Mobile Agent Rendezvous in the Ring

by

Cynthia Lynn Sawchuk

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario, Canada

January 2004
© Copyright 2004
Cynthia Lynn Sawchuk

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-89893-8

Our file Notre référence

ISBN: 0-612-89893-8

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

The mobile agent rendezvous problem is a search optimization problem based on the following question. How should $k \geq 2$ mobile agents move along the n nodes of a network in order to minimize the time required to meet or rendezvous?

In the asymmetric case of the problem, the mobile agents are uniquely identified and they can be assigned distinct algorithms in order to achieve a rendezvous. If the mobile agents cannot be distinguished, however, then they must run the same algorithm in an attempt to rendezvous. In the pure symmetric case of the rendezvous search problem, the mobile agents run the same deterministic algorithm while in the mixed symmetric case, they run the same randomized algorithm. When the rendezvous search problem is sufficiently symmetric, e.g., the mobile agents are identical, they have no common orientation, and the network is anonymous, rendezvous is impossible unless the symmetry is broken. Traditionally, the symmetry has been broken with randomization and thus almost all instances of the rendezvous search problem with significant symmetry are solved as mixed symmetric cases. We want to explore how rendezvous can occur in the pure symmetric cases of the rendezvous search problem. Since the mobile agents are executing the same deterministic algorithm, the symmetry in the problem must be broken by means other than randomization. We investigate how mobile agents can use identical tokens to break symmetry and thus rendezvous in the pure symmetric case. We are especially interested in the time and memory tradeoffs that such solutions make possible.

Acknowledgements

I thank Evangelos Kranakis and Danny Krizanc, my thesis advisors, for their inspiration, ideas, and enthusiasm. I am especially grateful for the extra time and effort that Evangelos Kranakis offered. I also thank Paola Flocchini, Nicola Santoro, Flaminia Luccio, and Lefteris Kirousis for sharing their invaluable insight into the mobile agent rendezvous problem.

I thank my husband, Kevin Fettig, for his unwavering support and encouragement throughout my studies at Carleton University.

Finally, I extend my thanks to the faculty and staff of the School of Computer Science at Carleton University. Over the years, I have come to know them as individuals whom I respect and appreciate.

Contents

1	Introduction	1
1.1	The Rendezvous Search Problem and Traditional Search Theory . . .	2
1.2	Review of the Rendezvous Search Problem	
	Literature	5
1.2.1	Mobile Agent Rendezvous with More than Two Searchers . . .	10
1.2.2	Topology of the Search Domain	12
1.2.3	The Markstart Rendezvous Search Problem	21
2	Our Contribution	26
3	Mobile Agent Rendezvous with Two Mobile Agents	29
3.1	The Network Model	30
3.2	Overview of Chapter 3	31
3.3	The Feasibility of Rendezvous	34
3.4	The Time and Memory Requirements of	
	Rendezvous	39

3.5	Time and Memory Tradeoff	46
3.6	Lower Bound for Memory	49
3.7	Movable Tokens	52
4	Mobile Agent Rendezvous with More than Two Mobile Agents	56
4.1	The Network Model	56
4.2	Overview of Chapter 4	57
4.3	The Feasibility of Rendezvous	58
4.4	Unconditional Solutions	64
4.5	Lower Bound on Memory	75
4.6	Conditional Solutions	76
4.7	Leader Election and Rendezvous	84
5	Mobile Agent Rendezvous when Tokens Fail	87
5.1	The Network Model	87
5.2	Overview of Chapter 5	87
5.3	The Feasibility of Rendezvous	88
5.4	Rendezvous When Tokens Fail Upon Release	88
5.5	Rendezvous When Tokens Fail After Release And the Ring Size is Unknown	92
5.6	Rendezvous When Tokens Fail After Release and the Ring Size is Known	99
5.7	The Cost of Token Failure	102

6	Mobile Agent Rendezvous with Randomization	104
6.1	The Network Model	104
6.2	Overview of Chapter 6	105
6.3	Rendezvous with Randomization	106
6.4	Rendezvous with Randomization and Tokens	111
6.5	The Performance of Deterministic and Randomized Algorithms . . .	112
7	Conclusion	114

List of Tables

3.1	Time Complexity for Rendezvous Search Problem with Two Mobile Agents at Distance $d < \frac{n}{2}$	33
4.1	The Rendezvous Search Problem with $k \geq 2$ Mobile Agents	57
5.1	The Cost of Token Failure	103
7.1	A Comparison of Rendezvous Times	115

Chapter 1

Introduction

A significant amount of research has focussed on the general rendezvous search problem, with operations research specialists producing the bulk of this research. Examples of the general rendezvous search problem include many problems that arise in the real world, e.g., searching for or regrouping animals, people, equipment, and vehicles.

Computer scientists, however, are interested in the rendezvous search problem in the context of mobile agents and networks. After being dispersed in a network, mobile agents may need to gather or rendezvous in order to share information, receive new instructions, or download new programs that extend the mobile agents' capabilities.

The mobile agent rendezvous problem, as explored in this thesis, differs significantly from the general rendezvous problem. First, network topologies are discrete and include specialized structures like rings, tori, and hypercubes, while the domain in a general rendezvous search problem is usually continuous. Second, memory and time requirements are important in any computer science problem but in the general

rendezvous problem research, the only calculated resource cost is time.

Third, in symmetric domains in the general rendezvous problem, e.g., a grid where locations cannot be uniquely identified, the symmetry is typically broken with randomization. In this thesis, however, we explore the use of tokens to break symmetry, where a token is represented by a single bit of memory. Network domains can easily be symmetric, e.g., an anonymous ring. When tokens can be used to break the symmetry in a network, the mobile agents can run a deterministic algorithm as opposed to a randomized algorithm. Fourth, even if a randomized algorithm is used to solve the mobile agent rendezvous problem, it is worthwhile investigating whether such solutions can be improved by using tokens.

Before discussing new research results, however, we first explain the relationship between the rendezvous search problem and traditional search theory, and then review the literature about the general rendezvous search problem.

1.1 The Rendezvous Search Problem and Traditional Search Theory

Alpern and Gal [12] provide the following description of traditional search theory.¹ A searcher wants to minimize the time T it takes to capture a given target. The goal of the target determines the nature of the search.

In the one-sided search, the target (mobile or stationary) does not react to the

¹The next five paragraphs rely heavily on their description.

searcher's actions. Extensive research has been conducted on this category of search theory. A 1991 survey on search theory literature [27] appearing in the Naval Research Logistics journal lists over 130 references on one-sided search.

Search games occur when the target of a search is a mobile evader or a stationary hider and the target's goal is to maximize the time to capture, T . The search can be formally described as a zero-sum game with payoff function T . Examples include pursuit-evasion games that model a predator chasing a prey. Differential game theory provides a framework for analyzing the structure and complexity of these games.

Both the searcher and the target are minimizers of the capture time T in the **rendezvous search problem** and both participants are referred to as searchers. In the asymmetric case of the problem, the searchers can be distinguished and they use different deterministic rendezvous algorithms. In the mixed symmetric case, the symmetry of the search problem prevents the searchers from using distinct algorithms so searchers use the same randomized algorithm. While randomization is typically used to break symmetry in the rendezvous search problem, the *markstart* approach of Baston and Gal [26] and the *token* approach presented in this thesis allow the searchers to run the same deterministic algorithm and are examples of the pure symmetric case.

In the rendezvous-evasion game of Alpern and Gal [12], the target's motives are uncertain. The probability p that the target will cooperate with the searcher is known to both the target and the searcher. Thus, for a given search context G , there is a continuous family $\Gamma_G(p)$ of search games where $\Gamma_G(0)$ is the search with a mobile or stationary hider and $\Gamma_G(1)$ is the asymmetric case of the rendezvous search problem.

The Alpern and Gal [12] definition of the rendezvous-evasion game differs from that of Lim [47] in that the latter involves three, rather than two, searchers. Two searchers are randomly placed among n indistinguishable locations and they want to rendezvous in the least expected time before either is found by an *enemy* searcher.

Buhrman [28] et al introduced the concept of mutual search where k agents, distributed over a complete network of n distinctly labeled nodes, must find each others' location. These agents, however, are not mobile but rather try to make contact by sending messages.

In the robot rendezvous problem discussed by Roy and Dudek [53], robots try to rendezvous in an unknown environment by choosing an appropriate rendezvous point, i.e., they try to find a focal point or landmark.

Ando [20] et al proved that rendezvous or *gathering* can occur among anonymous, oblivious robots with limited visibility and no common orientation if those robots can instantaneously sense, compute, and move.

Flocchini et al [38] proved that if the same robots can only complete calculations and movements in an uncertain but finite period of time, then gathering cannot occur. If, however, the robots are given a common orientation, then gathering is possible.

Prencipe and Cieliebak [51] also consider a group of robots that are decentralized, asynchronous, anonymous, oblivious, and have no common knowledge or direct communication. As is, the robots cannot gather but if they can detect multiplicity, i.e., more than one robot in a given position, then they can gather if there are three or four robots. If there are more than four robots, they prove that gathering can

occur if the robots' initial configuration is not bi-angular. They give an n -gon with equal angles $\alpha = \beta$ and equal distance to the center c as a special case of a biangular configuration.

1.2 Review of the Rendezvous Search Problem

Literature

In one of the earliest discussions of the rendezvous search problem, Schelling [54] considers the case of two parachutists who land in unknown territory and must meet as soon as possible. The proposed solution has the parachutists meeting at a *focal point*, i.e., a distinguished point in the landscape.

Despite Schelling's early interest in the problem, researchers largely ignored the rendezvous search problem until the early 1990s. At that time, Anderson and Weber [19] considered the mixed symmetric case of the rendezvous search problem with a search domain of n identical discrete locations and two identical searchers. They proved that with $n = 2$, the optimal strategy has searchers randomly visiting locations at each step and results in a minimum expected rendezvous time of 2. With $n = 3$, the optimal strategy has searchers, over blocks of $n - 1$ steps, either remaining in one location for $n - 1$ steps or visiting $n - 1$ different locations. The resulting minimum expected rendezvous time is $\frac{8}{3}$.

In addition to proving results for the rendezvous search problem on n identical

discrete locations, Anderson and Weber [19] suggested many interesting extensions and open problems. First, they suggested that the n discrete locations be placed in a structure that reflects the topology of a network and that the locations not be commonly labeled. They noted that if three discrete locations comprised a ring, then a searcher should follow the π strategy, i.e., choose the first location at random and then choose subsequent moves in pairs. That is, if a rendezvous does not occur after step j , $j = 1, 3, 5, \dots$, then for the next two steps a searcher either returns to the location visited in step 1 or it visits the two other locations in random order. The expected rendezvous time is then $\frac{10}{3}$.

Second, Anderson and Weber [19] suggested changes in the search model. They considered the case where α , the probability that two searchers find each other when searching the same location, is fixed but less than 1 and showed that, for $n = 2$, the random strategy is still optimal. They also considered the case where each searcher could search $m > 1$ of the n locations at each step. For $m = 2$ and $n = 4$, they proved that the optimal strategy is to search each possible pair of locations with equal probability. Another suggested change to the search model allowed rendezvous to occur if searchers were on the same or adjacent locations. Anderson and Weber [19] show that for some network topologies, this is equivalent to the problem where m of the n locations can be searched at the same time. For example, the $m = 2$, $n = 4$ problem is equivalent to the rendezvous search problem on the vertices of an octahedron such that searchers meet either at the same vertex or at adjacent vertices.

In a third extension to the rendezvous search problem, Anderson and Weber [19]

suggested that changes could be made in the synchronization of the searchers' moves. For example, searchers might start searching at different times or they could move between locations at any time.

Anderson and Weber [19] list a fourth and last extension to the rendezvous search problem. They discuss a suggestion made by Shmuel Gal that the searchers could leave messages at locations they have searched. Anderson and Weber [19] prove that in the case of n discrete locations, the expected rendezvous time will be at least $O(\sqrt{n})$ and they provide the following algorithm that results in an expected rendezvous time of $O(\sqrt{n})$:

- For exactly $\sqrt{n}$ steps, visit $\sqrt{n}$ locations and leave messages.
- The visited locations are referred to as the *original set*.
- For the moment, ignore other messages, if found.
- For the next $\sqrt{n}$ steps, revisit each location in the *original set* and note the locations at which the other searcher left a message.
- If there is a non-empty intersection between the *original sets* of the two searchers, revisit the locations in this intersection at random.
- Otherwise, visit at random those locations not yet visited. Upon first discovery of a location where the other searcher left a message, leave a message.
- Revisit locations in the *original set* at random until a message left by the other searcher is discovered.

- Then uniformly distribute visits between the two locations which both searchers have visited.

