right],
\]

\[
D_y = \frac{1}{2} \left[ (B_y + C_y) + a(C_x - B_x) \tan \phi \right],
\]

where \( \phi = \angle DBC \) and \( \tan \phi = \tan \beta \) if \( \tan \beta \leq \sin \gamma \), otherwise

\[
\tan \phi = \frac{2 \sqrt{c^2 - (a-b)^2}}{\sqrt{(a-3b)^2 - c^2} + \sqrt{(a+b)^2 - c^2}}.
\]

The competitive ratio of the algorithm equals \( c/b \) if \( \tan \beta \leq \sin \gamma \), otherwise it is \( 1/\cos \phi \).

**Proof.** *(Theorem 9.2.11)* First let us define \( \text{CR}_A \) and \( \text{CR}_B \) as follows

\[
\text{CR}_A = \frac{2|BD|}{a}, \quad \text{CR}_B = \frac{|BD| + |AD|}{b}.
\]
Since $D$ must lie on the perpendicular bisector of $B$ and $C$, we can see that, as $\phi$ ranges from 0 to $\beta$, $CR_B$ is a monotone continuous decreasing function of $\phi$ and $CR_A$ is a monotone continuous increasing function of $\phi$. On the other hand, as $\phi$ increases past $\beta$ both $CR_A$ and $CR_B$ monotonically increase with $\phi$. Thus, in order to minimize the competitive ratio we must choose $\phi \in [0, \beta]$.

We claim that, when $\tan \beta \leq \sin \gamma$, we will take $\phi = \beta$. Indeed, in this case we have $2 |BD|/a = 1/\cos \beta$ and $|BD| + |AD| = |AB| = c$. Thus, $CR_A = \frac{1}{\cos \beta}$ and $CR_B = \frac{c}{b}$. We observe that $\tan \beta \leq \sin \gamma$ implies

$$\tan \beta \leq \sin \gamma \rightarrow \frac{1}{\cos \beta} \leq \frac{\sin \gamma}{\sin \beta} = \frac{c}{b}$$

where we have invoked the sine law in the last step. Since $\hat{CR} = \max \{CR_A, CR_B\}$ and since $CR_B$ increases as $\phi$ moves away from $\beta$, we can conclude that the optimal value of $\phi$ is $\phi = \beta$ when $\tan \beta \leq \sin \gamma$. This case is depicted on the top of Figure 9.4.

Figure 9.4: Setup for the proof of Theorem 9.2.11. The shaded gray region indicates those positions of $A$ such that $a \leq b \leq c$. Top: $\tan \beta \leq \sin \gamma$. Bottom: $\tan \beta \geq \sin \gamma$

Now consider the case that $\tan \beta > \sin \gamma$. If $\phi = \beta$ then

$$\tan \beta > \sin \gamma \rightarrow \frac{1}{\cos \beta} > \frac{\sin \gamma}{\sin \beta} = \frac{c}{b}$$
and we can therefore improve the competitive ratio of the algorithm by taking \( \phi \in [0, \beta) \). Since \( CR_A \) decreases and \( CR_B \) increases as we decrease \( \phi \) from \( \beta \) we can conclude that the optimal competitive ratio is given when \( CR_A = CR_B \). The position of the optimal point \( D \) is then determined from the equation \( \frac{2|BD|}{a} = \frac{|BD| + |AD|}{b} \). We will write this equation in the following way:

\[
(2b - a)|BD| = a|AD|.
\]

(9.4)

Now, let us choose a coordinate system such that the midpoint of \( B \) and \( C \) is at the origin, and the positive \( y \)-axis lies along the perpendicular bisector of \( B \) and \( C \). In this coordinate system \( B = (-a/2, 0) \), \( C = (a/2, 0) \), \( A = (A_x, A_y) \), and \( D = (0, d) \) where we have introduced the parameter \( d \). Thus, we can write

\[
|BD| = \sqrt{\frac{1}{4}a^2 + d^2}, \quad |AD| = \sqrt{A_x^2 + (A_y - d)^2}.
\]

Plugging these into Eq. (9.4) and squaring both sides of the equation yields

\[
(2b - a)^2 \left[\frac{1}{4}a^2 + d^2\right] = a^2 \left[A_x^2 + (A_y - d)^2\right].
\]

After some basic manipulation we can write this as a quadratic equation for \( d \)

\[
[(2b - a)^2 - a^2]d^2 + 2a^2A_yd + \frac{a^2(2b - a)^2}{4} - a^2(A_x^2 + A_y^2) = 0.
\]

We can simplify this further to get

\[
4b(b - 1)d^2 + 2a^2A_yd + a^2 \left[b^2 - ab + \frac{a^2}{4} - (A_x^2 + A_y^2)\right] = 0.
\]

We now set \( \Delta = \frac{1}{2}aA_y \) as the area of \( \triangle ABC \). We also observe that \( \sqrt{A_x^2 + A_y^2} \) is the median of triangle \( \triangle ABC \) from the vertex \( A \) and we can write this median in terms of the side lengths \( a, b, c \) as

\[
\sqrt{A_x^2 + A_y^2} = \sqrt{\frac{b^2 + c^2}{2} - \frac{a^2}{4}}.
\]

We can therefore write the quadratic equation for \( d \) as

\[
4b(b - a)d^2 + 4a\Delta d + a^2 \left[b^2 - ab + \frac{a^2}{4} - \frac{b^2 + c^2}{2} + \frac{a^2}{4}\right] = 0
\]

which simplifies to

\[
4b(b - a)d^2 + 4a\Delta d - \frac{a^2}{2} \left[c^2 - (b - a)^2\right] = 0.
\]
Solving for \( d \) we get

\[
d = \frac{-4a\Delta + \sqrt{16a^2\Delta^2 + 8a^2b(b-a)[c^2 - (b-a)^2]}}{8b(b-a)}
\]

\[
= \frac{-4a\Delta + 4a\sqrt{\Delta^2 + \frac{b(b-a)[c^2-(b-a)^2]}{2}}}{8b(b-a)}
\]

\[
= \frac{a}{2b(b-a)} \left[ \sqrt{\Delta^2 + \frac{b(b-a)[c^2-(b-a)^2]}{2}} - \Delta \right].
\]

Multiplying and dividing the right hand side by \( \sqrt{\Delta^2 + \frac{b(b-a)[c^2-(b-a)^2]}{2}} + \Delta \) yields

\[
d = \frac{a \left[ \Delta^2 + \frac{b(b-a)[c^2-(b-a)^2]}{2} - \Delta^2 \right]}{2b(b-a) \left( \sqrt{\Delta^2 + \frac{b(b-a)[c^2-(b-a)^2]}{2}} + \Delta \right)}
\]

\[
= \frac{a[c^2-(b-a)^2]}{\sqrt{16\Delta^2 + 8b(b-a)[c^2-(b-a)^2] + 4\Delta}}.
\]

Finally, since \( c^2 - (b-a)^2 = (-a+b+c)(a-b+c) \) we have

\[
d = \frac{a\sqrt{(-a+b+c)(a-b+c)}}{\sqrt{\frac{16\Delta^2}{(-a+b+c)(a-b+c)}} + 8b(b-a) + \frac{4\Delta}{\sqrt{(-a+b+c)(a-b+c)}}}.
\]

We will now make use of Heron’s formula for the area of a triangle which states that

\[
\Delta = \frac{1}{4} \sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}.
\]

The expression for \( d \) now becomes

\[
d = \frac{a\sqrt{(-a+b+c)(a-b+c)}}{\sqrt{(a+b+c)(a+b-c) + 8b(b-a) + \sqrt{(a+b+c)(a+b-c)}}}
\]

\[
= \frac{a\sqrt{c^2-(a-b)^2}}{\sqrt{(a-3b)^2-c^2} + \sqrt{(a+b)^2-c^2}}
\]

Finally, by noting that \( d = \frac{a}{2} \tan \phi \), we get our final result:

\[
\tan \phi = \frac{2\sqrt{c^2-(a-b)^2}}{\sqrt{(a-3b)^2-c^2} + \sqrt{(a+b)^2-c^2}}.
\]
It is interesting to note that in some cases ($\tan \beta \leq \sin \gamma$) it was possible to achieve the lower-bound given in Theorem 9.2.3. This turns out to be true for the general case as well, and, in fact, it is possible to specify under which conditions this occurs:

**Lemma 9.2.12.** Consider an input $S$ such that $|\text{Sup}(S)| = 2$. Let $L$ be the line segment defined by the two points in $\text{Sup}(S)$, and let $P$ be the convex region defined by the intersection of all circles with centers given by the points in $S_0$ and radii given by $r_0 \cdot r_S/r_1$. Then, if $L$ intersects with $P$, the competitive ratio of Algorithm 15 is $\hat{CR} = r_S/r_1$.

**Proof.** (Lemma 9.2.12) Define the convex region $P$ as above and observe that $P \neq \emptyset$ since, at minimum, it contains the center of $MC(S_0)$. Also set $q = r_0r_S/r_1$.

Now, we let $A \in S_0$ and $C \in S_1$ be the two points that are in $\text{Sup}(S)$. Observe that for any point $D$ on the segment $L = AC$ we have $|AD| + |CD| = |AC| = 2r_S$. Thus, if $A$ is (one of) the furthest point(s) from $D$, the algorithm terminates in at most $r_S$ time.

Now observe that for any point $D \in P$ the distance between $D$ and any point in $S_0$ is at most $q$. In particular, for the point $A$, we have $|AD| \leq q$.

Thus, if it happens that: a) $P$ and $L$ intersect, and, b) we can choose $D$ in $P \cap L$ in such a way that $A$ is (one of) the furthest point(s) from $D$, then, the competitive ratio of Algorithm 15 is

$$\hat{CR} = \max \left\{ \frac{|AD|}{r_0}, \frac{|AD| + |CD|}{2r_1} \right\} \leq \max \left\{ \frac{q}{r_0}, \frac{2r_S}{2r_1} \right\} \leq \max \left\{ \frac{r_S}{r_1}, \frac{r_S}{r_1} \right\} = \frac{r_S}{r_1}.$$

Now, observe that there must be a point in $P \cap L$ such that $A$ is furthest from it. One such point, for example, is the point of intersection of the segment $L$ and the circle of radius $q$ centered on $A$. In order to get the optimal point we simply need to choose the $D$ that minimizes Eq. (9.2) on the portion of $L$ contained in the region of $\mathcal{FVD}(S_0)$ belonging to the point $A$. \qed

### 9.2.2 At most one faulty robot

Until now we have assumed that the robots know that there is exactly one faulty robot amongst them. However, our original formulation of the problem assumes that the number of faulty robots is only bounded and it is entirely possible that all $n$ of the robots are in fact reliable. Fortunately, this will not be a problem. Algorithm 13 is stated in a way that, even if the first $n-1$ robots that gather are reliable they will still need to meet the last remaining robot in order to terminate (they will know that not all reliable robots have gathered after the first meeting since they will not have gathered the information necessary to terminate). The main difference is that the competitive ratio will change since the optimal completion time
is now $r_S$ (instead of $r_0$ or $r_1$). However, since $r_S \geq r_1 \geq r_0$, the competitive ratio of the algorithm will actually be smaller if all robots are reliable. Thus, an adversary can achieve a worse case by choosing exactly one of the robots to be faulty and we can conclude that Algorithm 15 optimally solves the \textsc{Gather}($n, 1$) problem as well.

9.3 Bounded number of byzantine robots

We now consider instances of the \textsc{Gather}($n, F$) problem when the value of $F$ is a small constant fraction of $n$. We give two algorithms corresponding to the cases that $F < \lceil \frac{n}{3} \rceil$, and $F < \lfloor \frac{n}{2} \rfloor$. In both cases we show that a small constant competitive ratio is attainable. We start with the case that $F < \lceil \frac{n}{3} \rceil$.

**Theorem 9.3.1.** Consider the \textsc{Gather}($n, F$) problem with input $S$ and for any $F < \lceil \frac{n}{3} \rceil$. Then, there is a gathering algorithm solving this problem with competitive ratio at most 2. The computational complexity of the algorithm is $O(n)$.