Anderson and Weber [19] indicate that as early as 1990, Shmuel Gal considered the issue of searchers leaving messages at nodes they visited. However, no one analyzed the impact of leaving messages on the rendezvous search problem until much later and even then research on the topic is very limited. The most rigorous research was undertaken in a paper by Baston and Gal [26]. They introduced the *markstart* rendezvous problem, where searchers leave marks at their starting points. This variation of the rendezvous search problem will be discussed in detail later.

In addition to suggesting extensions to the rendezvous search problem for n discrete locations, Anderson and Weber [19] also made several conjectures:

- The minimal expected number of steps required to rendezvous is an increasing function of n .
- The π strategy for $n = 3$, described earlier, maximizes the probability of rendezvous in k steps for all k . It was shown to be true for $k \leq 4$.
- For $n = 4$, the earlier described strategy where searchers, over blocks of $n - 1$ steps, either remain in one location for $n - 1$ steps or visit $n - 1$ different locations is not optimal.
- The strategy just mentioned is, however, asymptotically optimal as $n \rightarrow \infty$.

- If searchers at the same location only find each other with probability $0 < \alpha < 1$, then the optimal strategy for $n = 2$ is unchanged, i.e., a searcher should visit a randomly chosen location at each step. As well, the optimal strategy is the same for all α , even for larger n .

As mentioned earlier, Anderson and Weber [19] used a randomized algorithm to break symmetry and ensure a rendezvous. A few years later, Alpern and Reyniers [16] also recognized the importance of symmetry in the rendezvous search problem. They noted that the absence of a Schelling [54] *focal point* resulted in a more symmetric problem which was subsequently more difficult to solve. For example, when the nodes in a ring are uniquely numbered, the rendezvous search is trivial since identical mobile agents can rendezvous at the node with the lowest number. The problem is more difficult if the nodes are anonymous but the ring is oriented and is even more difficult if the ring is anonymous and unoriented. Alpern and Reyniers [16] used randomization to break symmetry in the rendezvous search problem. Almost without exception, researchers have broken symmetry in the rendezvous search problem by using randomized algorithms or, if the symmetry is not absolute, by exploiting some asymmetric aspect of the problem domain so that the searchers can run different deterministic algorithms. In fact, Yu and Yung [56] provide a counterexample to prove that the pure symmetric case of the rendezvous search problem cannot be solved on a general graph. They consider an anonymous four node ring. They show that if the searchers are identical and initially placed two nodes apart on the ring, they cannot

rendezvous if they run the same deterministic algorithm. Yu and Yung [56] also suggest that symmetry be broken with a randomized algorithm or, where possible, with different deterministic algorithms. They present a random walk based algorithm for three symmetric cases, i.e., the anonymous n node complete graph, the anonymous n node hypercube, and the anonymous n node ring. They assume that there are k searchers and that $k \ll n$. The expected rendezvous times are $O((\frac{n}{k}) \log n)$ for the complete graph, $O((\frac{n}{k}) \log^2 n)$ for the hypercube, and $O((\frac{n}{k}) \log n + n)$ for the ring.

Dessmark et al [32] studied the case of deterministic rendezvous in graphs but broke the potential symmetry faced in such cases by assuming that the mobile agents have separate identifiers and can distinguish between the local ports of a node. Under these assumptions, Dessmark et al [32] determined that rendezvous could occur in any n -node tree in time $O(n + \log m)$, where m is the smaller of the two identifiers. They also determined that rendezvous could occur on the ring in time $\Theta(d \log m)$, where d is the initial distance between the mobile agents, if the mobile agents had the same start times. If the mobile agents started at different times, then the lower bound on the time necessary for rendezvous was $\Omega(n + d \log m)$.

1.2.1 Mobile Agent Rendezvous with More than Two Searchers

While Yu and Yung [56] look at the case of $k \geq 2$ searchers, most of the literature deals with the two searcher case. The few exceptions include Lim, Beck, and Alpern [49], Alpern [5], Pikounis and Thomas [50], and Gal [39].

Lim, Beck, and Alpern [49] considered $k > 2$ searchers that are initially randomly placed on the line at k consecutive integers and are randomly faced in either direction. For the asymmetric case with $k = 3$, they proved that the minimum expected rendezvous time for *two* of the three searchers is $\frac{47}{48}$. Lim, Beck, and Alpern [49] also proved that, for the symmetric case with $k \geq 2$ searchers, the minimum expected rendezvous time T for all searchers is $\frac{n}{2} \leq T \leq \frac{n}{2} + 5$ and is asymptotically equal to $\frac{n}{2}$. Gal [39] later proved that in the asymmetric case for $k > 2$ searchers on the line, the asymptotic worst case rendezvous time is $\frac{n}{2} + O(\log n)$.

Lim and Alpern [48] introduced the idea of the minimax rendezvous time for the rendezvous search problem, i.e., the minimum time required for the searchers to rendezvous. They proved that, in the asymmetric case for $k = 3$ searchers placed at adjacent integers on the line, the minimax rendezvous time is at least $\frac{7}{2}$ and Baston [24] subsequently proved that the minimax rendezvous time is exactly $\frac{7}{2}$. Gal [39] later proved that the asymptotic minimax rendezvous time for k players randomly placed at adjacent integers on the line is at most $\frac{k}{2} + 2 \log n + \frac{7}{2}$.

Lim and Alpern [48] also introduced the idea of a *sticky* rendezvous where searchers who meet during a rendezvous search stay together until all the searchers rendezvous. They proved that the *sticky* minimax rendezvous time for $k = 3$ adjacent searchers on the line is 5. Partial rendezvous among $k > 2$ searchers was also studied by Pikounis and Thomas [50]. They used simulations to compare the partial rendezvous strategies of *stick together* and *split and meet* for the rendezvous search problem on the complete graph. Pikounis and Thomas [50] restricted the searchers to Markov and stationary

strategies where, respectively, a searcher cannot recall anything other than its current position and the probability of a searcher moving at any step does not depend on the time period. Under such restrictions, the *split and meet* strategy was more effective when the number of searchers was small. As the number of searchers grew, however, congestion made meetings less productive and the *stick together* strategy became more effective.

Alpern [5] studied the rendezvous search problem for $k > 2$ searchers at unit distance apart on a circle. The searchers had no sense of direction and there was no common labeling of the circle. Alpern [5] proved that the minimum expected rendezvous time was at most $\frac{n+1}{2} + \frac{1}{2^{n-1}-1}$ in the symmetric case and at most $\frac{n+1}{2} + \frac{n}{2^n}$ in the asymmetric case.

1.2.2 Topology of the Search Domain

Instances of the rendezvous search problem are often distinguished by the topology of the search domain. The real line has attracted the most attention with contributions by, among others, Anderson and Essegaiier [17], Alpern [1], Baston and Gal [25], Lim and Alpern [48], and Baston [24].

Gal [39] also studied the asymmetric case of the rendezvous search problem on the line. He assumed that the initial distance between the searchers, d , was drawn from a known distribution function F with a finite mean and that the searchers had no sense of direction. Gal [39] proved that the strategy *wait for mummy*, where one searcher

remained stationary and the other searched at random, was never optimal in the asymmetric case of the rendezvous search problem on the line. Baston and Gal [25] proved that if the initial distance d between the searchers was drawn from a known distribution F with support $[0, \infty)$ and mean μ , then the asymmetric rendezvous time is at most 5.73μ .

Alpern and Beck [10] showed that the asymmetric case of the rendezvous search problem on the line is equivalent to the double linear search. In the latter search, the search domain is two distinct lines and each searcher is placed at the origin of one of the lines, i.e., one searcher per line. The searchers can alternate moving at speed 2 or can move with a combined speed bounded by 2. Using this approach, Alpern and Beck [10] proved that if d , the initial distance between the searchers, is drawn from a distribution with finite mean μ , then the upper bound on the asymmetric rendezvous search on the line is 5.514μ .

When considering the mixed symmetric case for the line, Alpern [1] restricted his attention to bounded distributions for d , the initial distance between the searchers, i.e., distributions F that have a finite maximum value D . He defined a *step* as the distance $\frac{D}{2}$ and used the strategy where each searcher picks a random direction, goes one step in that direction, then goes two steps in the next direction, and so on. Alpern [1] then proved that if the initial distance d was chosen from a distribution F with maximum D and mean μ , the mixed symmetric rendezvous time for the line would be at most $2D + \frac{\mu}{2}$. This mixed symmetric rendezvous time was bettered first by Anderson and Essegaiier [17] and then by Baston [24], who proved the upper bound

was $1.7091D + \frac{\mu}{2}$.

In the research discussed so far, the optimization goal is to minimize the time to rendezvous. When Alpern and Beck [8] considered the rendezvous search problem on the line with limited resources, however, they suggested that the appropriate optimization problem was to maximize the probability of rendezvous. They assumed that the total distance travelled by a searcher is limited and derived the optimal conditions for rendezvous under particular distributions of d , the initial distance between the searchers.

Howard [41] looked at the rendezvous search problem on the continuous closed interval $[0, 1]$. He proved that when the searchers' initial positions on the closed interval are determined by a uniform distribution, the asymmetric and mixed symmetric cases on $[0, 1]$ have the same rendezvous time of $\frac{1}{3}$ and this rendezvous time holds with or without a common labeling of the interval.

Howard [41] also studied the rendezvous search problem on the discrete interval. He assumed that the distribution function for the searchers' initial positions was non-decreasing as one moved from left to right in the interval. The uniform distribution is a special case of such a distribution. Howard [41] constructed an optimal strategy that is the same for both the asymmetric and mixed symmetric cases, and uses an artificial searcher called the *right sweeper*. The *right sweeper* starts at the right end of the discrete interval at time 0 and moves one position to the left in each time period until it reaches the left end of the interval. In the optimal strategy, each searcher moves at unit speed to the right until it meets the *right sweeper* and then

the searcher returns to the center with *right sweeper*.

In a subsequent paper, Chester and Tutuncu [29] noted that the distribution assumed by Howard [41] in his study of the rendezvous search problem on the discrete interval is non-decreasing as one moves from left to right in the interval. This class of distributions includes the uniform distribution but excludes the normal distribution, which is centrally symmetric but decreasing as one moves from the center outward in the interval. When Chester and Tutuncu [29] looked at the rendezvous search problem for the discrete and continuous interval, they focused on initial placement distributions that are centrally symmetric and non-increasing as one moves away from the center. Unlike Howard [41], they assumed that the searchers shared a common labeling for an interval so that each searcher could find the center of a given search domain. For both the discrete or continuous interval, they proved that optimal strategy has each searcher go directly to the center of the region and wait.

To study the rendezvous search problem on the infinite line, Chester and Tutuncu [29] considered the search domain to be the limit of a sequence of expanding intervals. Assuming that the searchers again shared a common labeling of the search domain, Chester and Tutuncu [29] proved that the optimal strategy on the infinite line is the same as that for the discrete or continuous interval, i.e., go to the center and wait. Chester and Tutuncu [29] subsequently studied the rendezvous search problem on the finite grid. The distribution of the searchers' initial placements was again assumed to be centrally symmetric and non-decreasing as one moves out from the center of the grid and the searchers again shared a common labeling of the grid

so that they could find the center of the grid. Chester and Tutuncu [29] decomposed the rendezvous search problem on the grid into two simultaneous and independent linear rendezvous search problems and proved that the symmetry and monotonicity assumptions for the initial placement distribution on the grid carried over to the linear problems. They proved that the two-dimensional strategy derived from the optimal solutions of the linear subproblems is optimal for the grid problem. The optimal strategy for the grid has each searcher going to the closest main diagonal of the grid, moving directly along that diagonal to the center of the grid, and waiting there for the other searcher.

Chester and Tutuncu [29] offer the conjecture that the optimal strategy for the rendezvous search problem on the continuous grid is to move directly to the center and wait for the other searcher. They note, however, that the optimal strategy in this case is not the limit of the optimal strategies for the rendezvous search problem on finer and finer discretizations of the two dimensional box and thus different proof techniques are necessary.

Howard [41] considered the asymmetric case of the rendezvous search problem on a circle with unit circumference. With a common labeling, searchers can move in opposite directions along the short side of the circle and the rendezvous time is for the asymmetric case is $\frac{1}{4}$. If there is no common labeling, however, then the rendezvous time for the asymmetric case is $\frac{1}{2}$ and the *wait for mummy* strategy is optimal.

Alpern and Howard [14] also studied the asymmetric case on a circle with unit circumference, but the initial distance between the searchers was chosen from the

monotone density h , $0 \leq h \leq \frac{1}{2}$. They proved that if h is increasing, then in the optimal strategy one searcher will go around the circle at maximum speed one and the other searcher will move at maximum speed one in a random direction for time $\frac{c}{2}$, then reverse direction and move at maximum speed one for time $\frac{c}{2}$, and then stay still until time 1. If h is decreasing and $h(\frac{1}{2}) > \frac{h(0)}{2}$, however, one searcher should still go around the circle at maximum speed one, but the other searcher should stay still for time c , then move at maximum speed one in a random direction for time $\frac{1-c}{2}$, and finally reverse direction and move at maximum speed one until time t . Alpern and Howard [14] also proved that if h is *strictly* increasing or decreasing, then the relevant strategy described above is *uniquely* optimal. They also proved that if h is increasing and decreasing, then h is the uniform density with $h'(x) = 2$ and the two strategies discussed above are optimal with $c = \frac{1}{2}$.