**Proof.** (Theorem 9.3.1) We will make use of the centerpoint theorem (see [92] [Theorem 4.3]) which states that any finite set $S$ of $n$ points in $\mathbb{R}^d$ admits a point $K$ (a centerpoint) such that any open half-space avoiding $K$ contains at most $\lfloor \frac{dn}{d+1} \rfloor$ points of $S$. In particular, for $d = 2$, this implies that we can always determine a $K$ such that any line $L$ through $K$ partitions $S$ into two sets each with at least $\lceil \frac{n}{3} \rceil$ robots. This result inspires the following algorithm,

**Algorithm 16 (Move to centerpoint)**

**Input:** Set of initial positions of robots, $S$

**Initialize:** Compute centerpoint $K$ of $S$;

**Begin:**

1: All robots move at full speed towards $K$;
2: The algorithm terminates once the final reliable robot reaches $K$;

Consider the reliable robot $A$ that is initially furthest away from the point $K$ determined in Algorithm 16. Draw a line $L$ through $K$ perpendicular to the line segment $AK$ (as done in Figure 9.5). Observe that, since $K$ is a centerpoint, there are at least $\lceil \frac{n}{3} \rceil$ robots on either side of $L$. Furthermore, by assumption, $F$ is strictly less than $\lceil \frac{n}{3} \rceil$ and we are thus guaranteed to have a reliable robot on either side of $L$. Consider any reliable robot $B$ on the opposite side of $L$ as $A$ and note that the robot $B$ is a distance $|AB| \geq |AK|$ away from the robot $A$. The competitive ratio of Algorithm 16 is therefore at most $CR \leq |AK|/(\frac{1}{2} |AB|) \leq 2$.

The complexity bound follows from the need to determine the centerpoint of the collection. The centerpoint of a set of $n$ points can be determined in $O(n)$ time using an algorithm by Jadhav [126].

257
The centerpoint theorem applies generally to any $d$-dimensional space and we thus have the following corollary,

**Corollary 9.3.2.** (Theorem 9.3.1) Consider the $\text{GATHER}(n, F)$ problem in $\mathbb{R}^d$ for any $F < \lceil \frac{n}{d+1} \rceil$. Then, there exists a gathering algorithm with competitive ratio at most 2.

Now consider the case that $F < \lceil \frac{n}{2} \rceil$. We claim the following:

**Theorem 9.3.3.** Consider the $\text{GATHER}(n, F)$ problem with input $\mathcal{S}$ and for any $F < \lceil \frac{n}{2} \rceil$. Then, there is a gathering algorithm solving this problem with competitive ratio at most $1 + \sqrt{2}$. The computational complexity of the algorithm is $O(n)$.

**Proof.** (Theorem 9.3.3) The proof is based on the following algorithm,

**Algorithm 17 (Move to intersection)**

**Input:** Set of initial positions of robots, $\mathcal{S}$

**Initialize:**
1: Compute horizontal line $L_H$ (resp. vertical line $L_V$) such that at most $\lceil \frac{n}{2} \rceil$ robots lie on each open half-plane bounded by $L_H$ (resp. $L_V$);
2: $K \leftarrow$ intersection point of $L_H$ and $L_V$;

**Begin:**
3: The robots move at full speed towards the point $K$;
4: The algorithm terminates once the final reliable robot reaches $K$;

Consider the four closed regions (quadrants) $R_1$, $R_2$, $R_3$, and $R_4$ created by the intersection of $L_H$ and $L_V$ (as depicted in Figure 9.5). Note that, by assumption, we have $F < \lceil \frac{n}{2} \rceil$.
and we are therefore guaranteed to have at least one reliable robot in each of the regions $R_1$ and $R_3$, or in each of the regions $R_2$ and $R_4$ (note that we assume that a robot lying on the line $L_H$ or $L_V$ belongs to both quadrants separated by the line and a robot at point $K$ belongs to all four quadrants).

Consider the reliable robot $A$ that is furthest from $K$ and assume without loss of generality that $A$ is located in the region $R_1$. If there is a reliable robot $B$ in $R_3$ then we have $|AB| \geq |AK|$ which implies that $\hat{C}R \leq |AK|/(\frac{1}{2}|AB|) \leq 2$. If there is not a reliable robot in $R_3$ then there must be reliable robots $B$ and $C$ in $R_2$ and $R_4$ respectively, neither of them being at point $K$. The configuration of the three robots $A$, $B$ and $C$ which maximizes our competitive ratio is presented in Figure 9.6. Namely, the robot $A$ will lie somewhere in $R_1$, robot $B$ will lie on the boundary line between $R_1$ and $R_4$ (not at the point $K$), and robot $C$ will lie on the boundary line between $R_1$ and $R_2$ (not at the point $K$).

Observe that the competitive ratio of our algorithm is equal to $|AK|/R$, where $R$ is the radius of the smallest disk containing the reliable robots. It is easy to see that the competitive ratio for the configuration from Figure 9.6 is $\sqrt{2} + 1$. We show below that the ratio of any configuration of points $A$, $B$ and $C$, that is different from the one from Figure 9.6 can be improved by moving some of these points to a new position, so that after this move either $|AK|$ is augmented, or the value of $R$ is reduced.

Consider the configuration $\mathcal{C}$ of points $A$, $B$ and $C$ maximizing the competitive ratio of our algorithm. Let $\mathcal{D}$ denote the smallest disk containing points $A$, $B$ and $C$. Observe first,
that segment $AK$ must contain the center of $D$. Indeed, otherwise point $A$ may be moved within disk $D$ to a position further from $K$ increasing the ratio $|AK|/R$.

![Figure 9.7: Setup for the proof of Theorem 9.3.3.](image)

Suppose that $C$ is such that the disk $D$ is tangent to neither $L_H$ nor $L_V$ (see Figure 9.7 (a)). Then, $D$ may be reduced slightly to the disk $D'$ such that $D'$: still contains point $A$, is tangent to one of the lines $L_H$ and $L_V$, and still intersects quadrants $R_2$ and $R_4$. Hence there exist points $B'$ and $C'$ from quadrants $R_2$ and $R_4$, respectively, so that the disk $D'$ containing $A$, $B'$ and $C'$ has a radius smaller than the radius of $D$. This contradicts with the fact that $C$ maximized the competitive ratio.

Suppose now that $C$ is such that the disk $D$ is tangent to one of the lines $L_H$ and $L_V$, but not to both of them (see Figure 9.7 (b)). Suppose by symmetry that $D$ is tangent to $L_V$. This point of tangency must coincide with point $B$. As $B \neq K$, the segment $BK$ is shorter than the diameter of $D$ and we can create disk $D'$, slightly smaller than $D$, that still contains points $A$, $B$ and $C'$, where $C' \in R_4$ assuming a larger competitive ratio than the one for the configuration $C$. Consequently, the worst competitive ratio is attained when the three robots $A$, $B$ and $C$ are in the configuration from Figure 9.6. This concludes the proof.

The two lines $L_H$ and $L_V$ may be found in linear time by first choosing some line $L'$ onto which we project the points in $S$. We then set $L_H$ as the line perpendicular to $L'$ dividing the points on $L'$ in half (i.e. we need to find the median in $O(n)$ time [28]). To find $L_V$ we repeat with $L'$ replaced with $L_H$. 

\[\square\]
9.4 Arbitrary number of byzantine robots

In this section we consider algorithms that solve the $GATHER(n, F)$ for any $n$ and any $F$. We give two algorithms: the first, grid-rendezvous, is adapted from [48] and gives a constant competitive ratio independent of $F$. The second, shrinking-the-shortest-interval (SSI), gives a competitive ratio dependent on $F$.

9.4.1 Grid rendezvous

We start with the grid-rendezvous algorithm which is a direct application of Algorithm 3 in [48]. The algorithm was originally designed to solve the rendezvous problem of two robots unaware of the other’s position (but sharing a common coordinate system).

The idea of the algorithm is to calculate a hierarchy of grids $\Pi = \{\pi_0, \pi_1, \ldots\}$ which partition the plane into non-overlapping cells. The robots then travel through a series of potential meeting points located at the centers of ever larger cells from successive grids in $\Pi$.

In detail, each $\pi_i$ exactly partitions the plane into square cells of side length $2^i$ such that one of the cells in $\pi_i$, the central cell, has its center at the origin. In order for the partition to be exact each cell is defined to include its top and right edges, as well as its top-right vertex (in addition to its interior). The algorithm then proceeds in stages. In stage $i$ each robot will determine the grid cell of $\pi_i$ in which it is contained and then move towards the center of that cell. As the robots will potentially take different amounts of time to reach the center of their respective cells, each stage of the algorithm lasts a fixed amount of time. In this way, robots arriving early to their target locations will simply wait at that location until the next stage begins. The length of each stage is chosen such that all robots have sufficient time to reach the center of their cells.

Whenever two or more robots arrive to the same location they will remain together for the duration of the algorithm (byzantine robots excluded, however these robots are not required to gather and this will not pose any problems). Since the size of the cells increase in each stage, there will eventually be a stage during which all reliable robots are located in the same grid cell and at the end of this stage the algorithm will terminate.

We can nearly apply Algorithm 3 as given in [48]. We only need to specify the finest grid division that will be used by the robots. Let $d_e$ be the size of this finest grid cell. We present (the slightly modified) Algorithm 3 from [48] below. We note that, although this algorithm is presented in a way that the robots are actively computing new trajectories throughout the algorithm, we will implicitly assume that the central authority first simulates the algorithm and then broadcasts to each robot its corresponding trajectory.
Algorithm 18 (Grid-rendezvous [48])

1: The robots choose a $d_\epsilon$ much smaller than the closest pair of robots in the set;
2: The robots compute the hierarchy of grids $\Pi$;
3: repeat for $i = 1, 2, 3...$ and for each robot in $S$
4: Set $H$ equal to the cell of $\pi_i$ containing your initial position $p$;
5: Move to the center of $H$;
6: Wait until $\sqrt{2} \cdot 2^{i-1}$ time has passed since the start of the current iteration;
7: until Gathering completed

The rendezvous time of the above algorithm is given by Corollary 9 in [48]. Using this time-bound we can state the following:

**Theorem 9.4.1.** Consider the $\text{GATHER}(n, F)$ problem for the input $S$. Assume that the robots $A$ and $B$ are the closest pair of robots in $S$. Then the competitive ratio of Algorithm 18 is $\hat{CR} \leq 2\sqrt{2} \left( 16 + \frac{d_\epsilon}{|AB|} \right)$ where $d_\epsilon$ can be made as small as one chooses. The computational complexity of this algorithm is $O(n \log n)$.

**Proof.** (Theorem 9.4.1) Choose any two robots $I$ and $J$ in $S$ and assume that all distances are scaled such that $d_\epsilon = 1$. Then Corollary 9 from [48] tells us that the robots $I$ and $J$ gather in time $T \leq 16\sqrt{2} \cdot |IJ| + \sqrt{2}$.

Let $\mathcal{N}$ be the subset of reliable robots and consider the two robots $A$ and $B$ which are the most distant in $\mathcal{N}$. Then the minimal time necessary to gather the robots in $\mathcal{N}$ is at least $T_* \geq |AB|/2$. The competitive ratio of Algorithm 18 is therefore

$$\hat{CR} = \frac{T}{T_*} \leq \frac{16\sqrt{2}|AB| + \sqrt{2}}{\frac{1}{2}|AB|} \leq 2\sqrt{2} \left( 16 + \frac{1}{|AB|} \right).$$

In the worst case, $|\mathcal{N}| = 2$ and the robots $A$ and $B$ were the closest pair in $S$.

The computational complexity of the algorithm is $O(n \log n)$ as one needs to find the closest pair of points in $S$ in order to determine a satisfactory grid division $d_\epsilon$. The closest pair in a set of $n$ points can be found optimally in $O(n \log n)$ time [26].

\[ \square \]

9.4.2 Shrink-shortest-interval

Consider the following algorithm, generalized from Algorithm 3 in [42]:
Algorithm 19 (Shrink-shortest-interval)

Input: Set of initial positions of robots, \( S \).