Alpern [5] also studied the asymmetric case of the rendezvous search problem on the circle. He assumed that the initial distance between the searchers was chosen from a slowly varying density and proved that any optimal strategy has one searcher going completely around the circle at maximum speed one, and the other searcher back at its starting point by the time 1. Rendezvous occurs in at most time 1. Alpern [5] also proved that the *wait for mummy* strategy is optimal for the asymmetric case on the circle only if the initial distance between the searchers is uniformly distributed.

In the mixed symmetric case of the rendezvous search problem on the circle, Alpern, Baston, and Essegaier [6] studied the strategy where at each step, a searcher moves clockwise for c steps with probability $p \leq \frac{1}{2}$, counterclockwise for c steps with

probability p , and remains still with probability $1 - 2p$. They find that the expected rendezvous time is minimized when $c = n - 1$ for an n node circle and prove that the expected rendezvous time is at most $(n - 1)(2n - 1)/(2n)$ if n is odd and at most $1.254 + O(1)$ if n is even.

Anderson and Fekete [18] studied the asymmetric case of the rendezvous search problem in the plane. They assumed that the searchers have a common two-dimensional orientation, that the maximal speed of the first searcher, v_1 , is no slower than the maximal speed of the second searcher, v_2 , and that the searchers know that the initial distance between them is d . Rendezvous will therefore take at least time $t_d = \frac{d}{v_1 + v_2}$. Anderson and Fekete [18] proved that the *wait for mummy* strategy, where the second searcher remains stationary, results in a worst case rendezvous time of $\frac{d}{v_1}(1 + 2\pi)$ and an expected time of $\frac{d}{v_1}(1 + \pi)$. However, Anderson and Fekete [18] also proved that the *kissing-circles* strategy yields the optimal worst case time $t_d(1 + 2\pi)$ and the optimal expected time $t_d(1 + \pi)$. In this strategy, the first searcher moves a small distance north from its initial position and then searches in a clockwise circle around its initial position. The second searcher moves a small distance south from its initial position and searches in a clockwise circle around its original position. The searchers move through ever increasing circles about their respective starting points and are guaranteed to rendezvous.

If the initial distance between two searchers in the plane is unknown, however, Anderson and Fekete [18] prove that the asymmetric case of the rendezvous search problem can be solved only if each searcher can see all other searchers that are within

a non-zero distance r .

Anderson and Fekete [18] also considered the rendezvous search problem in the plane under bounded resources. Assuming that each searcher could travel at most a given distance, they presented a strategy that was probability-optimal, i.e., maximized the probability of rendezvous, and then proved that among all probability-optimal strategies, this strategy also minimized the time to rendezvous.

In addition to the plane, Anderson and Fekete [18] also explored the asymmetric case of the rendezvous search problem on the infinite grid. They assumed that the searchers were initially 2 nodes apart, faced in random directions (referred to as *North*), have no common orientation, and have a maximum speed of 1. Anderson and Fekete [18] proved that the optimal strategy is $(NWSSEENN, NSNSNSNS)$, a modification of the *wait for mummy* strategy. The first searcher checks the four nodes at distance 2 from its initial position while the other searcher moves between its initial position and an adjacent node.

As mentioned earlier, Anderson and Weber [19] proved that for the asymmetric case on the complete graph for $n = 3$, the π strategy resulted in a minimal expected rendezvous time of $\frac{8}{2}$.

Also mentioned earlier, Alpern, Baston, and Essengaier [6] proved that a π strategy results in an expected rendezvous time of $\frac{5}{2}$ for the asymmetric rendezvous problem on the complete graph for $n = 3$. Yu and Yung [56] presented a random walk based algorithm for the complete graph where k searchers will rendezvous in time $O(\binom{n}{k} \log n)$ with high probability if $k \ll n$, e.g., $k < \sqrt{n}$.

Yu and Yung [56] also proved that rendezvous cannot be achieved on a general graph if all searchers use the same deterministic algorithm, i.e., the pure symmetric case. However, they present a randomized algorithm that solves the mixed symmetric case of the rendezvous search problem on a general graph in time $O(n \log \frac{|U|}{k-1})$. Yu and Yung [56] also prove that if the searchers do not start the randomized algorithm at the same time, then the rendezvous time is $O(\delta + n \log \frac{|U|}{k-1})$, where δ denotes the maximum discrepancy in the starting times.

Alpern, Baston, and Essengaier [6] also studied the general graph. They considered the graph $X = (V, E)$ along with a given group of its isometries, G . As Alpern, Baston, and Essengaier [6] note, the group G denotes that part of the symmetry of X that must be addressed in the searchers' strategies. Alpern, Baston, and Essengaier [6] proved that if G is transitive and $|V| = n$, then the rendezvous time in the asymmetric case is at least $\frac{n-1}{2}$. Furthermore, if X is Hamiltonian, then for any isometry group G , the rendezvous time in the asymmetric case is at most $\frac{n-1}{2}$. Thus if X is a Hamiltonian graph and G is a transitive group of isometries of X , the rendezvous time in the asymmetric case is exactly $\frac{n-1}{2}$. The same rendezvous time holds if there is a Hamiltonian path beginning at any node in X .

Alpern, Baston, and Essengaier [6] note that if G is transitive, then X is vertex-transitive. They recall the conjecture by Lovas that all vertex transitive graphs have Hamiltonian paths starting at any node. If the conjecture holds, then the rendezvous time on the graph X with a transitive group of isometries G is $\frac{n-1}{2}$.

1.2.3 The Markstart Rendezvous Search Problem

In the classical rendezvous search problem, i.e., with no messages or tokens used, the searchers receive information only at the beginning of the search, e.g., they may be told that there are n nodes in the network or they might be given a common labeling of the search domain. The *markstart* approach of Baston and Gal [26], however, has the searchers leave a mark at their respective starting nodes and that mark provides information to the searcher that happens upon it. While the acquisition of information during the search is characteristic of coordination games, the players in typical coordination games are able to communicate from the beginning of the game, i.e., rendezvous is assumed. As a result, little of the coordination game theory is currently applicable to the rendezvous search problem.

The *markstart* approach of Baston and Gal [26] also differs from the classical approach in that they consider not only the asymmetric and mixed symmetric cases, where the searchers respectively use different deterministic strategies or the same randomized strategy, but also considers the pure symmetric case where both searchers follow the same deterministic protocol. In the classical rendezvous search game, the problems of symmetry that can arise in the pure symmetric case are typically avoided by using randomized strategy. Baston and Gal [26] attempt to solve the problem of symmetry by having the searchers leave marks at their respective starting points.

Baston and Gal [26] do note, however, that the *markstart* approach can sometimes make the rendezvous problem more complicated. In the classical rendezvous search

problem on the complete graph K_n , the *wait for mummy* strategy is optimal in the asymmetric case, i.e., where the searchers use distinct deterministic protocols, and produces a upper bound on the run time that is asymptotically close to $0.8289n$. With the *markstart* approach, however, the optimal strategy results in a run time that is asymptotically close to $\frac{n}{3}$, indicating that when n is large, *both* players need to move.

Baston and Gal [26] discuss the *markstart* protocol for the rendezvous search problem on the line, the circle, and the complete graph, K_n . The latter two structures are of interest as they represent common network topologies. Anderson and Weber [19] studied the classical rendezvous search problem on the complete graph. They proved that *wait for mummy* is an optimal strategy pair and the resulting asymmetric rendezvous value is $\frac{(n-1)}{2}$. Only an asymptotic upper bound of $0.8289n$ has been established for the corresponding mixed symmetric problem. Baston and Gal [26] admit that the corresponding *markstart* problem is difficult and prove upper bounds for only some of the *markstart* problems.

If $n = 2$ or $n \geq 4$, Baston and Gal [26] state that the *markstart* rendezvous value for the pure symmetric case on the complete graph is infinite. When $n = 3$, the rendezvous value for the pure symmetric case of the *markstart* rendezvous is $\frac{7}{6}$. That is, a pure strategy instructs the players to move at time 1, after which they can deduce each others starting nodes and thus can rendezvous at time 2 at the node that was not a starting point.

Baston and Gal [26] prove that, for the complete graph K_n , the *markstart* ren-

rendezvous values for the asymmetric case and the mixed symmetric case respectively are $\frac{1}{2}$ and 1 when $n = 2$. When $n = 3$, Baston and Gal [26] prove that *wait for mummy* is optimal in the asymmetric case and the resulting *markstart* rendezvous value is 1. For the more difficult symmetric case, Baston and Gal [26] offer the conjecture that the *markstart* rendezvous value is $\frac{7}{6}$.

In calculating the asymptotic upper bounds for the *markstart* rendezvous values for the complete graph, Baston and Gal [26] note that if each searcher searches the n nodes randomly, the expected time for a searcher to locate the other searcher's starting node is about $\frac{n}{3}$.

In the asymptotic asymmetric case for the complete graph, the first searcher searches the n nodes randomly but returns to its starting node at times modulo 100. Once the first searcher finds the other searcher's starting node, it continues to return to its starting node at times modulo 100, but otherwise remains at the other searcher's starting node. The other searcher follows an analogous strategy except that it returns to its starting node at times $100i + 1$.

In the asymptotic mixed symmetric case for the complete graph, *each* searcher searches the n nodes randomly but returns to its starting node at times modulo 100. Once a searcher finds the other searchers starting node, it moves from one starting node to the other with probability $\frac{1}{2}$. In both the asymptotic asymmetric and the asymptotic mixed symmetric case of the *markstart* rendezvous search problem for the complete graph, a searcher can expect to rendezvous soon after it locates the other searchers starting node. Thus, for the complete graph, the asymptotic *markstart*

rendezvous value is at most $\frac{n}{3} + o(n)$ for both the asymmetric and the mixed symmetric case.

Baston and Gal [26] note that since the *wait for mummy* strategy is optimal for the asymmetric case of *markstart* rendezvous search problem when $n = 3$, then there must be a least n for which it is not optimal. At present, there are no approximations of this least value of n .

As mentioned earlier, Baston and Gal [26] also analyze the *markstart* rendezvous search problem for the circle. Baston and Gal [26] assume that the searchers know the distance d that separates them, that $0 \leq d < \frac{1}{2}$, and they have a sense of direction. The *markstart* rendezvous value for the asymmetric case is $\min\{d, \frac{1}{4}\}$ and equals the rendezvous value for the classical rendezvous search problem. To achieve this rendezvous value, the searchers begin by moving in opposite directions at a unit speed. If the searchers do not meet at time $\frac{d}{2}$, then the searchers reverse direction if $d \leq \frac{1}{4}$ and continue in the same direction otherwise.

Baston and Gal [26] note that the mixed symmetric case of the *markstart* rendezvous search problem is difficult to analyze and restrict their attention to the case where $\frac{1}{3} \leq d < \frac{1}{2}$. To achieve this rendezvous value, each searcher chooses a direction at random and moves for time d . If rendezvous does not occur, the searchers then follow the pure symmetric strategy. The *markstart* rendezvous value for the mixed symmetric case with $\frac{1}{3} \leq d < \frac{1}{2}$ is thus at most $\frac{3}{8}$.

Since we are currently investigating how tokens can be used to break symmetry on a ring network, the pure symmetric case of the *markstart* rendezvous search problem

is of most interest. As mentioned earlier, Baston and Gal [26] assume that, in addition to a common orientation, the searchers know the distance d that separates them and that $0 \leq d < \frac{1}{2}$. The *markstart* rendezvous value for the pure symmetric case is $\frac{3}{2}$. To achieve this rendezvous value, the searchers begin by moving distance d in the clockwise direction. The searcher that is then at a marked starting node continues in the same direction, while the other searcher reverses direction. As will be shown later, our research also proves that the appropriate rendezvous value, under the same assumptions and with the use of tokens, is $\frac{3}{2}$. In our work, however, the rendezvous search problem for the ring is studied under various degrees of symmetry. For example, we also study the more symmetric case where the searchers do not have a common sense of direction. As the next chapter will show, our approach to the rendezvous search problem is unique in several ways.

Chapter 2

Our Contribution

Chapters 3 through 6 contain the research contributions of the thesis. In Chapter 3, we investigate the two mobile agent case of the rendezvous problem. First, we prove an impossibility result, i.e., when identical, stationary tokens cannot be used to ensure rendezvous. We then provide several algorithms in which the mobile agents successfully use stationary, identical tokens to solve the rendezvous problem. The algorithms operate under various assumptions about the mobile agents' memory capacity and their knowledge of the ring. The resulting memory and time requirements of these algorithms suggests that the memory requirements can be traded off against the time requirements. We prove, however, that there is a non-constant lower bound on the memory required to solve the mobile agent rendezvous problem.

Since the mobile agent rendezvous problem cannot be solved with constant memory when stationary, identical tokens are used, we explore the use of movable tokens. We present an algorithm where, whenever the mobile agents meet a token, they are

able to move that token to an adjacent node. Using this algorithm, we prove that the mobile agent rendezvous problem can be solved with constant memory when the identical tokens are movable.

In Chapter 4, we study the mobile agent rendezvous problem with $k \geq 2$ mobile agents. We prove an impossibility result wherein identical, stationary tokens cannot be used to ensure rendezvous in the $k \geq 2$ mobile agent case. We present several general algorithms that solve the rendezvous problem and determine their time and memory requirements. We then discuss a set of algorithms that solve the rendezvous problem under specific conditions, e.g., the primality of the number of nodes in the network, and calculate the respective time and memory requirements. After proving the lower bound on the memory required for the $k \geq 2$ mobile agent rendezvous, we conclude by exploring the relationship between mobile agent rendezvous and leader election.

In Chapter 5, we investigate the case where tokens can fail. We present three algorithms that, respectively, solve the rendezvous problem when 1) tokens can fail upon release by the mobile agents, 2) tokens can fail at anytime, and 3) tokens can fail at anytime but the mobile agents have additional information. We calculate the time and memory requirements for these solutions and compare them to the memory and time requirements when tokens do not fail.

In Chapter 6, we return to the $k = 2$ mobile agent case and investigate the use of randomization to solve the rendezvous problem. We discuss a random walk-based algorithm and determine its time and memory requirements. We then explore how

randomization and tokens can be combined to solve the rendezvous problem. We calculate the time and memory of the respective algorithms and compare them to the time and memory requirements of the appropriate deterministic algorithms from Chapter 3.

We make our concluding remarks in Chapter 7.

Chapter 3

Mobile Agent Rendezvous with Two Mobile Agents

In this section, we study the rendezvous search problem for two identical mobile agents in an anonymous, synchronous, and possibly oriented n node ring network.¹ As mentioned in the previous chapter, our approach differs from that of other researchers in that, rather than using randomized algorithms or different deterministic algorithms to break symmetry, we investigate how **identical tokens** can be used to the same end. Tokens are commonly used in algorithms for mutual exclusion, resource allocation, and synchronization but, with one exception [26], they have not been used in the rendezvous search problem. If identical tokens can be used to break symmetry in the problem, then rendezvous can occur even in the pure symmetric case.

The following example illustrates how identical tokens can be used to break sym-

¹Most of these results were originally presented in [44].

metry. Consider a five node anonymous, synchronous ring where the two identical mobile agents are located one node apart. If both mobile agents run the same deterministic algorithm, they will not rendezvous if they move in the same direction, at the same speed, and at the same time. Rendezvous can be guaranteed, however, if each mobile agent has an identical token that it leaves at its respective starting node. The algorithm can direct each mobile agent to walk until a token is reached, change direction, and continue walking in the new direction. The two mobile agents will rendezvous after at most three steps.

3.1 The Network Model

In this section, the model consists of two identical mobile agents that are $d \leq \frac{n}{2}$ nodes apart in an anonymous, synchronous, n node ring. The mobile agents may or may not share a common orientation, i.e., agree on the direction that is clockwise. A given node requires only enough memory to host a token and, at most, two mobile agents. Each mobile agent, MA , owns a single identical token. The MAs follow the same deterministic algorithm and begin execution at the same time. The MAs walk along the ring and are able, at each node, to place a token, remove a token, move clockwise or counter-clockwise to an adjacent node, or stay at the given node. A token or MA at a given node is visible to all MAs on the same node, but is not visible to any other MAs .

Different versions of the model can be created by changing the memory size and

knowledge of the *MA*s. Memory permitting, a *MA* can count the number of nodes visited, the number of nodes to the next token, or the total number of nodes in the network. In addition, a *MA* might already know the number of nodes in the network, the number of nodes to the nearest token, or some other network parameter and requires sufficient memory to store this information. Since the *MA*s are identical, they face the same limitations on their knowledge of the network.

The mobile agent rendezvous problem is solved, i.e., rendezvous occurs, when the mobile agents either meet on a network node or cross the same network link at the same time while moving in opposite directions. While we are aware of the problems that arise from the latter definition of rendezvous, we defer the study of these problems to the sequel. Note that when the number of nodes in the network is odd, rendezvous occurs on a node but when both the number of nodes and the initial distance between the *MA*s is even, rendezvous occurs on a network link. We assume that rendezvous occurs as soon as possible after the mobile agents begin travelling in opposite directions, i.e., one *MA* is moving clockwise while the other *MA* is moving counterclockwise around the ring.

3.2 Overview of Chapter 3

In an earlier example, we saw that identical tokens could be used to break symmetry when there are five nodes and the *MA*s are two nodes apart. Each mobile agent releases their token on their starting node, walks until it reaches a token, and then

reverses direction and continues walking. Rendezvous occurs after the *MA*s have walked three nodes.

Using identical tokens, however, does not guarantee that rendezvous is possible. Consider the example where $n = 4$ and $d = 2$, i.e., $d \leq n$ is the distance between the two *MA*s. Identical tokens cannot be used, in the manner described above, to break symmetry in this case and thus rendezvous is impossible. Our first result is the generalization of this example, i.e., when $d = \frac{n}{2}$ it is impossible to construct an algorithm that uses identical stationary tokens to break symmetry and thus solve the rendezvous search problem.

Having established when identical stationary tokens **cannot** be used to break symmetry, we then consider the rendezvous problem where $d < \frac{n}{2}$. Each *MA* places its identical stationary token on its respective starting node in the ring and begins to walk around the ring at the speed of one node per unit of time. The *MA*s know that the network is a ring but they do not necessarily know d , n , whether $d < \frac{n}{2}$, or the orientation of the ring.

In Table 3.1, we show the upper and lower bounds on the time complexity necessary for a rendezvous between two mobile agents that use identical stationary tokens and know that $d < \frac{n}{2}$. If only one value appears in the *Lower/Upper Bound* column, then the bound is tight. Otherwise the lower bound precedes the upper bound. While the calculation of the lower bounds assumes that the *MA*s have unbounded memory, the calculation of the upper bounds assumes the amount of memory listed in the last column of Table 3.1.

n	Known		Lower/Upper Bound	Memory Required
	d	$oriented$		
Y	Y	Y	$3d/2$	$O(\log d)$
Y	Y	N	$3d/2, 5d/2$	$O(\log d)$
Y	N	Y	$(n + d)/2$	$O(\log n)$
Y	N	N	$(n + d)/2$	$O(\log n)$
N	Y	Y	$3d/2$	$O(\log d)$
N	Y	N	$3d/2, 5d/2$	$O(\log d)$
N	N	Y	$n + (d/2)$	$O(\log n)$
N	N	N	$n + (d/2)$	$O(\log n)$

Table 3.1: Time Complexity for Rendezvous Search Problem with Two Mobile Agents at Distance $d < \frac{n}{2}$

As Table 3.1 indicates, if the MAs know d , then the lower bound on the time complexity of the rendezvous search problem is $\frac{3d}{2}$. If d is unknown but n is known, the lower bound on the time complexity increases to $\frac{n+d}{2}$. If both d and n are unknown, however, then the lower bound on the time complexity increases to $n + \frac{d}{2}$.

To calculate the upper bounds on the time complexity, we present four solutions to the mobile agent rendezvous problem when $d < \frac{n}{2}$ and identical stationary tokens are used to break symmetry. These solutions, with one exception, prove that the lower bounds in Table 3.1 are tight. The exception occurs when d is known but the orientation of the ring is unknown. The lower bound on the time complexity is $\frac{3d}{2}$ but the corresponding upper bound is $\frac{5d}{2}$.

While the results in Table 3.1 assume that the MAs know $d < \frac{n}{2}$, the solutions can be easily changed so that the MAs stop if $d = \frac{n}{2}$. It merely requires that each MA have memory $O(\log n)$. The last row in Table 3.1 represents the case where the MAs do not know n , d , or the orientation of the ring. The solution used to

calculate the upper bound requires $O(\log n)$ memory and $\frac{5n}{4}$ time. In Algorithm 1, we present another solution for the case where the MAs do not know n , d , or the orientation of the ring, but this solution requires $O(\log \log n)$ memory and $O(\frac{n \log n}{\log \log n})$ time. While Algorithm 1 suggests that the MAs ' memory complexity can be reduced at the expense of a solution's time complexity, we prove that if the MAs do not know n , d , or the orientation of the ring, then an algorithm that solves the rendezvous search problem when $d < \frac{n}{2}$ and stops otherwise requires $\Theta(\log \log n)$ memory.

This result, plus the upper bounds stated earlier, implies that solving the rendezvous search problem when the MAs have only $O(1)$ memory requires a change in the model. Suppose that the tokens are no longer stationary, i.e., anytime a MA encounters a token, it can move the token to an adjacent node. Algorithm 2, which requires only $O(1)$ memory, successfully uses identical but movable tokens to break symmetry and solve the rendezvous search problem. The resulting time complexity approaches $O(n)$ as d approaches 1, and approaches $O(n^4)$ as d approaches $\frac{n}{2}$.

3.3 The Feasibility of Rendezvous

Before we construct algorithms that use identical stationary tokens to break symmetry in the rendezvous search problem, we need to identify the conditions under which identical stationary tokens can be used to that end.

We prove that if $d = \frac{n}{2}$, then identical stationary tokens cannot be used to break symmetry and thus the rendezvous search problem cannot be solved by deterministic

means.

Let the atomic operations in the algorithms be restricted to the following:

RT - release the token

QT - query the host node to see if it contains a token

QMA - query the host node to see if it contains another *MA*

MCL - move one node clockwise

MCC - move one node counterclockwise

DM - do not move

IC(count) - increment the variable *count* by one (if sufficient memory is available)

In addition to the atomic operations listed above, an algorithm may contain two constructs:

do – until loop - the *MA* will repeat a finite sequence of operations until a given state occurs.

if – then statement - if a given condition holds, the *MA* will perform a finite sequence of operations.

The following two lemmata are necessary for the proof of Theorem 3.1.

Lemma 3.1 *Assume that the MAs are initially $d > 0$ nodes apart on the ring. If the MAs execute an identical algorithm from a common start time and that algorithm is restricted to a finite sequence of atomic operations, i.e., there are no do – until loops or if – then statements, then d , the distance between the mobile agents, will not change.*

Proof of Lemma 3.1. Consider an algorithm comprised of just one atomic operation. Any one of the atomic operations $\{RT, QT, QMA, IC(c), DM, MCL, MCW\}$ will leave d unchanged, as either both MAs are stationary or each MA moves one node in the same direction. Assume that this is also true for an algorithm that is a sequence of k atomic operations.

Consider an algorithm that is a sequence of $k + 1$ atomic operations. Assuming that d is unchanged after k atomic operations, it will also be unchanged if the $k + 1$ th operation is one of $\{RT, QT, QMA, IC(c), DM\}$ since the MAs are stationary under these atomic operations. The value of d is also unchanged if the $k + 1$ th operation is one of MCL, MCC since the MAs move one node in the same direction. Thus d is unchanged after $k + 1$ atomic operations. This completes the proof of Lemma 3.1. ■

Lemma 3.2 *Assume that the MAs are initially $d = \frac{n}{2}$ nodes apart on the network such that $CLToToken = CCtoToken = 0$ where, for a given MA, $CLToToken$ and $CCtoToken$ denote the distance to the nearest token in the clockwise and counter-clockwise direction respectively. If time t^* is the first time during the algorithm that d is not equal to $n/2$, then t^* is also the first time that the MA's respective values for*

$CLToToken$ (and thus $CCToToken$) differ.

Proof of Lemma 3.2

For all $t < t^*$, $d = n/2$ so that either both MAs are on separate tokens or neither MAs is on a token. If both MAs are on separate tokens, then $CLToToken = CCToToken = 0$ for both MAs . Suppose that at time t^* , neither MA is on a token. Suppose further that there exists a time $t' < t^*$ such that t' is the first time that the MA 's respective values for $CLToToken$ differ, as do their respective values for $CCToToken$. Let $\{CLToToken_1, CCToToken_1\}$ and $\{CLToToken_2, CCToToken_2\}$ be the MA 's respective values. Suppose, without loss of generality, that $CLToToken_1 < CCToToken_2$. If both MAs move $CLToToken_1$ nodes in the clockwise direction, then one MA will be on a token, but the other MA will not. Thus the distance between the MAs was not $d = n/2$. This contradiction proves Lemma 3.2. ■

Lemma 3.1 implies that if the two MAs begin an algorithm with distance $d = n/2$ and the algorithm contains only atomic operations, then the MAs will not rendezvous. A rendezvous cannot occur unless *do – until* loops or *if – then* statements are used. The inclusion of these constructs in an algorithm is a necessary but not sufficient condition for rendezvous.