Initialize: (Compute list of meeting points and travel distances of robots)
1: Initialize list \( D \) of size \( n - 1 \);
2: Initialize list \( d \) of size \( n - 1 \);
3: \( S' \leftarrow S \);
4: for \( i = 1 \) to \( n - 1 \) do
   5: \((A, B) \leftarrow \text{closest points in } S' \text{ that are not at the same position};
   6: \( D \leftarrow \text{midpoint of } A \text{ and } B;\)
   7: \( D(i) \leftarrow \text{midpoint of } A \text{ and } B;\)
   8: \( d(i) \leftarrow |AB|/2;\)
   9: Move all points in \( S' \) distance \( d(i) \) towards \( D(i);\)
10: Update \( S' \) with the new locations of the points;

Begin:
11: for \( i = 1 \) to \( n - 1 \) do
   12: All robots move at full speed a distance \( d(i) \) towards the point \( D(i);\)
   13: if All reliable robots have gathered then terminate

Theorem 9.4.2. Algorithm 19 solves the \( \text{GATHER}(n, F) \) problem for the input \( S \) with competitive ratio at most \( F + 2 \). The computational complexity of the algorithm is \( O(n^2 \log n) \).

To prove this we will need the following lemma:

Lemma 9.4.3. Consider any point \( D \) and set of points \( S \) such that \( A \in S \) is the closest point to \( D \), and \( C \in \text{Sup}[S] \) is the furthest point from \( D \). Let \( S' \) be the positions of the points in \( S \) after moving them a distance \( d \leq |AD| \) towards the point \( D \). Then,

\[
\text{Radius}[S'] \leq \begin{cases} 
\text{Radius}[S] - d/2, & D \in \mathcal{M}C(S) \\
\text{Radius}[S], & \text{otherwise}
\end{cases}
\]

Proof. (Lemma 9.4.3) Let \( K \) and \( r_S \) be the center and radius of \( \mathcal{M}C(S) \) respectively and adopt a coordinate system which places \( K \) at the origin and oriented such that the line segment \( KD \) is along the positive \( x \)-axis. Then \( D = (D_x, 0) \) and, for any point \( C \) on the border of \( \mathcal{M}C(S) \) we have \( C = r_S(\cos \theta, \sin \theta) \) where \( \theta \) is the angle between \( KC \) and the \( x \)-axis.

Define the point \( C' \) as the point obtained by moving the point \( C \) a distance \( d \leq |AD| \) towards \( D \). We can write \( C'_x = r_S \cos \theta + \frac{d}{|AD|}(D_x - r_S \cos \theta) = r_S \cos \theta \left(1 - \frac{d}{|AD|}\right) + \frac{dD_x}{|AD|}, \)
and, \( C'_y = r_S \sin \theta - \frac{d}{|AD|} \cdot r_S \sin \theta = r_S \sin \theta \left(1 - \frac{d}{|AD|}\right) \). Observe that the equations for \( C'_x \), and \( C'_y \) describe a parametric curve that is completely contained within a circle of
radius \( r_S(1 - d/|AD|) \) \( \leq r_S \). Thus, for any point \( D \), we can conclude that \( \text{Radius}[S'] \leq r_S(1 - d/|AD|) \) \( \leq r_S \).

Now consider the case that \( D \) is inside \( \mathcal{MC}(S) \). In this case \(|AD| \leq 2r_S\) such that the curve defined by \( C'_x \), and \( C'_y \) is completely contained within a circle of radius \( r_S(1 - d/|AD|) \) \( \leq (r_S - d/2) \).

**Proof.** (Theorem 9.4.2) Consider the \textsc{Gather} \((n, F)\) problem for the input \( S \), let \( N \) be the subset of \( S \) that contains only reliable robots, and let \( f \) be the (actual) number of byzantine robots in \( S \).

Let \( S^{(i)} \) and \( N^{(i)} \) represent the unique positions of the robots in \( S \) and \( N \) after the \( i^{th} \) iteration of the algorithm, and let \( r_i = \text{Radius}[N^{(i)}] \). We also let \( D_i \) be the midpoint and \( d_i \) be half the distance between the closest pair of points in \( S^{(i)} \). Finally, set \( C_i \in \text{Sup}[N^{(i)}] \) be the furthest point from \( D_i \).

Now, if in the \( i^{th} \) iteration the midpoint \( D_i \) lies within \( \mathcal{MC}(N^{(i)}) \) then by Lemma 9.4.3 we have \( r_{i+1} \leq r_i - d_i/2 \). If we assume that their are \( m \) iterations of this kind then the time needed to complete these iterations is at most \( T_m = \sum_{i=0}^{m} d_i \leq 2 \sum_{i=0}^{m} (r_i - r_{i+1}) \). However, observe that \( \sum_{i=0}^{m} r_i = r_0 + \sum_{i=1}^{m} r_i = r_0 + \sum_{i=0}^{m-1} r_{i+1} \) such that \( T_m \leq 2(r_0 - r_{m+1}) \) \( \leq 2r_0 \).

If \( D_i \) does not lie within \( \mathcal{MC}(N^{(i)}) \), then we can only say that \( r_{i+1} \leq r_i (1 - d_i/|C_i D_i|) \leq r_i \). However, observe that Algorithm 19 always gathers the two closest robots in \( S^{(i)} \) and we know that there is at least one pair of robots in \( N^{(i)} \) with separation no greater than \( 2r_i \). This tells us that \( d_i \leq r_i \). Furthermore, since all reliable robots are, by definition, within \( \mathcal{MC}(N) \), it is impossible for \( D_i \) to simultaneously be: a) the midpoint of two reliable robots, and, b) lie outside of \( \mathcal{MC}(N) \). This implies that this type of iteration can occur at most \( f \) times (as it will happen at most once for each of the byzantine robots). Thus, the time needed to complete these iterations is at most \( T_f = f \cdot r_0 \).

Combining \( T_m \) and \( T_f \) gives us a bound on the total time necessary to complete the algorithm. We get \( T \leq T_m + T_f = f r_0 + 2r_0 = (f + 2)r_0 \). The bound on the competitive ratio follows from the fact that \( f \leq F \), and \( r_0 = \text{Radius}[N] \) is the minimal time necessary to gather the robots in \( N \).

The computational complexity bound follows from the fact that we need to determine the closest pair of points \( O(n) \) times. \( \square \)

In the case that we have no knowledge of the number of byzantine robots in our collection (i.e. \( F = n - 2 \)) the algorithm has a worst-case bound on the competitive ratio of \( n \). This reflects the fact that an adversary, if allowed, would always choose \( f = F \) robots in \( S \) to be byzantine. It is worth noting, however, that it was not necessary to know \( F \) in the proof of Theorem 9.4.2 and thus the algorithm has a competitive ratio that is bounded by the actual number of byzantine robots in \( S \). That is, for a particular instance \( N \subseteq S \) such that \( f = |S| - |N| \) we have \( \text{CR}(N) \leq f + 2 \leq F + 2 \).
9.5 Conclusion

In this paper we analyzed the gathering problem for $n > 2$ robots in the plane at most $F$ of which, $F \leq n - 2$, are byzantine. The robots were equipped with GPS and they could communicate their positions to a central authority. Several algorithms were designed with competitive ratio depending on the number of byzantine robots and the knowledge available to the robots.

In addition to improving the competitive ratio and/or complexity of our algorithms, several interesting open problems remain. In particular, one could consider models that allow the robots to communicate/exchange their positions at any time during the gathering process. Additionally, it would be interesting to consider robot gathering (in the presence of byzantine robots) under local (limited) communication range.

References


Chapter 10

Gathering and Election by Mobile Robots in a Continuous Cycle

The last paper included in the thesis is another gathering problem, however, in this case randomization is used in order to achieve convergence of the agents. The included paper “Gathering and Election by Mobile Robots in a Continuous Cycle” is a conference paper appearing in the proceedings of ISAAC2019.
Gathering and Election by Mobile Robots in a Continuous Cycle

Paola Flocchini\textsuperscript{1}, Ryan Killick\textsuperscript{2}, Evangelos Kranakis\textsuperscript{2}, Nicola Santoro\textsuperscript{2}, Masafumi Yamashita\textsuperscript{3}

\textsuperscript{1}School of Electrical Eng. and Comp. Sci., University of Ottawa, Ottawa, Ontario, Canada
\textsuperscript{2}School of Computer Science, Carleton University, Ottawa, Ontario, Canada
\textsuperscript{3}Dept. of Comp. Sci. and Comm. Eng., Kyushu University, Motooka, Fukuoka, 819-0395, Japan

Abstract

Consider a set of \( n \) mobile computational entities, called \textit{robots}, located and operating on a continuous cycle \( \mathcal{C} \) (e.g., the perimeter of a closed region of \( \mathbb{R}^2 \)) of arbitrary length \( \ell \). The robots are identical, can only see their current location, have no location awareness, and cannot communicate at a distance. In this weak setting, we study the classical problems of \textit{gathering} (\textsc{GATHER}), requiring all robots to meet at a same location; and \textit{election} (\textsc{ELECT}), requiring all robots to agree on a single one as the “leader”. We investigate how to solve the problems depending on the amount of knowledge (exact, upper bound, none) the robots have about their number \( n \) and about the length of the cycle \( \ell \). Cost of the algorithms is analyzed with respect to time and number of random bits. We establish a variety of new results specific to the continuous cycle – a geometric domain never explored before for \textsc{GATHER} and \textsc{ELECT} in a mobile robot setting; compare Monte Carlo and Las Vegas algorithms; and obtain several optimal bounds.

10.1 Introduction

10.1.1 The Framework

Consider a distributed system composed of a set \( \mathcal{R} \) of autonomous mobile computational entities, called \textit{robots}, located and operating in an Euclidean space \( \mathcal{U} \). The robots are identical: without identifiers or distinguishing features, they have the same capabilities and execute the same algorithm. Although autonomous, their goal is to collectively perform some assigned system task or to solve a given problem. Among the important tasks and problems are: \textit{gathering} (\textsc{GATHER}), requiring all robots to meet at a same location; and \textit{election} (\textsc{ELECT}), requiring all robots to agree on a single one as the “leader”. Indeed, \textsc{GATHER} is
one of the fundamental problems in theoretical mobile robotics, while ELECT is typically solved as an intermediate step in the resolution of many important problems, in particular pattern formations. Both GATHER and ELECT have been extensively investigated under a variety of assumptions on the capabilities of the robots (e.g., memory, communication, visibility, orientation, speed), on the space in which they operate, and on the power of the adversary. From the point of view of the behaviour of the robots, the two main models are Look-Compute-Move (LCM) and Continuous Time (CT). In LCM the robots operate by cycling through three separate processes: observing the space (Look), executing the algorithm to determine a destination (Compute), and moving towards it (Move). In CT the robots are permanently active and continuously performing all three processes. For a recent overview see [97] and the chapters therein.

In all investigations, in both models, the theoretical concern is to identify the weakest possible conditions that make the problems solvable.

In this paper, we consider GATHER and ELECT by identical robots when the space $U$ is a continuous cycle $C$ (e.g., the perimeter of a closed region of $\mathbb{R}^2$). This spatial setting has been investigated in the LCM model with respect to the scattering problem, requiring identical robots to place themselves at uniform distance along the cycle [99]. In the CT model, a continuous cycle has been studied in the context of solving patrolling when the robots are identical [80] and when they have different motorial capabilities [58]; gathering has also been investigated, but only with robots having different motorial capabilities [121].

We study GATHER and ELECT in the CT model in a very weak computational setting: the identical robots can only see their current location and have no location awareness; furthermore they cannot communicate at a distance (i.e., communication is possible only between robots located at the same point at the same time).