Consider the conditional statements that are part of the *do – until* loop and the *if – then* statement. The boolean values of these conditional statements determine, respectively, how many iterations of the loop occur or if the body of the *if – then* statement is executed. In either case, a conditional statement to be evaluated at time

t is a function of the known history of the given MA at time $t - 1$. Two MA s cannot evaluate the same conditional statements at time t and receive different boolean values unless their known histories differ at time $t - 1$.

Theorem 3.1 *Consider two identical MA s in an anonymous n node ring. The MA s are initially $d = \frac{n}{2}$ nodes apart on the network. To begin an algorithm, each MA places an indistinguishable token on its respective starting node and these tokens remain in place for the rest of the algorithm. The two MA s will not rendezvous if they use the same deterministic algorithm.*

Proof of Theorem 3.1

A MA 's state includes its values of $CLToToken$ and $CCToToken$. Let s_0^i represent the i th MA 's state at time 0. The i th MA 's history at time t is defined as the sequence $H_t^i = s_0^i, s_1^i, \dots, s_t^i$.

Suppose that the two MA s rendezvous at time t . This implies that at time t , $d = 0$. Consider the first time, $t^* < t$, that d is not equal to $n/2$. Lemma 3.2 implies that this will also be the first time that, respectively, $CLToToken$ and $CWToToken$ are not equal for the two MA s. At time t^* , d (and thus $CLToToken$ and $CCToToken$) changes because either:

1. one MA executed DM and the other MA executed ML or MR so that d changed by one
2. one MA executed ML and the other MA executed MR so that the distance changed by two

Since d changed at time t^* , the MA s necessarily executed conditional statements at that time and received different values. This implies that the histories of the MA s differed at time $t^* - 1$. Let time $t' \leq t^*$ denote the first time that the histories of the MA s differ. If the histories of the MA s first differ at time t' , then Lemma 3.1 implies that a single atomic operation will leave the states unchanged. The states at time t' could differ only if a conditional statement evaluated at that time yielded different values for the two MA s, which implies that the histories of the MA s differed at time $t' - 1$. This contradicts the fact that t' was the first time that the histories differed. This completes the proof of Theorem 3.1. ■

3.4 The Time and Memory Requirements of Rendezvous

While Theorem 3.1 proves that identical stationary tokens cannot be used to break symmetry when $d = \frac{n}{2}$, Theorems 3.2 and 3.3 below provide the lower and upper bounds for the time requirements of the rendezvous search problem when symmetry can be broken using identical stationary tokens, i.e., $d < \frac{n}{2}$.

Theorem 3.2 *Consider two identical MA s at distance $d < \frac{n}{2}$ from each other in an anonymous, synchronous, n node ring. The MA s know that $d < \frac{n}{2}$ and are assumed to have enough memory to count and store information as necessary. When the MA s simultaneously run the same deterministic algorithm for the rendezvous search*

problem, the lower bounds stated in Table 3.1 hold.

Proof of Theorem 3.2. Consider the case where the *MA*s know d . The two *MA*s begin the rendezvous search problem with the same state. After a *MA* travels d nodes in given direction from its starting node, it has new information because it either finds a token or learns that the token is in the other direction from the starting node. The *MA*s are still at least d nodes apart, however, so a rendezvous will require that the *MA*s travel at least another $\frac{d}{2}$ nodes. Since the *MA*s must travel at least $\frac{3d}{2}$ nodes for a rendezvous to occur and moving to an adjacent node takes one unit of time, a rendezvous requires at least $\frac{3d}{2}$ units of time when d is known. This proves the lower bound for all cases where d is known.

Now assume that the *MA*s do not know d but they know n , the number of nodes in the ring. One of the *MA*s may find a token after travelling d nodes in a given direction. The other *MA*, however, will not have any new information until it travels $\frac{n}{2}$ nodes in the same direction. At that time, the *MA*s will still be at least d nodes apart. Given that the *MA*s must travel at least $\frac{n+d}{2}$ nodes for a rendezvous to occur, and moving to an adjacent node takes one unit of time, then a rendezvous requires at least $\frac{n+d}{2}$ units of time when n is known and d is unknown. This proves the lower bound for all cases where n is known and d is unknown.

In the remaining cases, the *MA*s know neither n nor d . A *MA* can count m , the number of nodes from its starting node to the first token, but it cannot tell if $m = d$ or $m = n - d$ until it has travelled around the ring and counted n . When the *MA*s

return to their respective starting nodes, they are still d nodes apart. Given that the MAs must travel at least $n + \frac{d}{2}$ nodes for a rendezvous to occur, and moving to an adjacent node takes one unit of time, then a rendezvous requires at least $n + \frac{d}{2}$ units of time when n and d are unknown. This proves the lower bound for all cases where n and d are unknown. This completes the proof of Theorem 3.2. ■

The upper bounds for the time complexities stated in Table 3.1 are proven in Theorem 3.3. They are derived by constructing a solution that ensures a rendezvous under the given conditions, e.g., when d and the orientation of the ring are known but n is not necessarily known. While the calculation of the lower bounds in Theorem 3.2 assumes that the MAs have sufficient memory to count and store information, the memory requirements for the upper bounds are those of the corresponding solution.

Theorem 3.3 *Consider two identical MAs at distance $d < \frac{n}{2}$ from each other in an anonymous, synchronous n node ring. The MAs know that $d < \frac{n}{2}$. When the MAs simultaneously run the same deterministic algorithm for the rendezvous search problem, the upper bounds stated in Table 3.1 hold. In particular, with the exception of the case where d is known and the ring is not oriented, the lower bounds of Theorem 3.2 are tight.*

Proof of Theorem 3.3. The following four mutually exclusive cases exhaustively describe the subsets of knowledge that a MA may possess about a ring, i.e., n , d , or the ring's orientation.

Case 1: Each mobile agent has memory $O(\log d)$, knows d and the ring's orientation, but may or may not know n . With d and the orientation known, one of the mobile agents is guaranteed to find a token in d steps.

Algorithm for Case 1

1. Release the token.
2. Begin walking around the ring in a counter-clockwise direction.
3. If a token is found within d steps, continue walking in the same direction.
4. Otherwise, if no token is found by d steps, reverse direction and continue walking.

The mobile agent that finds a token at d steps continues walking in the same direction. The other mobile agent reverses its direction and continues walking. Ren-

rendezvous will occur in time $\frac{3d}{2}$.

Case 2: Each mobile agent has memory $O(\log d)$, knows d , does not know the ring's orientation, and may or may not know n . Given d , a mobile agent needs to walk at most d steps before deciding whether to reverse direction.

Algorithm for Case 2

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. If a token is found within d steps, continue walking in the same direction.
4. Otherwise, if no token is found by d steps, reverse direction and continue walking.

In the worst case, $n > 3d$ and the two mobile agents choose opposite directions in step 2 of the algorithm such that they are walking on the *long* side of the ring. They each walk d steps without reaching a token, so they then reverse direction and walk until rendezvous occurs. At time $2d$, each mobile agent is back at its original position, i.e., where it released its token in step 1, so the mobile agents will rendezvous in an additional $\frac{d}{2}$ steps. As a result, the algorithm ensures a rendezvous in time $\frac{5d}{2}$.

Case 3: Each mobile agent has memory $O(\log n)$, knows n , does not know d and may or may not know the orientation. Given n , a mobile agent walks at most $\frac{n}{2}$ steps before deciding whether to reverse direction.

Algorithm for Case 3

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. If a token is found within $\frac{n}{2}$ steps, continue walking in the same direction.
4. Otherwise, reverse direction at $\frac{n}{2}$ steps and continue walking.

In the worst case, the two mobile agents choose the same direction in step 2 of the algorithm. One of the mobile agents walks $n/2$ steps, finds no token, and thus reverses direction and continues walking. The other mobile agent walks d steps, finds a token, maintains the current direction, and continues walking. As a result, rendezvous occurs in time $\frac{n+d}{2}$.

Case 4: Each mobile agent has memory $O(\log n)$, may or may not know the ring's orientation, but does not know n or d .

Algorithm for Case 4

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. Count the number of steps to the first token, δ_1 , and continue walking.
4. Count the number of steps to the second token, δ_2 .
- /*The mobile agent is back at its starting node.*/
5. If $\delta_1 < \delta_2$, continue walking in the same direction.
6. Otherwise, reverse direction and continue walking.

In the worst case, the two mobile agents choose the same direction in step 2 of the algorithm and thus rendezvous will occur in time $n + \frac{d}{2}$. This completes the proof of Theorem 3.3. ■

All of the algorithms discussed so far require either $O(\log d)$ or $O(\log n)$ memory. In the *special case* where d is known to be less than or equal to $\frac{n}{3}$, there exists an algorithm that achieves rendezvous with memory $O(1)$.

Theorem 3.4 *Consider two identical MAs at distance d from each other in an anonymous, synchronous n node ring. In this special case, the MAs know that $d \leq \frac{n}{3}$ but do not know n , d , or the orientation of the ring. Let each MA have memory $O(1)$. The mobile agent rendezvous problem can be solved in time $\frac{n+d}{2}$.*

Proof of Theorem 3.4. Each mobile agent has memory $O(1)$, knows that $d \leq \frac{n}{3}$, i.e., d is at most *one third* of n , but does not know n , d , or the orientation.

Algorithm for $O(1)$ memory case when $d \leq \frac{n}{3}$

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. At the first token, reverse direction and continue walking until rendezvous occurs.

In the worst case, the MAs will travel in the same direction. One MA will find a token after travelling d nodes in the given direction, and will reverse direction and continue to travel. At time $2d$, this MA will have returned to its original position.

The other *MA*, however, has not yet changed direction since $d \leq \frac{1}{3}$. The two *MA* will rendezvous in an additional $\frac{n-3d}{2}$ steps. Thus the two *MA* will rendezvous in time $\frac{n+d}{2}$. This completes the proof of Theorem 3.4. ■

While the preceding solutions assume that the *MA*s know $d < \frac{n}{2}$, the solutions can be easily changed so that the *MA*s stop if $d = \frac{n}{2}$. Suppose that in all cases, the *MA*s have $O(\log n)$ memory. If the intertoken distance is $d = \frac{n}{2}$, then the *MA* can stop as rendezvous is impossible. Otherwise, the *MA* proceeds as before.

3.5 Time and Memory Tradeoff

With $O(\log n)$ memory, the *MA*s can detect if $d = \frac{n}{2}$ and act appropriately, i.e., stop if $d = \frac{n}{2}$ and rendezvous otherwise. If the *MA*s did not know n , d , or the orientation of the ring, the time complexity of the solution is $n + \frac{d}{2}$. When the knowledge of the *MA*s is unchanged but their memory is reduced to $O(\log \log n)$, the following algorithm can detect if $d = \frac{n}{2}$ and act appropriately. The time complexity of the algorithm is $O(\frac{n \log n}{\log \log n})$, which suggests that the *MA*s' memory size can be reduced at the expense of a solution's time complexity.

Algorithm 1

Let $p_1, \dots, p_r$ denote the first r prime numbers such that $\prod_{i=1}^r p_i > n$.

1. Release the token.
2. Set $m = p_1$.
3. Choose a direction and begin travelling around the ring.

4. Count the number of steps, mod m , to the first token, δ_1 , and continue walking.
5. Count the number of steps, mod m , to the second token, δ_2 .
- /* The MA is back at its starting node. */
6. If $\delta_1 \bmod m = \delta_2 \bmod m$,
 - if $m = p_r$, stop. /* Rendezvous is not possible */
 - if $m < p_r$, set $m = p_{i+1}$ and repeat from step 4.
7. If $\delta_1 \bmod m < \delta_2 \bmod m$, continue travelling in the same direction.
8. Else, reverse direction and continue travelling.
- /* If step 7 or 8 is executed, rendezvous occurs in another $\frac{d}{2}$ steps. */

Theorem 3.5 *Consider two identical MAs with memory $O(\log \log n)$ that are distance d from each other in an anonymous, synchronous n node ring. The MAs do not know n , d , the orientation of the ring, or if $d = \frac{n}{2}$. The MAs simultaneously run the same deterministic algorithm. In the rendezvous search problem for two such MAs, Algorithm 1 correctly detects if $d = \frac{n}{2}$ and acts appropriately, i.e., stops if $d = \frac{n}{2}$ and ensures rendezvous otherwise. The time complexity of the algorithm is $O(\frac{n \log n}{\log \log n})$.*

Proof of Theorem 3.5. The Chinese Remainder Theorem implies that if $d \equiv (n - d) \bmod p_i$ for all $p_i, i = 1, \dots, r$, then $d \equiv (n - d) \bmod \prod_{i=1}^r p_i$. Let the p_i be the first r prime numbers such that $\prod_{i=1}^r p_i > n$. The algorithm checks each p_i in turn to see if $d \equiv (n - d) \bmod p_i$. If the statement is true for all p_i , then $d = n - d = \frac{n}{2}$ and the algorithm stops since rendezvous is impossible. If $d \not\equiv (n - d) \bmod p_i$ for some

p_i , however, then $d < \frac{n}{2}$. The first time the algorithm discovers such a p_i , one of the M As reverses its direction and a rendezvous occurs $\frac{d}{2}$ steps later.