It is immediate to observe that, in our setting, both problems are deterministically unsolvable: there is no deterministic algorithm that, in all possible executions of the algorithm by the robots and regardless of the initial position of the robots in the cycle, will always correctly solve the problem within finite time. This is obvious in the case of ELECT because, to render a single robot uniquely different from all others it requires the existence of some asymmetry in the system (e.g., in the initial placement of the robots, in shape of the Euclidean space) if no difference is present among the robots (e.g., distinct ids, different speeds). In our setting the impossibility holds also for GATHER, which does not have such a stringent requirement, and can sometimes be deterministically solved in absence of asymmetries and differences among the robots (e.g. [44]). Further observe that, since visibility is limited to the current robot’s location, in our setting both problems are deterministically unsolvable even if the initial configuration is asymmetric, and the robots are aware of this fact. Summarizing, the only possible solution algorithms are randomized ones.
10.1.2 Main Contributions

In this paper we start the investigation of solving GATHER and ELECT by the set of robots $R$ deployed in a continuous cycle $C$. Since GATHER is of easy resolution once a leader has been elected, we primarily focus on ELECT.

We propose both Las Vegas and Monte Carlo decentralized election protocols where: a Las Vegas algorithm correctly terminates with probability one in an unpredictable amount of time; a Monte Carlo algorithm has a fixed termination time but pays for this determinism with a positive – yet bounded – probability that it has terminated incorrectly. In other words, a Las Vegas algorithm “gambles with resources” and a Monte Carlo algorithm “gambles with correctness”.

We evaluate the complexity of the proposed algorithms with respect to two cost measures: the time until the algorithm terminates, and the total number of random bits (coin flips) required by the algorithm. The costs depend not only on the length $\ell$ of the cycle and the number $n$ of mobile robots (note that $n$ can be arbitrarily larger than $\ell$), but also and more importantly on the knowledge (none, exact, upper bound) the robots have on $\ell$ and/or $n$.

We establish several results. In particular, we prove that, with knowledge of $\ell$, a leader can be elected with probability one in optimal time with an optimal number of random bits, even without any knowledge of (an upper bound on) $n$. If only an upper bound $L = O(\ell)$ is known, then a leader can be elected with high probability in optimal time with an optimal number of random bits, even without any knowledge of (an upper bound on) $n$.

The results of the paper are summarized in Tables [10.1] and [10.2]. As we are analyzing randomized algorithms, the cost measures are often random variables; when this is the case, we give both the value achieved in the average and that with high probability.

Table 10.1: Results according to the knowledge of the robots (“Ex.” = exact, “-” = no knowledge, “UB” = upper bound). $T_{\exp}$ (resp. $B_{\exp}$) represents the expected time (resp. random-bit) complexity. The column “Type” gives the type of randomized algorithm (LV = Las Vegas, MC = Monte Carlo). The last column gives the corresponding algorithm label in the text. When an upper bound on $\ell$ (resp. $n$) is known it is represented by $L = O(\ell)$ or $N = O(n)$; and the constructed upper bound on $n$ is $\tilde{N} = \frac{L n}{\ell}$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\ell$</th>
<th>$T_{\exp}$</th>
<th>$B_{\exp}$</th>
<th>Type</th>
<th>Algo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex.</td>
<td>UB</td>
<td>$O(L)$</td>
<td>$O(n)$</td>
<td>LV</td>
<td>A20</td>
</tr>
<tr>
<td>Ex.</td>
<td>-</td>
<td>$O(n+\ell)$</td>
<td>$O(n+n \log \frac{\ell}{n})$</td>
<td>LV</td>
<td>A20 + A26</td>
</tr>
<tr>
<td></td>
<td>Ex.</td>
<td>$O(\ell)$</td>
<td>$O(n)$</td>
<td>LV</td>
<td>A20 + A25</td>
</tr>
<tr>
<td>UB</td>
<td>UB</td>
<td>$O(L)$</td>
<td>$O(n)$</td>
<td>MC</td>
<td>A22</td>
</tr>
<tr>
<td>UB</td>
<td>-</td>
<td>$O(N + N \cdot \ell/n)$</td>
<td>$O(n+n \log \frac{\ell}{n})$</td>
<td>MC</td>
<td>A22 + A26</td>
</tr>
<tr>
<td></td>
<td>UB</td>
<td>$O(L)$</td>
<td>$O(n)$</td>
<td>MC</td>
<td>A22 + A27</td>
</tr>
</tbody>
</table>
Table 10.2: Same as Table 10.1 for time and bit complexities with high probability.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \ell )</th>
<th>( T_{whp} )</th>
<th>( B_{whp} )</th>
<th>Type</th>
<th>Algo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex.</td>
<td>UB</td>
<td>( O(L \log n) )</td>
<td>( O(n \log n) )</td>
<td>LV</td>
<td>A20</td>
</tr>
<tr>
<td>Ex.</td>
<td>-</td>
<td>( O(n + \ell \log n) )</td>
<td>( O(n \log n + n \log \lceil \ell/n \rceil) )</td>
<td>LV</td>
<td>A20 + A26</td>
</tr>
<tr>
<td>-</td>
<td>Ex.</td>
<td>( O(\ell \log n) )</td>
<td>( O(n \log n) )</td>
<td>LV</td>
<td>A20 + A25</td>
</tr>
<tr>
<td>UB</td>
<td>UB</td>
<td>( O(L \log N) )</td>
<td>( O(n \log N) )</td>
<td>MC</td>
<td>A22</td>
</tr>
<tr>
<td>UB</td>
<td>-</td>
<td>( O(N + N \cdot \ell/n \cdot \log N) )</td>
<td>( O(n \log n + n \log \lceil \ell/n \rceil) )</td>
<td>MC</td>
<td>A22 + A26</td>
</tr>
<tr>
<td>-</td>
<td>UB</td>
<td>( O(L \log N) )</td>
<td>( O(n \log N) )</td>
<td>MC</td>
<td>A22 + A27</td>
</tr>
</tbody>
</table>

The paper is organized as follows. We first consider the case when the robots have some level of knowledge (exact or upper bound) of both parameters (Section 10.3). We prove that, when the robots possess knowledge of \( n \), the knowledge of an upper bound \( L = O(\ell) \) allows for a LV solution which is optimal with respect to both complexity measures. In case the robots know only upper bounds on both \( n \) and \( \ell \), we give a Monte Carlo algorithm. In Section 10.4 we consider the cases when the robots have no knowledge (exact nor upper bound) of one of the two parameters. In these cases we provide Las Vegas algorithms by which the robots can obtain knowledge of the unknown parameter efficiently, and subsequently elect a leader using the algorithms of Section 10.3. In Section 10.5 we demonstrate that unless the robots know \( n \) and/or \( \ell \) exactly, a Las Vegas algorithm cannot exist that solves ELECT. Extensions, including the solutions for GATHER using the results for ELECT, and open questions are discussed in Section 10.6.

### 10.1.3 Related work

There exists an extensive literature on problem solving by \( n \) identical mobile robots in continuous spaces, both within the distributed computing and the control communities; e.g., see the books [39, 98, 97]. In distributed computing, the problem of gathering identical robots has been the focus of intensive investigations under a variety of assumptions on the computational power and communication capabilities of the robots (e.g., [44, 47, 101, 139]). Similarly, the problem of electing a leader and its relationship to asymmetry has been observed, investigated and discussed when studying solvability of a variety of problems by autonomous mobile robots, in particular pattern formations (e.g., [90, 100, 107]). Indeed, a great deal of research has been devoted to the link between degree of symmetries and deterministic problem solving; see [97] and chapters therein for a recent account, in particular [159]. Almost all of this work is on deterministic solutions, with few exceptions (e.g., [115]).

Robots operating specifically in a continuous cycle have been studied in the context of rendezvous and gathering, but only with robots having different motorial capabilities [94].
Other investigated problems in a continuous cycle are: *patrolling*, studied both when the robots are identical and when they have different motorial capabilities (e.g. see [58, 69, 80]); and *scattering*, where the robots must place themselves at uniform distance on the cycle [99].

The geometric continuous settings in which the mobile entities can move freely are in general more suitable than discrete settings for distributed computing applications in robotics [39]. This is further enforced by the fact that after a system shut-down in a robot application the participating robots cannot be guaranteed to occupy the vertices of a graph but rather might be placed at arbitrary locations in the underlying geometric domain.

Settings of identical *mobile entities* operating in *discrete spaces* (i.e., in graphs) are extremely important as they naturally describe a wide variety of computational environments, including networked systems supporting mobile software agents, and ad-hoc wireless networks. In these settings, the analogue of a set of mobile robots in a continuous cycle is a set of identical *mobile agents* in a *ring* of identical nodes. Interestingly, this discrete setting has been extensively studied, especially for rendezvous and gathering; e.g., see the monograph [133]. In absence of distinct features of the agents and of the nodes (e.g., ids, markers, tokens), solutions are necessarily randomized, and their development has been the object of several investigations. In particular Ooshita et al. studied the gathering problem in anonymous unidirectional ring networks for multiple (mobile) agents with limited knowledge and characterized the relation between probabilistic solvability and termination detection [142]. Izumi et al. investigated the feasibility of polynomial-expected-round randomized gathering for $n$ robots and show that any randomized algorithm has $\Omega(\exp(n))$ expected-round lower bound [123].

In the computational universe of *static* (or *stationary*) *entities* connected via a communication network (i.e. the traditional message-passing universe in distributed computing), the computational entities coincide with the network nodes (i.e., the nodes are the active agents). Note that, in this universe, the problem *GATHER* does not exist; on the other hand, *ELECT* is a fundamental problem. When the entities are identical, the system is known as an *anonymous network*, and several researchers have focused on computing in an *anonymous ring* (e.g., [11, 12, 96]). The problem of electing a leader in an anonymous network, known also as *symmetry breaking* and for which clearly only probabilistic solutions exist, has been investigated in an anonymous ring network (e.g., [16, 104, 122]). In particular, Itai and Rodeh proposed probabilistic algorithms for both the synchronous and asynchronous case; they considered both cases when the size of the ring may be either known or unknown to the nodes and studied its impact on termination with a nonzero probability [122].

Interestingly, of all the related work, the one closest in spirit to our investigation is that of symmetry breaking in an anonymous ring, in spite of the fact that the computational universes are completely different: static entities and discrete space in one while mobile entities and continuous space in ours.
10.2 Model

Let $R$ be a set of $n \geq 2$ autonomous mobile computational entities, called robots, located in a continuous cycle $C$ (e.g., the perimeter of a closed region of $\mathbb{R}^2$) of real length $\ell$ in arbitrary and pairwise distinct positions.

The robots are identical: without identifiers or distinguishing features, they have the same (computational, motorial and communication) capabilities and execute the same algorithm. We assume that all robots move at speed one. Each robot $r \in R$ has a local memory composed of a finite set of registers, including a special register $\text{state}(r)$ which stores the current state of $r$; initially, the content of the memory of every robot is the same. Each robot is in possession of a fair coin which outputs H or T each with probability $\frac{1}{2}$. At any time a robot may flip its coin and base a decision on the outcome of that flip. For a robot $r$ we will use the notation $b(r)$ to represent a special register which always contains the outcome of its most recent coin-flip. We will use the notation $b(r) \leftarrow \text{flip()}$ to represent the action of flipping a coin and assigning the outcome to $b(r)$.

The robots can only see their current location and have no location awareness. Furthermore they cannot communicate at a distance; that is, communication is possible only between robots located at the same point at the same time (face-to-face). A robot may move along $C$ in either the $\text{CW}$ (clockwise) or $\text{CCW}$ (counter-clockwise) direction and may stop and/or reverse its direction of movement at any time. For simplicity, we will assume that the robots have consistent orientations and argue in Section 10.6 why this assumption is not necessary.

The robots are permanently active and continuously performing three processes: executing the algorithm (which might require flipping a coin), moving in a given direction or not at all (if so prescribed by the algorithm), and communicating with co-located robots. A robot can distinguish among its co-located robots and is able to instantaneously exchange any amount of information with each of them. When two robots moving in opposite directions meet, or a moving robot meets a stopped robot, the two robots become co-located; we call this an encounter. During an encounter, one of the robots can decide to merge with the other, thereby committing itself to following all actions of the robot it has merged with. As a result of this process, robots will form robot stacks with the head of the stack the only robot actively participating in an algorithm (the stack acts as a single robot). A robot $r$ will keep track of the number of robots present in its stack in a special register denoted by $\text{CNR}(r)$.

We assume a fully synchronous system in the following sense. Each robot possesses an identical copy of the same clock and each robot can use their respective clocks to measure arbitrarily small intervals with respect to the same unit of time (which we may take to be 1 without loss of generality). All robots will begin an algorithm at the same moment and all robots move with the same speed (which we may also take to be 1 without loss of generality). This implies that robots can fix a unit length as the distance traveled in one unit of time.

We study how such robots can solve $\text{ELECT}$ and $\text{GATHER}$, and at what cost. The
election problem, ELECT, requires the robots to transition from an initial configuration where each robot is in an identical state, to one where a single robot can be uniquely distinguished from the others. When solving this problem, we will assume the robots can be found in one of the three states CANDIDATE, FOLLOWER, or LEADER. The gathering problem, GATHER, requires the robots to transition from an initial configuration where each robot is in an identical state, to one where all robots are co-located and will no longer move. Since GATHER is of easy resolution once a leader has been elected, we primarily focus on ELECT.

We distinguish between two types of randomized algorithms: those of the Las Vegas type and those of the Monte Carlo type [14][17]. An algorithm is of the Las Vegas type, if, for any problem instance, it is correct when it terminates and it terminates with probability 1. In contrast, an algorithm is of the Monte Carlo type if, for any problem instance, it always terminates and it is correct with a probability \( p \) which is bounded away from zero.

The costs of a solution algorithm are evaluated with respect to two measures: 1) time complexity – the time until the algorithm terminates; and 2) random-bit complexity – the total number of random bits/coin flips used by the algorithm. The costs depend not only on the system parameters, the length \( \ell \) of the cycle and the number \( n \) of mobile robots, but also and more importantly on the type of knowledge available to the robots about the values of those parameters. As we are analyzing randomized algorithms, these complexity measures will often be random variables. When this is the case, we will give the value achieved in the average and with high probability.

### 10.3 Election with knowledge of both \( n \) and \( \ell \)

In this section we consider ELECT when the robots possess knowledge of both \( n \) and \( \ell \) (either exact or upper bounds). We begin with the case that the robots have exact knowledge. Pseudocode for all algorithms can be found in the appendix.

#### 10.3.1 Exact knowledge of \( n \) and \( \ell \)

**Theorem 10.3.1.** Let \( n \) and \( \ell \) be known to the robots. There is a Las Vegas algorithm solving ELECT which terminates in time \( O(\ell) \) on average and in time \( O(\ell \log n) \) with high probability; and requires \( O(n) \) random bits on average and \( O(n \log n) \) with high probability.

The proof is based on the algorithm ELECTLV\((n, \ell)\). This algorithm is formally described as Algorithm [20] and takes as inputs the number of robots \( n \) and the length of the cycle \( \ell \). Initially all robots begin in the same CANDIDATE state and each robot \( r \) has \( \text{CNR}(r) \) set to 1. The algorithm proceeds in a series of rounds beginning with the round \( t = 0 \). In each round the CANDIDATE robots will run the procedure ELECTROUND\((D)\)
with input $D_t = \min\{\frac{\ell}{2}, \frac{\ell}{n}(4/3)^t\}$, the result of which is that a subset of the robots merge and enter the FOLLOWER state. This will continue on until only a single CANDIDATE robot remains with a stack containing all $n$ robots. As the robots know the value of $n$, this last remaining robot will know it is the last and will thus enter the LEADER state.

The procedure $\text{ELECTIONROUND}(D)$ is formally described as Algorithm 21. The idea of this procedure is as follows. Each robot begins by flipping a coin. Those that flip $T$ will remain stationary for a time $4D_t$. Those that flip $H$ will: move CCW a distance $D_t$; return to their initial positions; move CW a distance $D_t$; and again return to their initial positions. If ever it occurs that a robot $r$ who flipped $H$ encounters a robot $s$ who flipped $T$ then $s$ will merge with $r$ and $r$ will update the value of $\text{CNR}(r)$ to reflect this.

We begin our analysis by determining how effective the procedure $\text{ELECTIONROUND}(D)$ is at reducing the number of candidates. This will be the subject of the next two lemmas.

**Lemma 10.3.2.** Let $n$ and $n'$ respectively represent the number of CANDIDATE robots before and after $\text{ELECTIONROUND}(D)$ is run with input $D > 0$. Then $E[n'] \leq \frac{n}{2} + \frac{1}{2}\left[\frac{\ell}{2D}\right]$.

**Proof.** Partition the cycle into $m = \left\lceil \frac{\ell}{2D} \right\rceil$ disjoint intervals such that each interval has length $\frac{\ell}{m} \leq 2D$. For each $i \in [1, m]$ let $n_i$ and $n'_i$ respectively represent the number of CANDIDATE robots contained in the $i$th interval at the beginning and end of $\text{ELECTIONROUND}(D)$. Then it is clear that $n = \sum_{i=1}^{m} n_i$ and $n' = \sum_{i=1}^{m} n'_i$. This latter expression allows us to write the expectation of $n'$ as follows:

$$E[n'] = \sum_{i=1}^{m} E[n'_i] = \sum_{i=1}^{m} \sum_{x=1}^{n_i} x \Pr[n'_i = x].$$  \hfill (10.1)

To determine the probability $\Pr[n'_i = x]$ consider the $i$th interval which initially contains $n_i > 0$ CANDIDATE robots. If at least one of these $n_i$ robots flipped $H$ then the number of them that will remain CANDIDATE is exactly the number of them that flipped $H$. Thus, if we let $k_i$ represent the random variable which counts the number of CANDIDATE robots that flipped $H$ in an interval $i$ then we can conclude that $\Pr[n'_i = x | k_i \geq 1] = \frac{1}{x}$ if $x = k_i$ and 0 otherwise. For $x \in [1, n_i]$ this implies that $\Pr[n'_i = x] = \sum_{j=0}^{n_i} \Pr[n'_i = x | k_i = j] \Pr[k_i = j]$ or $\Pr[n'_i = x] = \Pr[k_i = x] + \Pr[n'_i = x | k_i = 0] \Pr[k_i = 0]$. Using this expression for $\Pr[n'_i = x]$ we find that $E[n'_i] = \sum_{x=0}^{n_i} x \Pr[k_i = x] + \sum_{x=0}^{n_i} x \Pr[n'_i = x | k_i = 0] \Pr[k_i = 0]$.

It is not hard to see that $k_i$ is binomially distributed with parameters $n_i$ and $p = 1/2$ implying that $\sum_{x=0}^{n_i} x \Pr[k_i = x] = n_i/2$, and that $\Pr[k_i = 0] = (1/2)^{n_i}$. The sum $\sum_{x=0}^{n_i} x \Pr[n'_i = x | k_i = 0]$ represents the expected number of CANDIDATE robots surviving in an interval $i$ given that they all flipped $T$. Clearly this expectation is bounded by $n_i$ and we can thus conclude that $E[n'_i] \leq \frac{n_i}{2} + n_i \left(\frac{1}{2}\right)^{n_i} \leq \frac{n_i}{2} + \frac{1}{2}$.

To bound the expectation of $n'$ we can substitute this inequality into (10.1) to get $E[n'] = \sum_{i=1}^{m} E[n'_i] \leq \sum_{i=1}^{m} \left(\frac{n_i}{2} + \frac{1}{2}\right) = \frac{n}{2} + \frac{m}{2}$ where we have used the fact that $n = \sum_{i=1}^{m} n_i$ in the last step. Since $m = \left\lceil \frac{\ell}{2D} \right\rceil$ the lemma follows. \qed

276
Lemma 10.3.3. Let \( n_t \) count the number of CANDIDATE robots remaining in round \( t \geq 0 \) of \textsc{ElectLV}(n, \ell). Then \( E[n_t] \leq \left\lceil \left( \frac{3}{4} \right)^t n \right\rceil \).

Proof. The proof is by induction on \( t \). The base case \( t = 0 \) is clearly true. We assume that the claim holds up to \( t = k \). Using the induction hypothesis and Lemma 10.3.2 we can write \( E[n_{k+1}] \leq \frac{1}{2} \left\lceil \left( \frac{3}{4} \right)^k n \right\rceil + \frac{1}{2} \left[ \frac{\ell}{2D_k} \right] \) where \( D_k = \min \left\{ \frac{\ell}{2}, \left( \frac{4}{3} \right)^t \right\} \). The lemma clearly holds if \( D_k \geq \frac{\ell}{2} \). If this is not the case then \( D_k = \left( \frac{4}{3} \right)^k \) and again it is easy to see that the lemma holds.

In the next three lemmas (Lemma 10.3.4, Lemma 10.3.5, and Lemma 10.3.6) we bound the number of rounds, time, and random-bits required until only a single candidate robot remains. In order to do so we will employ a useful theorem by Karp [129] concerning the solutions of stochastic recurrence relations. This theorem is described in the appendix as Theorem 10.A.1

Lemma 10.3.4. Let \( T \) be the first round of \textsc{ElectLV}(n, \ell) in which only a single CANDIDATE robot remains. Then \( E[T] \leq \left\lceil \frac{\log_{4/3}(n)}{\log_{4/3}(n)} \right\rceil + 1 \) and, for any positive integer \( w \), \( \Pr[T \geq \left\lceil \frac{\log_{4/3}(n)}{\log_{4/3}(n)} \right\rceil + 1 + w] \leq \left( \frac{3}{4} \right)^w \frac{n}{\left(4/3\right)^{\left\lceil \log_{4/3}(n) \right\rceil}} \).

Proof. Observe that \( T = T(n) \) satisfies the stochastic recurrence relation \( T(n) = 1 + T(h(n)) \) with base condition \( T(1) = 0 \) and where the expectation of \( h(n) \) is bounded using Lemma 10.3.3, i.e., \( E[h(n)] \leq \left\lceil \frac{3}{4} n \right\rceil \). With this observation the lemma follows easily from Theorem 10.A.1

Lemma 10.3.5. Let \( \tau \) be the time required until only a single CANDIDATE robot remains in \textsc{ElectLV}(n, \ell). Then \( E[\tau] \leq 8\ell \) and, for any positive integer \( w \), \( \Pr[\tau \geq 2\ell(4 + w)] \leq \left( \frac{3}{4} \right)^w \frac{n}{(4/3)^{\left\lceil \log_{4/3}(n) \right\rceil}} \).

Proof. Set \( t_\ell \) as the first round which satisfies \( \ell/n(4/3)^t \geq \ell/2 \), i.e. \( t_\ell = \left\lceil \frac{\log_{4/3}(n/2)}{\log_{4/3}(n)} \right\rceil \). Assume that it takes \( T > t_\ell \) rounds until only one CANDIDATE robot remains. The time \( \tau \) required to complete these \( T \) rounds is \( \tau = 4\ell \sum_{t=0}^{t_\ell-1} \left( \frac{4}{3} \right)^t + 2 \sum_{t=t_\ell}^{T} \ell \leq 12\ell \left( \frac{4}{3} \right)^t + 2(T - t_\ell) \ell \leq 8\ell + 2(T - t_\ell) \). The lemma now follows from Lemma 10.3.4

Lemma 10.3.6. Let \( B \) be the random variable which counts the number of coin-flips used in \textsc{ElectLV}(n, \ell). Then \( E[B] \leq 4n \) and, for any positive integer \( w \), \( \Pr[B \geq (4 + w)n] \leq \left( \frac{3}{4} \right)^w \).

Proof. Similarly to the proof of Lemma 10.3.4 we observe \( B = B(n) \) satisfies the stochastic recurrence relation \( B(n) = n + B(h(n)) \) with base condition \( B(1) = 0 \) and where \( h(n) \) has expectation \( E[h(n)] \leq \left\lceil \frac{3}{4} n \right\rceil \). With this observation the lemma follows easily from Theorem 10.A.1
The proof of Theorem 10.3.1 now follows immediately from Lemmas 10.3.5 and 10.3.6.

10.3.2 Inexact knowledge of \( n \) and/or \( \ell \)

We now consider the cases that the robots are provided with inexact knowledge (upper bounds) of at least one of \( n \) or \( \ell \). We begin with the case that the robots know \( n \) and an upper bound on \( \ell \).