The worst case occurs when $d = \frac{n}{2}$ since all r of the p_i values have to be checked. The resulting run time is $O(rn)$, but we need to determine the value of r .

Consider the smallest r such that $\prod_{i=1}^r p_i > n$. This implies that $\prod_{i=1}^{r-1} p_i \leq n$ and thus $\prod_{i=1}^r p_i \leq n * p_r \leq n^2$. As a result, $\prod_{i=1}^r p_i \in O(n^2)$.

Let $\pi(m)$ denote the number of prime numbers less than or equal to m . Apostol [21] states that for any integer m , $\frac{m}{6 \log m} \leq \pi(m) \leq \frac{8m}{\log m}$. Since p_i is prime by definition, $\pi(p_i) = i$ for all i so that $\frac{p_i}{6 \log p_i} \leq \pi(p_i) = i \leq \frac{8p_i}{\log p_i}$ and thus $p_i \geq \frac{i \log p_i}{8}$. Taking the product over all i and using Stirling's approximation [30] results in

$$\prod_{i=1}^r p_i \geq \prod_{i=1}^r \frac{i \log p_i}{8} = r! 8^{-r} \prod_{i=1}^r \log p_i \geq r! 8^{-r} \geq 2^{\Omega(r \log r)} \quad (3.1)$$

The logarithm of equation 3.1, given that $\prod_{i=1}^r p_i \in O(n^2)$, indicates that $O(\log n) \geq r \log r$, so a valid value for r must satisfy $r \log r \leq c \log n$.

Let $r = (\log n)^{1-\epsilon}$. Substituting for r yields $(1-\epsilon) \log \log n \leq c(\log n)^\epsilon$ so that, given a constant $c > 0$, a corresponding value of ϵ can be calculated such that $(\log n)^{1-\epsilon}$ is a valid value for r . The maximum value for r is then at least $(\log n)^{1-\epsilon}$. Substituting $(\log n)^{1-\epsilon}$ for r in $r \log r \leq c \log n$ yields $r \log(\log n)^{1-\epsilon} \leq c \log n$ such that $r \leq \left(\frac{c}{1-\epsilon}\right) \frac{\log n}{\log \log n}$ and thus $r \in O\left(\frac{\log n}{\log \log n}\right)$. As a result, the time complexity of the algorithm, $O(rn)$, is $O\left(\frac{n \log n}{\log \log n}\right)$. This completes the proof of Theorem 3.5. ■

3.6 Lower Bound for Memory

While the discussion in section 3.5 implies that the *MA*s' memory size can be reduced at the expense of a solution's time complexity, the following theorem proves that Algorithm 1 achieves the lower bound of memory required to solve the rendezvous search problem under the given model.

Theorem 3.6 *Consider two identical *MA*s that are distance d from each other in an anonymous, synchronous n node ring. The *MA*s do not know n , d , the orientation of the ring, or if $d = \frac{n}{2}$. The *MA*s simultaneously run the same deterministic algorithm. To begin an algorithm, each *MA* places an identical token on its respective starting node and these tokens remain in place for the rest of the algorithm. Any algorithm that correctly detects if $d = \frac{n}{2}$ and acts appropriately, i.e., stops if $d = \frac{n}{2}$ and ensures rendezvous otherwise, requires that each *MA* have at least m bits of memory, where m is $\Omega(\log \log n)$.*

Proof of Theorem 3.6. Consider a unidirectional ring of n nodes where $k = 2$ *MA*s are placed $d = \frac{n}{2}$ nodes apart. Each *MA* has m bits of memory. Without loss of generality, label one of the *MA*s as MA_1 and the other as MA_2 . Let Γ denote the non-empty collection of algorithms that correctly solve the rendezvous search problem for two *MA*s with m bits of memory each, i.e., every algorithm in Γ stops if $d = \frac{n}{2}$ and otherwise ensures a rendezvous. Let A denote an arbitrary algorithm in Γ .

Construct the following sequence of bits for each node in the ring. Let the first m bits in the sequence for a given node be the m bits stored in the memory of the

first MA to visit that node. Every time that a MA visits a node, the m bits stored in the MA 's memory at that time are appended to the sequence. Let r denote the number of complete trips around the ring completed by a MA in algorithm A . Given a unidirectional ring, the two MAs will complete the same number of rounds when $d = \frac{n}{2}$, otherwise one MA would pass the other and rendezvous would occur. After r rounds of algorithm A , the resulting sequence of bits for a given node will contain at least krm bits, where $k = 2$ is the number of MAs . The $2rm$ bits stored at a given node represent the histories, i.e., the interleaved sequence of states, of the MAs at that node. In a ring with n nodes where n is sufficiently large, i.e., $n \gg 2rm$, two such constructed sequences of bits will be identical, i.e., two distinct nodes will have the same sequence of $2rm$ bits. If n is large enough, these two nodes will not be separated by a token.

Let $node_a$ and $node_b$ denote the two nodes with identical bit sequences, where $node_a$ precedes $node_b$ in the unidirectional ring. By the construction of the bit sequences, MA_1 is in the same state at $node_a$ as it is at $node_b$ for *each* of the r complete rounds of the algorithm. The same is true for MA_2 . If all of the nodes from $node_a$ up to, but not including, $node_b$ are removed from the ring, the behaviour of MA_1 and MA_2 at $node_b$, and any other remaining node, is unchanged. This implies that the behaviour of the two MAs , under the given algorithm A , is the same in the original case where $d = \frac{n}{2}$ as in the case where nodes are deleted such that $d < \frac{n}{2}$. This contradicts the assumption that algorithm A will correctly stop without a rendezvous when $d = \frac{n}{2}$ and ensure a rendezvous otherwise.

Algorithm A is an arbitrary algorithm chosen from Γ , the collection of algorithms that correctly solve the rendezvous search problem for two MA s with m bits of memory each. The algorithm fails because $n \gg 2rm$. We want to find the value of m , i.e., the amount of memory available to a MA , such that for a given n , $n \leq 2rm$ holds and thus algorithm A and any other algorithm arbitrarily chosen from Γ will succeed.

In r rounds, the two MA s will deposit a total of $2rm$ bits at each of the n nodes. These $2rm$ bit sequences represent 2^{2rm} different strings, so n must exceed $c_0 2^{2rm}$, where c_0 is a non-zero positive integer constant, if any of the strings are to be repeated at different nodes, i.e., as in $node_a$ and $node_b$ above.

Consider r , the number of rounds of the arbitrary algorithm A . We claim that $r \leq 2^{km}$ where $k = 2$. Suppose not. If $r > 2^{2m}$, then both MA s will eventually return to their respective starting nodes with the same bits in memory as when they started.

With $r \leq 2^{2m}$, then $c_0 2^{2rm} \leq c_0 2^{m2^{m+1}}$ and $n \geq c_0 2^{m2^{m+1}}$ will cause the algorithm A to fail. Thus each MA needs m bits of memory such that $n < c_0 2^{m2^{m+1}}$ in order to guarantee the correctness of algorithm A . This implies that $\log n < m2^{m+1}$ so that $\log \log n < 2m + \log m + 1$. Thus the correctness of an arbitrary algorithm in Γ is guaranteed if each MA has at least m memory bits where $m \in \Omega(\log \log n)$. This completes the proof of Theorem 3.6. ■

Together, Theorem 3.5 and Theorem 3.6 imply that $\Theta(\log \log n)$ memory bits are required to ensure the correctness of an algorithm for the rendezvous search problem for two identical MA s with identical tokens on an anonymous, synchronous n node

ring.

3.7 Movable Tokens

The preceding theorem indicates that if the MA s have $O(1)$ memory and do not know n , d , or the orientation of the ring then, with the exception of some special cases, the rendezvous search problem cannot be solved in the given model. We need to change the model if we want to solve the rendezvous search problem with $O(1)$ memory for arbitrary values of d and n such that $d < \frac{n}{2}$. Suppose that, upon meeting a token, a MA can move that token to an adjacent node.

Algorithm 2 (The Movable Tokens Algorithm)

1. Release the token.
2. Choose a direction and begin walking.
3. Upon finding a token, reverse direction.
4. Move the token one node in the new direction.
5. Continue walking in the new direction.
6. Repeat from step 3 until rendezvous occurs.

Theorem 3.7 *Consider two identical MA s with memory $O(1)$ in an anonymous, synchronous, n node ring. The MA s do not know n , d , or the orientation of the ring. If, upon reaching a token, a MA can move the token to an adjacent node, then the mobile agent rendezvous problem can be solved in time $O\left(d^2 \left(\frac{d^2}{n-2d} + n\right)\right)$. Thus, as d*

approaches 1, the time approaches $O(n)$ and, as d approaches $\frac{n}{2}$, the time approaches $O(n^4)$.

Proof of Theorem 3.7. In the worst case, the mobile agents will choose the same direction in step 2 of the algorithm. Note that because $d < n-d$, the distance between the two tokens does not increase in the course of the algorithm. It is important to establish when the distance between the two tokens decreases.

Suppose that mobile agent MA_1 has walked back and forth between the tokens i times and has returned to its starting position, i.e., the node where it released its token. Suppose that the second mobile agent MA_2 has walked back and forth between the tokens j times and has walked r additional steps away from its starting position. The distance between the two tokens will decrease by 1 step if the first mobile agent MA_1 can move both tokens before the other mobile agent MA_2 reaches the next token. This requires that

$$\begin{aligned} id &= j(n-d) + r \\ r &\geq 0 \\ n-d-r &> d \end{aligned} \tag{3.2}$$

for some $0 \leq j < i$ and r .

These requirements imply that if

$$\frac{j}{i+j} \leq \frac{d}{n} < \frac{j+1}{i+j+2} \tag{3.3}$$

then the distance between the two tokens will decrease to $d - 1$ in at most

$$id + n - d = (i - 1)d + n \quad (3.4)$$

steps. Setting $j = i - 1$ in Equation 3.3 results in the expression

$$\frac{i - 1}{2i - 1} \leq \frac{d}{n} < \frac{i}{2i + 1} \quad (3.5)$$

The left-hand side of Equation 3.5 is equivalent to $i - 1 \leq \frac{d}{n - 2d}$. As well, when

$$d - \frac{i - 1}{2i - 1}n < k \leq d - \frac{i - 2}{2i - 3}n \quad (3.6)$$

holds, the distance between the two tokens will decrease to $d - k$ and the ratio $\frac{d - k}{n}$ will satisfy

$$\frac{i - 2}{2i - 3} \leq \frac{d - k}{n} < \frac{i - 1}{2i - 1} \quad (3.7)$$

It follows that the distance between the tokens will decrease to at most $d - k$ in at most

$$k((i - 1)d + n) \leq k \left(\frac{d^2}{n - 2d} + n \right) \leq \left(d - \frac{i - 2}{2i - 3}n \right) \left(\frac{d^2}{n - 2d} + n \right) \quad (3.8)$$

steps, so rendezvous occurs in time

$$O \left(d^2 \left(\frac{d^2}{n - 2d} + n \right) \right).$$

This completes the proof of Theorem 3.7. ■

Chapter 4

Mobile Agent Rendezvous with More than Two Mobile Agents

4.1 The Network Model

The network model described in Chapter 3 is extended to include $k \geq 2$ MAs. A given node now requires enough memory to host a token and at most k MAs. Each *MA* still owns a single identical token and the *MAs* always place their tokens on their respective starting nodes in the first step of any algorithm. Because the tokens are stationary, the intertoken distances that exist at the beginning of an algorithm persist throughout the algorithm. As in the previous chapter, all algorithms discussed in this chapter are deterministic.

Algorithm	Memory	Time
3	$O(k \log n)$	$O(n)$
4	$O(\log n)$	$O(kn)$
5	$O(k \log \log n)$	$O(\frac{n \log n}{\log \log n})$
6 ⁴	$O(\log n)$	$O(n)$
7 ⁴	$O(\log k)$	$O(n)$
8 ⁴	$O(\log k)$	$O(n \log k)$

Table 4.1: The Rendezvous Search Problem with $k \geq 2$ Mobile Agents

4.2 Overview of Chapter 4

First, we prove that if neither k , the number of MAs , nor n , the number of nodes in the ring, are known then the rendezvous problem is unsolvable in the given model. For the remainder of the section, we assume that k is known. Next, we prove that when k is known, rendezvous is possible if and only if $\mathcal{S}$, the sequence of intertoken distances, is aperiodic.

We present three algorithms that solve the rendezvous problem. When $\mathcal{S}$, the sequence of intertoken distances between the MAs , is aperiodic, the algorithms guarantee that rendezvous occurs. However, if $\mathcal{S}$ is periodic and thus rendezvous is impossible, the algorithms guarantee that the MAs stop. The memory and time complexities for these algorithms are presented in Table 4.1. We also present three algorithms that solve the rendezvous problem for values of k and n that satisfy various conditions such as primality. The memory and time complexities for these algorithms are also presented in Table 4.1.

⁴In this algorithm, the values for k and n must satisfy various conditions such as primality.

In the previous chapter, we proved that solving the rendezvous search problem with $k = 2$ mobile agents requires $\Theta(\log \log n)$ memory. In this section, we prove that solving the rendezvous problem with $k \geq 2$ mobile agents requires at least $\Omega(\log \log n + \log k)$ memory.

Finally, we prove that if the MAs share a common orientation, then the rendezvous problem and the leader election problem for MAs are equivalent. A solution to the first problem can be used to derive a solution for the second problem and vice versa. If the MAs do not share a common orientation, however, then the leader election problem for MAs is strictly more complex than the rendezvous problem since a solution to the latter problem does not always imply a solution to the former problem.

4.3 The Feasibility of Rendezvous

To solve the rendezvous problem, mobile agents must recognize when rendezvous is possible. As the next theorem shows, knowledge of k or n is a necessary condition for the rendezvous of identical MAs in an anonymous, synchronous ring.

Theorem 4.1 *When each MA in the ring knows neither n , the number of nodes, nor k , the number of mobile agents, the mobile agent rendezvous problem is unsolvable in the given model.*

A proof of Theorem 4.1 follows directly from the proof of Theorem 2.1 in Barriere et al [22]. Unlike our model, each network node in the Barriere et al [22] model has

a whiteboard that the *MA*s can access with mutual exclusion.

Theorem 4.1 indicates that knowing either k or n is a necessary condition for solving the rendezvous problem. For the remainder of the thesis, we shall assume that k is known unless it is stated otherwise.

As mentioned earlier, we assume that the *MA*s always place their tokens on their respective starting nodes in the first step of any algorithm they execute. The tokens are identical and stationary so that the intertoken distances that exist at the beginning of an algorithm persist throughout the algorithm. In this section, we prove that rendezvous is possible if and only if the sequence of intertoken distances in the ring is aperiodic.

If $\mathcal{S} = d_1, \dots, d_k$, the sequence of intertoken distances, is periodic, then $\gcd(k, n) = g > 1$ and the k *MA*s can be placed on an n node ring as described in Barriere et al [22]:

- i) denote the nodes adjacent to node x_i as x_{i+1} and x_{i-1} where the operations on the node indices are mod n ,
- ii) starting at node x_1 , divide the n nodes of the ring into g non-intersecting segments of n/g nodes each,
- iii) the first such segment, σ_1 , is comprised of the nodes x_1 to $x_{n/g}$ inclusive,
- iv) arbitrarily place k/g *MA*s in segment σ_1 such that no node holds more than one *MA*,
- v) place the remaining $k - (k/g)$ *MA*s such that if there is a node at x_i ,

then there is a *MA* on node $x_{i+j(n/g)}$ for $1 \leq j \leq g - 1$.

As a result, each *MA* initially has a *sibling MA* exactly n/g nodes away in either direction.

Let the atomic operations in any rendezvous algorithm be restricted to those defined in Section 3.3. In addition to these atomic operations, an algorithm may contain two constructs. In a **do-until loop**, a *MA* performs a finite sequence of atomic operations until a given state occurs, and if a given condition holds in an **if-then statement**, a *MA* performs a finite sequence of atomic operations.

Lemma 4.1 *Assume that the MAs are initially placed as described above so that each MA is exactly n/g nodes away from its respective siblings. If the MAs simultaneously start to execute an identical algorithm and that algorithm is restricted to a finite sequence of atomic operations, i.e., there are no do-until loops or if-then statements, then the distance between a mobile agent and its respective siblings will not change.*

Proof of Lemma 4.1. Consider an algorithm comprised of a single atomic operation. Any one of the atomic operations listed above will leave the distance between every pair of siblings unchanged, since either both *MAs* are stationary or each *MA* moves one node in the same direction. Assume that this is true for an algorithm that is a sequence of m atomic operations.

Consider an algorithm that is a sequence of $m + 1$ atomic operations. Assuming that the distance between every pair of siblings is unchanged after m atomic operations, it will also be unchanged if the $m+1$ th operation is one of *RT*, *QT*, *QMA*, *IC(c)*, *DM*

since the MAs are stationary under these atomic operations. The distance between every pair of siblings is also unchanged if the $m + 1$ th operation is one of MCL or MCC since both MAs move one node in the same direction. Thus the distance between every pair of siblings is unchanged after $m + 1$ atomic operations. This completes the proof of Lemma 4.1. ■

Lemma 4.2 *Assume that the MAs are initially placed as described above so that each MA is exactly n/g nodes away from its respective siblings. Let any MA and its two siblings be referred to as a trio. For a given MA , let $CLToToken$ and $CCToToken$ denote the distance to the nearest tokens in the clockwise and counterclockwise directions respectively. Initially, the MAs comprising a trio have identical values for $CLToToken$ and identical values for $CCToToken$. If time t^* is the first time in an algorithm that at least one MA is not exactly n/g nodes away from its respective siblings, then t^* is also the first time that the respective values of $CLToToken$ (and thus $CCToToken$) are not equal for that MA and its trio.*

Proof of Lemma 4.2. For all $t < t^*$, each MA is exactly n/g nodes away from its respective *siblings* and thus, given the placement of the tokens, either all the MAs of a given *trio* $\mathcal{T}$ are on separate tokens or none of the MAs in $\mathcal{T}$ are on tokens. If the MAs in $\mathcal{T}$ are on separate tokens at time $t < t^*$, then those MAs have identical values for $CLToToken$ and identical values for $CCToToken$.

Suppose that at time t^* , none of the MAs in $\mathcal{T}$ are on tokens. Suppose further that there exists a time $t' < t^*$ such that t' is the first time that the *trio's* respective

values for $CLToken$ differ. Choose two of the MAs , A and B , whose values for $CLToken$ differ. Without loss of generality, assume that A 's value for $CLToken$ is less than B 's value for $CLToken$, i.e., $CLToken_A < CLToken_B$. If both A and B moved $CLToken_A$ nodes in a clockwise direction, then A would be on a node containing a token but B would not. Given the placement of the tokens, this implies that MAs A and B are not n/g nodes apart at time t' and thus contradicts the fact that t^* is the first time that at least one MA is not exactly n/g nodes from its respective *siblings*. This completes the proof of Lemma 4.2. ■

Theorem 4.2 *Rendezvous is guaranteed if and only if $\mathcal{S}$, the sequence of intertoken distances, is aperiodic.*

Proof of Theorem 4.2.

Part A: Suppose not, i.e., rendezvous is guaranteed but $\mathcal{S}$, the sequence of intertoken distances, is periodic. The k MAs can be placed on the ring as described on page 56. This is the MA arrangement that is assumed in Lemmata 4.1 and 4.2. Lemma 4.1 implies that if a MA begins an algorithm with its *siblings* at a distance of n/g nodes and the algorithm contains only atomic operations, then the MA will not rendezvous with its *siblings*. The inclusion of a *do-until* loop or an *if-then* statement is therefore a necessary condition for rendezvous.

The boolean value of the conditional statement in a *do-until* loop or an *if-then* statement determines, respectively, how many iterations of the loop occur or if the body of the *if-then* statement is executed. In either case, a conditional statement

evaluated at time t is a function of the known history of the given MA at time $t - 1$.

Let s_0^i represent the i th MA 's state at time 0. The history of the i th MA at time t is defined as the sequence $H_t^i = s_0^i, s_1^i, \dots, s_t^i$. If the k MAs rendezvous at time t , let $t^* < t$ denote the first time that the distance between some MA and at least one of its *siblings* is not n/g . Denote this *trio* of MAs by $\mathcal{T}$. Lemma 4.2 indicates that t^* is also the first time that the values of *CLToken* differ among the MAs in $\mathcal{T}$.

At time t^* , the distance between an MA in $\mathcal{T}$ and at least one of its *siblings* changes because the MA and its *sibling* executed the same conditional statements but received different Boolean values as a result. This implies that the histories of the MA and its *sibling* differed at time $t^* - 1$. Let $t' < t^*$ denote the first time that the histories of these MAs differed, i.e., the MAs ' respective states differ at time t' . The states of the two MAs can differ at time t' only if conditional statements evaluated at time $t' - 1$ yielded different values. This implies, however, that the histories of the MAs differed at time $t' - 2$ and contradicts the fact that the MAs ' histories first differed at time $t^* - 1$.

As a result, the distance between a MA and at least one of its *siblings* is always n/g nodes and thus rendezvous cannot occur when $\gcd(k, n) = 1$. This contradicts the fact that rendezvous is guaranteed and thus statement 1 implies statement 2.

Part *B*: Suppose not, i.e., $\mathcal{S}$, the sequence of intertoken distances, is aperiodic but rendezvous is not guaranteed. If rendezvous cannot occur, then the rotations of the sequence $\mathcal{S}$ have no lexicographic maximum. Otherwise the MA could rendezvous at the node that begins the lexicographically maximum rotation. This implies, however,

that the sequence of intertoken distances is periodic. This contradicts the fact that $\mathcal{S}$, is aperiodic and thus statement 2 implies statement 1. This completes the proof of Theorem 4.2. ■

4.4 Unconditional Solutions

Given the results of the previous section, the mobile agent rendezvous problem can be solved when the MA s can determine that $\mathcal{S}$, the sequence of intertoken distances, is aperiodic.

First, assume that each MA has $O(k \log n)$ memory. The MA s know k but do not necessarily know n or share a common orientation. Consider the following algorithm.

Algorithm 3

1. Release the token at the starting node.
2. Choose a direction and begin to walk around the ring.
3. Compute the k intertoken distances $d_1, \dots, d_k$.
4. If $\mathcal{S} = d_1, \dots, d_k$ is periodic, then stop. (Rendezvous is not guaranteed.)
5. Set $forward = d_1, \dots, d_k$ and $reverse = d_k, \dots, d_1$.
6. Let $lexi(someSequence)$ denote the lexicographically maximum rotation of the sequence $someSequence$.
7. Set $forward = lexi(forward)$ and $reverse = lexi(reverse)$.
8. If $forward$ and $reverse$ differ, then determine which of these sequences is the lexicographic maximum and rendezvous at the node where this

sequence starts.

9. Else let MA_i and MA_j denote the MA s at the beginning of *forward* and *reverse* respectively.
10. If MA_i and MA_j are the same MA , then rendezvous at the node where MA_i resides.
11. If MA_i and MA_j are distinct MA s, then look at the two paths between MA_i and MA_j in the ring.
 - i) If only one of the paths had an odd number of nodes, then rendezvous at the node in the midpoint of that path.
 - ii) If both paths have an odd number of nodes, then
 - a) if the paths differ in length, rendezvous at the midpoint of the shorter path,
 - b) else compare the sequences of intertoken distances for the two paths and rendezvous at the node in the midpoint of the path that is the lexicographic maximum.
 - iii) If both paths have an even number of nodes, then rendezvous at the node in the midpoint of the path that contains an odd number of MA s.

Theorem 4.3 *If each MA has memory $O(k \log n)$, then the mobile agent rendezvous search problem can be solved in time $O(n)$.*

Proof of Theorem 4.3. With $O(k \log n)$ memory, each MA can calculate $\mathcal{S}$, the

sequence of intertoken distances, after walking around the ring once. There is sufficient memory to determine if $\mathcal{S}$ is periodic in step 4 of Algorithm 3 and thus the MAs will stop in step 5 if rendezvous is not possible. If the MAs reach step 6 of Algorithm 3, $\mathcal{S}$ is aperiodic and thus the *forward* and *reverse* sequences in step 7 have unique lexicographically maximum rotations. Steps 8 to 11 in Algorithm 3 ensure that the respective lexicographically maximum rotations of the *forward* and *reverse* sequences can be used to identify a unique node at which rendezvous will occur. Since the calculation of the initial sequence of intertoken distances and the determination of periodicity occurs in $O(n)$ time, Algorithm 3 guarantees that the rendezvous problem is solved in time $O(n)$. This completes the proof of Theorem 4.3. ■

If the MAs are restricted to memory $O(\log n)$, the mobile agent rendezvous problem is still solvable.

Consider the following algorithm. In each round, a MA may become inactive and thus spend the rest of the algorithm at its starting node. If the MAs share a common orientation, then the MAs travel in the same direction, so an active MA can identify an inactive MA because the former will find the latter stopped. If the MAs do not share a common orientation, however, an active MA that meets another MA can not tell if the latter MA is inactive or merely travelling in the opposite direction. In this case, each MA needs to set a bit that indicates whether it is active or inactive.

With only $O(\log n)$ memory, a MA needs more than one traversal of the ring to determine if $\mathcal{S}$ is aperiodic.