Observe that nowhere in the proof of Theorem 10.3.1 did we require the robots to know exactly the value of \( \ell \). In particular, if the robots were to instead use an upper bound \( L \) on \( \ell \) then the only change we need to make is to replace \( \ell \) with \( L \) in the time complexity. This observation thus easily leads to the following corollary of Theorem 10.3.1:

Corollary 10.3.7. Let \( n \) and an upper bound \( L \geq \ell \) be known to the robots. There is a Las Vegas algorithm solving this problem which terminates in time \( O(L) \) on average and in time \( O(L \log n) \) with high probability; and requires \( O(n) \) random bits on average and \( O(n \log n) \) with high probability.

The same argument does not work if the robots know \( \ell \) and an upper bound \( N \geq n \) since ELECTLV requires the exact value of \( n \) in order to terminate. We will see in the next section that exact knowledge of \( \ell \) however allows the robots to determine \( n \) and we will therefore postpone a discussion of this case until then.

If the robots only possess upper bounds on both \( n \) and \( \ell \) then a Las Vegas algorithm does not exist (see Section 10.5). We thus provide a Monte Carlo algorithm (Algorithm 22) to solve the problem.

Theorem 10.3.8. Let upper bounds \( N \geq n \) and \( L \geq \ell \) be known to the robots. Then, for any positive integer \( w \) there is a Monte Carlo algorithm solving ELECT with error probability \( O((3/4)^w) \). This algorithm terminates in time \( O(wL) \) and requires \( O(wn) \) random bits.

Proof. The proof is based on the algorithm ELECTMC\((N, L, w)\) which takes as inputs the upper bounds \( N \) and \( L \), and a positive integer \( w \) which controls the runtime. This algorithm is formally described as Algorithm 22. This algorithm is identical to ELECTLV\((N, L)\) except that it deterministically terminates on the round \( t_\infty = \lceil \log_{4/3}(N) \rceil + w \). We may therefore reuse many of our previously derived results. In particular, the time \( \tau \) until termination follows from the proof of Lemma 10.3.5 and is given by \( \tau = 8L + 2(w + 1)L \). The random-bit complexity follows from Lemma 10.3.6. The error probability of the algorithm is also easy to derive. In particular, if we let \( T \) be the number of rounds required until only a single CANDIDATE remains then the probability that the algorithm terminates incorrectly is simply the probability \( \Pr[T > t_\infty] = \Pr[T > \lceil \log_{4/3}(N) \rceil + w] = \Pr[T \geq \lceil \log_{4/3}(N) \rceil + 1 + w] \) and this probability is given by Lemma 10.3.4. \(\square\)
10.4 Election with knowledge of either \( n \) or \( \ell \)

In this section we investigate \textsc{ELECT} when the robots are provided with knowledge of only one of \( n \) or \( \ell \) (exact or upper bounds). In all cases we use the same strategy to solve the problem: we develop algorithms by which the robots gain knowledge of the unknown of \( n \) or \( \ell \) and then use the algorithms of the previous section to solve \textsc{ELECT}. Pseudocode for all algorithms presented can be found in the appendix.

10.4.1 Exact knowledge of \( n \) or \( \ell \)

**Theorem 10.4.1.** Let either \( n \) or \( \ell \) be known to the robots. Then there are Las Vegas algorithms solving \textsc{ELECT}. If \( \ell \) is known the algorithm terminates in time \( O(\ell) \) on average and in time \( O(\ell \log n) \) with high probability; and requires \( O(n) \) random bits on average and \( O(n \log n) \) with high probability. If \( n \) is known the algorithm terminates in time \( O(n + \ell) \) on average and in time \( O(n + \ell \log n) \) with high probability; and requires \( O(n + n \log \left\lfloor \frac{\ell}{n} \right\rfloor) \) random bits on average and \( O(n \log(n) + n \log \left\lfloor \frac{\ell}{n} \right\rfloor) \) with high probability.

As previously stated, our proof strategy is to first develop algorithms by which the robots can gain knowledge of the unknown of \( n \) or \( \ell \). More specifically, the goal of this section is to constructively demonstrate the validity of the following two lemmas from which Theorem 10.4.1 will easily follow.

**Lemma 10.4.2.** Consider \( n \) robots on a cycle of length \( \ell \) and assume the robots know only the value of \( \ell \). Then there exists a Las Vegas algorithm by which the robots can determine the value of \( n \). This algorithm terminates in time \( O(\ell) \) on average and with high probability; and requires \( O(n) \) random bits on average and with high probability.

**Lemma 10.4.3.** Consider \( n \) robots on a cycle of length \( \ell \) and assume the robots know only the value of \( n \). Then there exists a Las Vegas algorithm by which the robots can determine an \( O(\ell) \) upper bound \( L \) on \( \ell \). This algorithm terminates in time \( O(n + \ell) \) on average and with high probability; and requires \( O(n + n \log \left\lfloor \frac{\ell}{n} \right\rfloor) \) random bits on average and with high probability.

We will begin by introducing two procedures which will be used throughout the remainder of the section. The first procedure will be used by the robots to count coin flips, and the second is a minimum finding procedure.

**A procedure to count coin flips:** The procedure \textsc{CountFlips}(\( D \)) is formally described as Algorithm 23 and takes as input a distance \( D \). For simplicity in the following description we will assume that \( D = \ell \). The procedure presumes that each robot \( r \) has
flipped a coin and stored the result in \( b(r) \). It will result in each robot either knowing the total number of robots or that all robots have flipped the same thing.

At the beginning the robots that flip H will move \( \text{cw} \) a distance \( \ell \) around the cycle and count each robot they encounter which flipped T. The robots that flipped T will likewise wait for a time \( \ell \) and count each robot they encounter that flipped H. Since each moving robot makes a full traversal of the cycle they are guaranteed to see all stationary robots. Thus, after the first \( \ell \) time units, each robot will determine the number of robots which flipped opposite to themselves. In the last \( \ell \) time units of the algorithm the robots which initially flipped H (resp. T) will move \( \text{ccw} \) a distance \( \ell \) around the cycle (resp. wait for \( \ell \) time units). In either case, a robot will determine the total number of robots that flipped the same as themselves from the first robot they encounter which flipped opposite to themselves. Thus, after the first \( 2\ell \) time units each robot will have determined both the total number of robots which flipped H and the number that flipped T and from this they can compute \( n \). If all robots flipped the same thing then the robots will know this since each will have determined that \( N_H(r) = N_T(r) = 0 \). From this description it is easy to establish the following lemma:

**Lemma 10.4.4.** Assume that all robots have flipped a coin. Then in exactly \( 2\ell \) time units the procedure \( \text{COUNTFLIPS}(\ell) \) will result in either each robot knowing \( n \) or that all robots have flipped the same thing.

When an input \( D > \ell \) is used in the procedure we claim the following:

**Lemma 10.4.5.** Assume that all robots have flipped a coin and that \( D \geq \ell \). Then in exactly \( 2D \) time units the procedure \( \text{COUNTFLIPS}(D) \) will result in each robot \( r \) computing an upper bound \( N(r) \geq n \) or that all robots have flipped the same thing.

**Proof.** Clearly, if all robots flip the same then each robot will compute \( N_H(r) + N_T(r) = 0 \). Thus, assume that at least two robots flip differently. Let \( n_T \) and \( n_H \) represent the actual number of robots that flipped T and H respectively, i.e. \( n_T + n_H = n \). Since each robot that flipped H traverses the cycle at least once each such robot is guaranteed to encounter all robots that flipped T. Likewise, each robot that flipped T is guaranteed to encounter each robot that flipped H. It is therefore not possible for a robot \( r \) to compute a value of \( N_H(r) < n_H \) or \( N_T(r) < n_T \) and thus it is ensured that \( N_T(r) + N_H(r) \geq n \) for all robots. \( \square \)

Finally, if an input \( D < \ell \) is used in the procedure then we claim the following:

**Lemma 10.4.6.** Assume that all robots have flipped a coin and that \( D < \ell \). Then in exactly \( 2D \) time units the procedure \( \text{COUNTFLIPS}(D) \) will result in each robot \( r \) computing a lower-bound \( N(r) \leq n \).

**Proof.** The only thing we need to demonstrate is that all robots will compute a value \( N(r) \leq n \). Clearly, in order for this not to be true, at least one of the robots must double count another robot. This, however, is not possible unless a robot traverses the cycle more than once and this will clearly not be the case if \( D < \ell \). \( \square \)
A minimum finding procedure: The minimum finding procedure $\text{FINDMIN}(L, N_0)$ is formally described as Algorithm 24 and takes as input an upper bound $L \geq \ell$ on the cycle length, and a value $N_0$ (which is specific to each robot). The algorithm results in each robot computing the minimum of the inputs $N_0$. It assumes that all robots have flipped a coin and that at least two robots have flipped differently.

Each robot that flipped H will initially move CW a distance $L \geq \ell$ around the cycle and is guaranteed to encounter every robot that flipped T. Likewise every robot that flipped T will encounter every robot that flipped H. Thus, after the first $L$ time units, every robot that flipped H (resp. T) will know the minimum value of every robot that flipped T (resp. H). In the second $L$ time units the robots that flipped H will move CCW a distance $L$ and will again encounter every robot that had flipped T. They can thus determine the minimum value of all robots that flipped H from the first robot they encounter that flipped T. Likewise, each robot that flipped T will determine the minimum value of all robots that flipped T from the first robot they encounter that flipped H. The algorithm clearly terminates after $2L$ time units.

We can thus claim the following without proof:

**Lemma 10.4.7.** Assume that all robots have flipped a coin, at least two have flipped differently, and that $L \geq \ell$. Then in exactly $2L$ times units the procedure $\text{FINDMIN}(L, N_0(r))$ will result in each robot $r$ computing the minimum of all inputs $N_0(r)$.

**Computing $n$ using $\ell$:** We will now tackle the proof of Lemma 10.4.2 which is based off of the algorithm $\text{COUNTROBOTS}(\ell)$. This algorithm is formally described as Algorithm 25 and takes as input the length of the cycle. The idea is to repeatedly flip coins and run the procedure $\text{COUNTFLIPS}(\ell)$ until the first round in which at least two robots flip differently. When this occurs each robot will compute the total number of robots that flipped T and the total number that flipped H and will thus determine $n$ to be the sum of these values.

**Proof.** (Lemma 10.4.2) The correctness of $\text{COUNTROBOTS}(\ell)$ is obvious. The algorithm will terminate on the first round during which at least two robots flip differently. The probability that all robots flip the same is $2^{1-n}$ and therefore the algorithm terminates after an expected $\frac{1}{1-2^{1-n}} \leq 2$ rounds. The probability that the algorithm terminates after $T$ rounds is $2^{(T-1)(1-n)}(1-2^{1-n})$. From this it is clear that the algorithm terminates after $O(1)$ rounds with high probability. The time and random-bit complexities follow from the fact that each round lasts time at most $2\ell$ and in each round all $n$ robots flip their coins.

**Computing a $O(\ell)$ upper bound on $\ell$ using $n$:** The proof of Lemma 10.4.3 is based off of the algorithm $\text{BOUNDCYCLE}(n)$. This algorithm is formally described as Algorithm 26 and takes as input the number of robots on the cycle. In each round $t \geq 0$ the robots will employ the procedure $\text{COUNTFLIPS}$ in an attempt to determine a strict upper bound...
on the number of robots using an estimate $L_t = n \cdot 2^t$ for an upper bound on $\ell$. This will result in each robot $r$ computing a value $N(r)$. If $L_t < \ell$ then, by Lemma [10.4.6], the robots will each compute $N(r) \leq n$ and the algorithm will proceed to the next round. If $L_t \geq \ell$ then the robots will each compute $N(r) \geq n$ and, after performing FINDMIN, they will all agree on the computed value of $N(r)$. Let $t_*$ be the first round in which all robots compute $N(r) > n$. The corresponding value of $L_t$ in the round $t_*$ will then be an upper bound on $\ell$.

We reduce $L_t$ by a factor $\frac{1}{2} \left\lceil \frac{N(r)}{n} \right\rceil$ to ensure that the returned upper bound is $O(\ell)$.

**Proof.** (Lemma [10.4.3]) To determine the running time we let $t_0$ be the first round for which $L_t > 2\ell$. Then $t_0 = \left\lceil \log \frac{2L}{n} \right\rceil$ if $n < 2\ell$ and $t_0 = 0$ if $n \geq 2\ell$. The algorithm will certainly terminate in the first round $t_* > t_0$ in which at least two robots flip differently. Since the probability that all robots flip the same is $2^{1-n}$ we will have $t_* = t_0 + O(1)$ with high probability. The algorithm will therefore take at most $\left\lceil \log \frac{2L}{n} \right\rceil + O(1)$ rounds. Since the procedures COUNTFLIPS($L_t$) and FINDMIN($L_t$) each take time $2L_t$ to complete, each round of the algorithm lasts time $4L_t = n \cdot 2^{t+2}$. The total time required is thus $\sum_{t=0}^{t_*} n \cdot 2^{t+2} = 4n(2^{t_*+1} - 1)$. If $n > 2\ell$ then the above is clearly $O(n)$. If $n \leq 2\ell$ then we have that $4n(2^{t_*+1} - 1) = 4n \left(2^{\left\lceil \log \frac{2L}{n} \right\rceil + O(1)} - 1 \right) = O(\ell)$.

Thus, we can conclude that the algorithm terminates in time $O(n + \ell)$ on average and with high probability. In each round of the algorithm all robots flip a coin and thus the algorithm requires $O(n)$ random bits if $n > 2\ell$ and otherwise $O(n \log \left\lceil \frac{2L}{n} \right\rceil)$ when $n \leq 2\ell$. 

### 10.4.2 Inexact knowledge of $n$ or $\ell$

We now consider the cases that the robots are only provided with an upper bound on $n$ or only an upper bound on $\ell$. The main result follows:

**Theorem 10.4.8.** Let only an upper bound $L \geq \ell$ or an upper bound $N \geq n$ be known to the robots. Then, for any positive integer $w$ there are Monte Carlo algorithms solving ELECT with error probability $O((3/4)^w)$. If the robots know $L \geq \ell$ then the algorithm terminates in time $O(wL)$ and requires $O(wn)$ random bits. If the robots know $N \geq n$ then the algorithm terminates in time $O\left(\left\lceil N + w\frac{N}{n}\ell \right\rceil\right)$ and requires $O\left(\left\lceil wn + n \log \left\lceil \frac{\ell}{n} \right\rceil \right\rceil\right)$ random bits.

Our goal is again to develop algorithms by which the robots will gain knowledge of the unknown of $n$ or $\ell$ and then employ the algorithm ELECTMC to solve ELECT. We therefore want to demonstrate the following two lemmas:

**Lemma 10.4.9.** Consider $n$ robots on a cycle of length $\ell$ and assume the robots know an upper bound $L \geq \ell$. Then there exists a Las Vegas algorithm by which the robots can
determine an upper bound \( N = O\left(\frac{L}{\ell}n\right) \) on \( n \). This algorithm terminates in time \( O(L) \) on average and with high probability; and requires \( O(n) \) random bits on average and with high probability.

Lemma 10.4.10. Consider \( n \) robots on a cycle of length \( \ell \) and assume the robots know only an upper bound on the value of \( n \). Then there exists a Las Vegas algorithm by which the robots can determine an \( O\left(\frac{N}{n}\ell\right) \) upper bound \( L \) on \( \ell \). This algorithm terminates in time \( O\left(N + \frac{N}{n}\ell\right) \) on average and with high probability; and requires \( O\left(n + n \log \left\lceil \frac{\ell}{n} \right\rceil\right) \) random bits on average and with high probability.

Clearly Theorem 10.4.8 will directly follow from the above two lemmas as well as Theorem 10.3.8. We begin with the case that the robots know \( L \geq \ell \).

Computing an upper bound on \( n \) from an upper bound on \( \ell \): Here we will use an algorithm essentially identical to COUNTROBOTS(\( \ell \)) except with the addition of a FINDMIN procedure. The robots will repeatedly flip coins and run the procedure COUNTFLIPS(\( L \)) until at least two robots flip differently. At this point each robot \( r \) will know an upper bound \( N(r) \geq n \). They will then run the procedure FINDMIN(\( L, N(r) \)) in order to determine the same upper bound. The correctness of the algorithm follows easily from Lemmas 10.4.5 and 10.4.7. The fact that the robots compute a \( O\left(\frac{L}{\ell}n\right) \) upper bound follows from the fact that the robots will traverse the cycle \( \frac{L}{\ell} \) times. The asymptotic running time of the algorithm is identical to that of COUNTROBOTS with \( \ell \) replaced with \( L \). The random-bit complexity does not change. Lemma 10.4.9 follows without proof from this discussion.

Computing an upper bound on \( \ell \) from an upper bound on \( n \): Here we simply use the algorithm BOUNDCYCLE with the input \( N \geq n \) instead of \( n \).

Proof. The proof is nearly identical to that of Lemma 10.4.3 except we replace \( n \) with \( N \) and require at least \( t_0 \) rounds where \( t_0 \) is the first round in which \( L_t = N \cdot 2^t \geq 2 \left\lceil \frac{N}{n} \right\rceil \ell \), i.e.

\[
t_0 = \left\lceil \log \left( \left\lceil \frac{N}{n} \right\rceil \ell \right) \right\rceil = O\left( \log \left\lceil \frac{\ell}{n} \right\rceil \right).
\]

\hfill \Box

10.5 Impossibility results

In the previous sections we have developed Las Vegas algorithms which solve ELECT when one of \( n \) or \( \ell \) is known exactly to the robots. We have also developed Monte Carlo algorithms when only upper-bounds on \( n \) and/or \( \ell \) are known. In the sequel we demonstrate that, unless the robots know at least one of \( n \) or \( \ell \) exactly, there does not exist a Las Vegas algorithm which solves ELECT.
Theorem 10.5.1. Assume that the robots do not know $\ell$ nor $n$ exactly. Then there is no Las Vegas type algorithm which solves \textsc{ELECT}.

To demonstrate this we first prove the weaker statement that a Las Vegas algorithm cannot exist if the robots know nothing of $n$ nor $\ell$.

Lemma 10.5.2. If neither $n$ nor $\ell$ is available then there is no Las Vegas type algorithm which solves \textsc{ELECT}.

Proof. To derive a contradiction suppose that there is a Las Vegas type algorithm $A$ which solves the problem. Consider an instance $I$ in which there are two robots $r_1$ and $r_2$ at antipodal positions on a cycle with circumference $\ell$. Since $A$ solves the problem it terminates with probability 1 in a finite, though unpredictable, amount of time $T$. Let $O_1$ and $O_2$ be the sequence of outcomes of coin flips of $r_1$ and $r_2$.

Consider another instance $I'$ in which there are four robots $r_1$, $r_2$, $r'_1$, and $r'_2$ at equally spaced locations of a cycle with circumference $2\ell$ such that $r_1$ and $r'_1$ (resp. $r_2$ and $r'_2$) are antipodal (see Figure 10.1). Assume that the pair $r_1$ and $r'_1$ (resp. $r_2$ and $r'_2$) each have the same orientation and each receives the outcome of coin flips $O_1$ (resp. $O_2$). Call an encounter between a pair of robots $r_1$ and $r_2$ a left encounter (resp. a right encounter) if $r_1$ and $r_2$ encounter each other while either $r_1$ is moving CCW and $r_2$ is stationary, $r_2$ is moving CW and $r_1$ is stationary, or $r_1$ is moving CCW and $r_2$ is moving CW (resp. while either $r_1$ is moving CW and $r_2$ is stationary, $r_2$ is moving CCW and $r_1$ is stationary, or $r_1$ is moving CW and $r_2$ is moving CCW). Then for every left encounter of $r_1$ and $r_2$ in $I$ there is a corresponding identical left encounter between $r_1$ and $r'_2$ in $I'$ and between $r_2$ and $r'_1$ in $I'$. Likewise, for every right encounter of $r_1$ and $r_2$ in $I$ there are corresponding identical right encounters between $r_1$ and $r'_2$ in $I$ and between $r_2$ and $r'_1$ in $I'$. Thus, at time $T$, each of $r_1$ and $r'_1$ (resp. $r_2$ and $r'_2$) in $I'$ must come to the same conclusion as $r_1$ (resp. $r_2$) in $I$. However, this implies that at the end of the execution of $A$ in $I'$ we will have elected two leaders. Since there is a positive probability that $r_1$ and $r'_1$ (resp. $r_2$ and $r'_2$) both get the outcome of coin flips $O_1$ (resp. $O_2$) then there is a positive probability that $A$ incorrectly
terminates in time $T$. This contradicts our assumption that $A$ correctly terminates with probability one.

It is not hard to extend this to the situation that the robots know only an upper bound on $n$:

**Corollary 10.5.3.** Suppose that the robots only know an upper bound $N$ on $n$. Then there is no Las Vegas type algorithm which solves $\text{ELECT}$.

**Proof.** To derive a contradiction suppose that there is a Las Vegas algorithm $A$ for $\text{ELECT}$. We use the instances $I$ and $I'$ given in the proof of Theorem 10.5.1. Provided that $N = 5$ is given, consider the execution of $A$ for $I$. Then in time $T$, $A$ terminates in which $O_1$ and $O_2$ are the sequences of outcomes of the coin flips of $r_1$ and $r_2$.

Then $A$ terminates incorrectly in time $T$, when it is executed for $I'$ with $N = 5$, as argued in the proof of Lemma 10.5.2 which is a contradiction.

**Proof.** (Theorem 10.5.1) Assume that a Las Vegas algorithm $A$ exists by which the robots can solve $\text{ELECT}$ if they know upper bounds $N$ and $L$ on $n$ and $\ell$ respectively. Now consider an instance of the problem when only an upper bound $N$ on $n$ is known. Then by Lemma 10.4.10 there exists a Las Vegas algorithm by which the robots can determine $L$. Once the robots know $L$ they run algorithm $A$ to elect a leader. This implies that there exists a Las Vegas algorithm by which the robots can elect a leader when they only know an upper bound $N$ on $n$. This contradicts the previous result of Corollary 10.5.3 which states that such an algorithm cannot exist. We may therefore conclude that a Las Vegas algorithm does not exist if the robots know both upper bounds $N$ and $L$. This further implies that a Las Vegas algorithm does not exist when the robots know only $L$.

### 10.6 Extensions and Open Questions

Here we discuss why the consistent orientation assumption is unnecessary; the extension of our election algorithms to the $\text{GATHER}$ problem; and other extensions/open problems.

**Orientation:** In the previous sections we have assumed that the robots have consistent orientations. Here we will argue why this assumption is not required.

First, observe that with the consistent orientation assumption it will never occur that two moving robots encounter each other. By removing this assumption we will have to deal with the extra encounters involving two robots which move in opposite directions. For most of these encounters the solution is simple – the two moving robots will simply ignore each other. A more problematic encounter occurs if two moving robots encounter a stationary robot from opposite directions at the same time. Fortunately, this is also easily remedied – we simply have the stationary robot choose to “process” the moving robot arriving from its,
say, cw direction first. We can thus conclude that all of our results still hold if we remove the consistent orientation assumption.

**Gathering:** In the previous sections our primary goal has been on how to solve ELECT. However, it is easy to see that our algorithms also solve GATHER at no extra cost. Indeed, consider Algorithm 20 where, during the election process, robots only enter a FOLLOWER state when they merge with a remaining CANDIDATE robot. When only a single CANDIDATE remains all other robots will be part of its stack. This is also the case for Algorithm 22 however, since this is a Monte Carlo algorithm, there is a bounded probability that more than one stack remains when the algorithm terminates. Thus, by construction, Algorithm 20 is a Las Vegas algorithm which solves GATHER and Algorithm 22 is a Monte Carlo algorithm which solves GATHER. Clearly, the complexities of these algorithms remain the same when applied to either the ELECT or GATHER problems.

10.6.1 Discussions and Open Problems

In this paper we have studied the ELECT and GATHER problems for $n$ identical robots in the $CT$ model on a continuous cycle of length $\ell$. We have established several results including optimal algorithms with respect to time and random bits when the robots know $\ell$, or an upper bound $L = O(\ell)$ (in the latter case with high probability).

There are a number of open questions remaining. Firstly, we have not considered the possibility (or lack thereof) of a Monte Carlo algorithm when the robots do not possess any knowledge of $n$ or $\ell$. In addition, we have only considered a fully synchronous time model and a natural extension is therefore to study ELECT and GATHER when this assumption is removed. In particular one can consider a model where the robots do not begin an algorithm simultaneously but otherwise their respective clocks tick at the same rate, or a model where even the robots’ clocks are not synchronized.

References


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
<th>Venue</th>
</tr>
</thead>
</table>
10.A Description of Karp’s theorem

Consider the stochastic recurrence relation

\[ T(n) = a(n) + T(h(n)) \]  \hspace{1cm} (10.2)

which describes a process in which we start with an input of size \( n \) and after investing some amount of resources (represented by \( a(n) \)) we are left with a smaller problem of size \( h(n) \) upon which we recurse. As it applies here, \( n \) represents the number of candidate robots, \( a(n) \) will represent the number of rounds/time/random-bits, and \( h(n) \) the expected number of robots remaining after one iteration of a leader election algorithm.

Formally, \( n \) is a nonnegative integer variable; \( a(n) \) a nonnegative real-valued function of \( n \); \( h(n) \) a random variable with support \([0, n]\) and expectation bounded by \( m(n) \); and \( m(n) \) is a nonnegative real-valued function of \( n \). The equation \( \tau(n) = a(n) + \tau(m(n)) \) is the deterministic analogue of (10.2) and, when it exists, has the unique least nonnegative solution \( u(n) \) given by

\[ u(n) = \sum_{k=0}^{\infty} a(m[k](n)) \]  \hspace{1cm} (10.3)

with \( m[k](n) \) inductively defined by \( m[0](n) = n \) and \( m[k](n) = m(m[k-1](n)) \), \( k \geq 1 \).

Karp proved the following:

**Theorem 10.A.1.** (Karp [129], Theorems 1.1 and 1.2). Consider the stochastic recurrence (10.2), a continuous function \( m(n) \) with \( m(n)/n \) non-decreasing, and let \( u(n) \) be given by (10.3).

1. Suppose there is a constant \( d \) such that \( a(n) = 0, \; n < d \); and \( a(n) = 1, \; n \geq d \). Let \( c_k = \min\{n|u(n) \geq k\} \). Then, for every positive integer \( n \) and every positive integer \( w \), \( \Pr[T(n) \geq u(n) + w] \leq \left(\frac{m(n)}{n}\right)^{w-1} \frac{m(n)}{c_u(n)} \).

2. Suppose that \( a(n) \) is strictly increasing on \( \{n|a(n) > 0\} \). Then, for every positive integer \( n \) and every positive integer \( w \), \( \Pr[T(n) > u(n) + wa(n)] \leq \left(\frac{m(n)}{n}\right)^{w} \).
10.B  Pseudocode for algorithms of Section 10.3.1

**Algorithm 20** \textsc{ElectLV}(n, \ell)

**Input:** \( n > 0 \) (integer); \( \ell > 0 \) (real); \( \triangleright \) The number of robots and the length of the cycle.

**Initialize:** \( \text{state}(r) \leftarrow \text{CANDIDATE}; \text{CNR}(r) \leftarrow 1; t \leftarrow 0; \)

**Begin:**

1. \textbf{repeat}
2. \( D \leftarrow \min \left\{ \frac{\ell}{2}, \frac{\ell}{n} \left( \frac{1}{3} \right)^{t} \right\}; \)
3. \textsc{ElectionRound}(D); \( t \leftarrow t + 1; \) \( \triangleright \) Run one election round.
4. \textbf{if} \( \text{CNR}(r) = n \) \textbf{then} \( \text{state}(r) \leftarrow \text{LEADER}; \) \( \triangleright \) Stack contains \( n \) robots, terminate.
5. \textbf{until} \( \text{state}(r) = \text{FOLLOWER} \) or \( \text{LEADER} \)

:**Algorithm 21** \textsc{ElectionRound}(D)

**Input:** \( D > 0 \) (real);

**Begin:** \( b(r) \leftarrow \text{flip}(); \)

1. \textbf{if} \( b(r) = H \) \textbf{then} \( \triangleright \) H was flipped
2. \textbf{if} a robot \( s \) with \( b(s) = T \) is encountered while moving then
3. \( \text{CNR}(r) \leftarrow \text{CNR}(r) + \text{CNR}(s); \) \( \triangleright \) Update \( \text{CNR}(r) \) since \( s \) will merge with \( r \).
4. \textbf{else} \( \triangleright \) T was flipped
5. \textbf{if} a robot \( s \) with \( b(s) = H \) is encountered while waiting then
6. \( \text{state}(r) = \text{FOLLOWER}; \)
7. \textbf{if} a robot \( s \) with \( b(s) = H \) is encountered while waiting then
8. \( \text{state}(r) = \text{FOLLOWER}; \)
9. \textbf{Merge with robot} \( s; \)

:**End**
10.C  Pseudocode for algorithms of Section 10.3.2

Algorithm 22 ELECTMC$(N, L, w)$

Input: $N > 0$ (integer); $L > 0$ (real); $w > 0$ (integer); \(\triangleright\) upper bounds on $n$ and $\ell$; termination parameter $w$.

Initialize: $\text{state}(r) \leftarrow \text{CANDIDATE}$; $t \leftarrow 0$; $t_\infty \leftarrow \left\lceil \log_{4/3}(n) \right\rceil + w$; \(\triangleright\) $t_\infty$ = termination round.

Begin:
1: repeat
2: \(D_t \leftarrow \min \left\{ L, \frac{L}{N} \left(\frac{4}{3}\right)^t \right\}\);
3: \(\text{ELECTIONROUND}(D_t); t \leftarrow t + 1; \)
   \(\triangleright\) Run one election round.
4: until $\text{state}(r) = \text{FOLLOWER}$ or $t = t_\infty$
5: if $\text{state}(r) = \text{CANDIDATE}$ then $\text{state}(r) \leftarrow \text{LEADER}$;

:End

10.D  Pseudocode for algorithms of Section 10.4.1

Algorithm 23 COUNTFLIPS$(D)$

Input: $D > 0$ (real); \(\triangleright\) An estimate of the length of the cycle.

Initialize: $N_H(r) \leftarrow 0$; $N_T(r) \leftarrow 0$; \(\triangleright\) To count the robots flipping H and T.

Begin:
1: if $b(r) = H$ then \(\triangleright\) H was outcome of last coin flip
2: Move CW a distance $D$;
3: if a robot $s$ with $b(s) = T$ is encountered while moving then $N_T(r) \leftarrow N_T(r) + 1$;
4: Move CCW a distance $D$;
5: if $N_H(r) = 0$ and a robot $s$ with $b(s) = T$ is encountered while moving then \(\triangleright\) Determine $N_H$.
6: $N_H(r) \leftarrow N_H(s)$;
7: else \(\triangleright\) T was outcome of last coin flip
8: Wait for time $D$;
9: if a robot $s$ with $b(s) = H$ is encountered while waiting then $N_H(r) \leftarrow N_H(r) + 1$;
10: Wait for time $D$;
11: if $N_T(r) = 0$ and a robot $s$ with $b(s) = H$ is encountered while waiting then \(\triangleright\) Determine $N_T$.
12: $N_T(r) \leftarrow N_T(s)$;
13: return $N_H(r) + N_T(r)$; \(\triangleright\) Returns 0 if all robots flipped the same.

:End
Algorithm 24 FindMin\((L, N_0)\)

Input: \(L > 0\) (real); \(N_0\) (real); \(\triangleright\) upper bound cycle length; quantity to find the minimum of.
Initialize: \(N(r) \leftarrow N_0\); \(\triangleright\) Will contain the minimum of the inputs \(N_0\).

Begin:
1: \(\text{if } b(r) = H \text{ then} \) \(\triangleright H\) was outcome of last coin-flip
2: Move \(\text{CW}\) a distance \(L\) and then move \(\text{CCW}\) a distance \(L\);
3: \(\text{if robot } s \text{ with } b(s) = T\) is encountered \(\text{then } N(r) \leftarrow \min\{N(r), N(s)\}\);
4: \(\text{else} \) \(\triangleright T\) was outcome of last coin-flip
5: Wait for time \(2L\);
6: \(\text{if robot } s \text{ with } b(s) = H\) is encountered \(\text{then } N(r) \leftarrow \min\{N(r), N(s)\}\);
7: \(\text{return } N(r)\);
\text{:End}

Algorithm 25 CountRobots\((\ell)\)

Input: \(\ell > 0\) (real); \(\triangleright\) The length of the cycle.
Initialize: \(N(r)\); \(\triangleright\) Will contain the computed value of \(n\).

Begin:
1: \(\text{repeat} \)
2: \(\text{\hspace{1em}} b(r) \leftarrow \text{flip}(); \text{\hspace{1em}} N(r) \leftarrow \text{CountFlips}(\ell)\);
3: \(\text{\hspace{1em}} \text{until } N(r) > 0\)
4: \(\text{\hspace{1em}} \text{return } N(r)\);
\text{:End}

Algorithm 26 BoundCycle\((n)\)

Input: \(n > 0\) (integer); \(\triangleright\) The number of robots.
Initialize: \(N(r); t \leftarrow -1\);

Begin:
1: \(\text{repeat} \)
2: \(\text{\hspace{1em}} t \leftarrow t + 1\);
3: \(\text{\hspace{1em}} L_t = n \cdot 2^{t-1}\);
4: \(\text{\hspace{1em}} b(r) \leftarrow \text{flip}()\);
5: \(\text{\hspace{1em}} N(r) \leftarrow \text{CountFlips}(L_t)\);
6: \(\text{\hspace{1em}} N(r) \leftarrow \text{FindMin}(L_t, N(r))\);
7: \(\text{\hspace{1em}} \text{until } N(r) > n\)
8: \(\text{\hspace{1em}} \text{return } \frac{2L_t}{\lceil N(r)/n \rceil}\);
\text{:End}
10.E  Pseudocode for algorithms of Section 10.4.2

Algorithm 27 BOUNDROBOTS(L)

Input:  \( L > 0 \text{, real} \)  \( \triangleright \) upper bound on the length of the cycle.
Initialize:  \( N(r) \);  \( \triangleright \) Will contain the computed upper bound on \( n \).

Begin:
1:  \( \text{repeat} \)
2:  \( b(r) \leftarrow \text{flip}(); \quad N(r) \leftarrow \text{COUNTFLIPS}(L); \)
3:  \( \text{until } N(r) > 0 \)
4:  \( N(r) \leftarrow \text{FINDMIN}(L, N(r)); \)
5:  \( \text{return } N(r); \)

:End
Chapter 11

Conclusions

In this thesis I have presented a large body of work completed during my degree on the topics of search and rendezvous by mobile agents in continuous environments. It is clearly demonstrated that although the Basic Linear Search problem has been around for over half a century, it, and its derivatives, still manage to attract the attention of a great many researchers. The reasons for this are simple. This is a highly enjoyable research topic that continues to yield interesting and surprising results (e.g. in Chapter 8 it was demonstrated that two completely unaware agents can manage to meet each other in nearly all situations where there is some amount of difference between them). Moreover, the increasing prevalence of multi-robot systems in our every day lives is only likely to increase the interest in these types of problems as they become more practically interesting. Indeed, one does not have to try very hard to imagine a future where search and rescue operations are taken over by teams of robots; or even an age of space exploration wherein robots are tasked with seeking out mineral rich asteroids or even extraterrestrial life. In all of these scenarios it is clear that search and rendezvous will necessarily play a central role.
Bibliography


[162] Animation of Algorithm 2 for \( n = 4 \). [https://drive.google.com/open?id=1OhmWegFZLFLiwQalvPoZSTg9Ah860mMn](https://drive.google.com/open?id=1OhmWegFZLFLiwQalvPoZSTg9Ah860mMn) Feb. 13, 2018.