Algorithm 4

1. Release the token at the starting node.
2. Set $c = 1$. (The number of the current round.)
3. Set $active = 1$. (A bit to indicate whether the MA is active.)
4. Set $inactive = 0$ and $laggedInactive = 0$. (Count the number of inactive MAs .)
5. Choose a direction and begin to walk around the ring.
6. Increment $inactive$ each time an inactive MA is met.
7. Compute the distance to the c th token, d_c , i.e., if $c = 1$, count the distance to the first token and if $c = 2$, count the distance to the second token, etc.
8. Continue to walk around the ring and compare d_c to each intertoken distance between c tokens.
9. If MA sees an intertoken distance d_i such that $d_i > d_c$, then the MA continues in the same direction and becomes inactive, i.e., sets $active = 0$, when it reaches its starting node.
10. If MA did not see an intertoken distance d_i such that $d_i > d_c$, then the MA remains active when it returns to its starting node.
11. If only one MA remains active, i.e., $inactive = k - 1$, then walk around the ring and arrange the rendezvous. (A MA that reaches this point is the sole active MA .)

12. If $inactive == laggedInactive$, then stop. (No additional MA s became active in this round.)
13. Else set $c = c + 1$ and $inactive = 0$.
14. Repeat from step 5.

Theorem 4.4 *If each MA has memory $O(\log n)$, then the mobile agent rendezvous problem is solvable in time $O(kn)$.*

Proof of Theorem 4.4. In round c of Algorithm 4, an active MA calculates the intertoken distance between the first c tokens and then walks around the ring comparing this distance to the corresponding distances met. If $\mathcal{S}$, the sequence of intertoken distances, is aperiodic, then in each round of the algorithm, i.e., with each traversal of the ring, at least one MA becomes inactive and at least one MA remains active. Otherwise the intertoken distances are equal. Thus after a given round, if the number of inactive MA s remains unchanged, then rendezvous is impossible. Otherwise, after at most k rounds of the algorithm, one MA will remain active and that MA can arrange the rendezvous. Each round of the algorithm requires a complete traversal of the ring and thus takes time $O(n)$. With at most k rounds required, the resulting time for Algorithm 4 is $O(kn)$. This completes the proof of Theorem 4.4. ■

Algorithms 3 and 4 solve the mobile agent rendezvous problem when each MA has memory $O(k \log n)$ and $O(\log n)$ respectively. It is also possible, however, to solve the mobile agent rendezvous problem when each MA has memory $O(k \log \log n)$.

Let $p_1, \dots, p_r$ denote the first r prime numbers such that $\prod_{i=1}^r p_i > n$. An active

MA needs to recognize if another MA is active. Without a common orientation, each MA needs to set a bit to indicate whether it is active.

Algorithm 5

1. Release token at the starting node.
2. Set $active = 1$.
3. Set p_r to the first prime such that $\prod_{i=1}^r p_i > n$.
4. Set $p_i = p_1 = 2$.
5. Set $\alpha = k$, the number of active MA s.
6. Walk around the ring and compute the intertoken distances mod p_i
between the α active MA s, i.e., $d_1, \dots, d_\alpha \bmod p_i$.
7. Set $forward = d_1, \dots, d_\alpha \bmod p_i$.
8. Set $reverse = d_\alpha, \dots, d_1 \bmod p_i$.
9. If $forward$ is periodic, i.e., $forward = (d_1, \dots, d_{\alpha/a})^a \bmod p_i$, then
 - i) if at start of a block $(d_1, \dots, d_{\alpha/a})$, remain active.
 - ii) else set $active = 0$.
 - iii) if $p_i < p_r$, then
 - a) set $p_i = p_{i+1}$, $\alpha = a$, and repeat from step 6.
 - b) else stop, since rendezvous is impossible.
10. If $forward$ is aperiodic, then let $lexi(someSequence)$ denote the lexicographically maximum rotation of the sequence $someSequence$.
11. Set $forward = lexi(forward)$ and $reverse = lexi(reverse)$.

12. If *forward* and *reverse* differ, then
 - i) determine which of these sequences is the lexicographic maximum and rendezvous at the node where this sequence starts.
 - ii) else let MA_i and MA_j denote the MA s at the beginning of *forward* and *reverse* respectively.
13. If MA_i and MA_j are the same MA , then rendezvous at the node where MA_i resides.
14. If MA_i and MA_j are distinct MA s, then do the following.
 - i) Look at the two paths between MA_i and MA_j in the ring.
 - ii) If only one of the paths had an odd number of nodes, then rendezvous at the node in the midpoint of that path.
 - iii) If both paths have an odd number of nodes, then
 - a) if the paths differ in length, rendezvous at the midpoint of the shorter path,
 - b) else compare the sequences of intertoken distances for the two paths

and rendezvous at the node in the midpoint of the path that is the lexicographic maximum.
 - iv) If both paths have an even number of nodes, then rendezvous at the node in the midpoint of the path that contains an odd number of MA s.

The following lemma is necessary for the proof of Theorem 4.5 below.

Lemma 4.3 *Consider a prime p such that $2 < p \leq r$. Assume that for all primes $p_i < p$, the sequence of distances mod p_i between the α_i active MAs is periodic such that $d_1, \dots, d_{\alpha_i} \equiv (d_1, \dots, d_{\alpha_i/a})^a \pmod{p_i}$ and $a \mid \alpha_i$. Let the first MA in each occurrence of the block $d_1, \dots, d_{\alpha_i/a}$ remain active, while the remaining MAs become inactive. If the sequence of distances mod p between the a active MAs is periodic, then the original intertoken distances can be partitioned into $t \mid k$ equal length blocks with sums $\sigma_1, \sigma_2, \dots, \sigma_t$ that are congruent modulo all the primes $p_1, p_2, \dots, p$.*

Proof of Lemma 4.3.

Base case:

Let $p_i = p_2 = 3$. By assumption, the sequence of distances mod p_1 between the k active MAs is periodic, so that $d_1, \dots, d_k \equiv (d_1, \dots, d_{k/a})^a \pmod{p_1}$ and $a \mid k$. The original intertoken distances are thus partitioned into a equal length blocks:

$$\begin{aligned} b_1^1 &= d_1, \dots, d_{k/a} \\ b_2^1 &= d_{(k/a)+1}, \dots, d_{2(k/a)} \\ &\dots \\ b_a^1 &= d_{(a-1)(k/a)+1}, \dots, d_k \end{aligned}$$

Let σ_i^1 denote the sum of the distances in the block b_i^1 . By construction, $\sigma_1^1 \equiv \sigma_2^1 \equiv$

$\dots \equiv \sigma_a^1 \pmod{p_1}$, i.e., the sums of the distances mod p_1 in the respective blocks are congruent mod p_1 . The MAs at the head of the respective blocks $b_1, \dots, b_a$ remain active, so that number of active MAs becomes a .

If the sequences of distances mod p_2 between the a active MAs is periodic, then $d_1, \dots, d_a \equiv (d_1, \dots, d_{a/a'})^{a'} \pmod{p_2}$ where $a' \mid a \mid k$. Thus the a distances can be partitioned into a' equal length blocks mod p_2 :

$$\begin{aligned} b_1^2 &= d_1, \dots, d_{a/a'} \\ b_2^2 &= d_{(a/a')+1}, \dots, d_{2(a/a')} \\ &\dots \\ b_{a'}^2 &= d_{(a'-1)(a/a')+1}, \dots, d_a \end{aligned}$$

Let σ_i^2 denote the sum of the distances in the block b_i^2 . By construction, $\sigma_1^2 \equiv \sigma_2^2 \equiv \dots \equiv \sigma_{a'}^2 \pmod{p_2}$, i.e., the sums of the distances in the a' respective blocks are congruent mod p_2 . Each block b_i^2 , however, contains the a/a' blocks from $b_{(i-1)(a/a')+1}^1$ to $b_{i(a/a')}^1$ and thus $\sigma_i^2 = \sum_{j=1}^{a/a'} \sigma_{(i-1)(a/a')+j}^1$. As shown above, however, the σ_i^1 terms are congruent mod p_1 for all i and since each σ_j^2 term is the sum of a/a' σ_i^1 terms, then the σ_j^2 terms are also congruent mod p_1 .

Thus the original distances can be partitioned into $a' \mid k$ equal length blocks with

sums $\sigma_1^2, \dots, \sigma_{\alpha/a'}^2$ that are congruent modulo p_1 and p_2 . This proves that the lemma holds for the base case where $p = p_2 = 3$.

Assume that the lemma holds for all primes $p < p_r$.

Let α denote the number of active MAs at the end of the round for p_{r-1} . Suppose that in the next round, the sequence of distances mod p_r between the α active MAs is periodic, i.e., $d_1, \dots, d_\alpha \equiv (d_i, \dots, d_{\alpha/a_r})^{a_r} \pmod{p_r}$ where $a_r \mid \alpha$. The sequence of distances can therefore be partitioned into a_r equal length blocks mod p_r :

$$\begin{aligned} b_1^r &= d_1, \dots, d_{\alpha/a_r} \\ b_2^r &= d_{(\alpha/a_r)+1}, \dots, d_{2(\alpha/a_r)} \\ &\dots \\ b_{a_r}^r &= d_{(a_r-1)(\alpha/a_r)+1}, \dots, d_\alpha \end{aligned}$$

Each block b_i^r , $i = 1, \dots, (\alpha/a_r)$ contains the α/a_r blocks $b_{(i-1)(\alpha/a_r)+1}^{r-1}$ to $b_{i(\alpha/a_r)}^{r-1}$ and thus $\sigma_i^r = \sum_{j=1}^{\alpha/a_r} \sigma_{(i-1)(\alpha/a_r)+j}^{r-1}$.

The inductive hypothesis implies that the blocks b_i^{r-1} , $i = 1, \dots, \alpha$, partition the original intertoken distances into $\alpha \mid k$ equal length blocks with sums σ_i^{r-1} , $i = 1, \dots, \alpha$, that are congruent for all $p \leq p_{r-1}$. Since each of the b_i^r blocks contains α/a_r of the b_i^{r-1} blocks, then the b^r blocks partition the original intertoken distances into

$a_r \mid \alpha \mid k$ equal length blocks. By construction, the sums for the b^r blocks, σ_i^r , are congruent mod p_r for all i , $i = 1, \dots, a_r$. Since each σ_i^r , however, is merely the sum of $\alpha/a_r \sigma_j^{r-1}$ terms and the inductive hypothesis implies that the σ_j^{r-1} are congruent mod p for all $p \leq p_{r-1}$, then the sums σ_i^r , $i = 1, \dots, a_r$ are congruent mod p for all $p \leq p_r$.

Thus for p_r , the original intertoken distances can be partitioned into $a_r \mid k$ equal length blocks with sums σ_i^r , $i = 1, \dots, a_r$, that are congruent for all $p \leq p_r$. This completes the proof of Lemma 4.3. $\blacksquare$

Theorem 4.5 *If each MA has memory $O(k \log \log n)$, then the mobile agent rendezvous search problem can be solved in time $O(\frac{n \log n}{\log \log n})$.*

Proof of Theorem 4.5. Algorithm 5 solves the mobile agent rendezvous problem if it stops the MAs when rendezvous is impossible and otherwise ensures a rendezvous. Suppose that for all $p_i \leq p_r$, the sequence of distances mod p_i between the active MAs is periodic. Algorithm 5 will stop the MAs in step 8 and indicate that rendezvous is impossible. Lemma 4.3 implies that in the last round of Algorithm 5, where $p = p_r$, the original intertoken distances can be partitioned into $a_r \mid k$ equal length blocks with sums σ_i^r , $i = 1, \dots, a_r$, that are congruent mod p for all $p \leq p_r$. The Chinese Remainder Theorem then implies that the sums σ_i^r , $i = 1, \dots, a_r$, are congruent mod $\prod_{i=1}^r p_i$. Since $\prod_{i=1}^r p_i > n$, then the original intertoken distances can be partitioned $a_r \mid k$ into equal length blocks with sums σ_i^r that are equal for all i , $i = 1, \dots, a_r$. This implies, however, that $n = a_r \sigma_i^r$ for any i and thus $a_r \mid n$. Since

$\gcd(k, n) = g > 1$, $\mathcal{S}$ is periodic, and thus Algorithm 5 correctly stops the MAs in step 8.

Algorithm 5 must also guarantee that rendezvous occurs when possible, i.e., when $\mathcal{S}$ is aperiodic. Suppose not, i.e., $\mathcal{S}$ is aperiodic but rendezvous does not occur. This implies that for all $p_i \leq p_r$, the sequences calculated in step 7 of Algorithm 5 are periodic and thus all rounds of the algorithm will be executed. In the final round, where $p_i = p_r$, Algorithm 5 will stop the MAs and indicate that rendezvous is impossible. By the Chinese Remainder Theorem, however, this implies that $\mathcal{S}$ is periodic and thus contradicts the fact that $\mathcal{S}$ is aperiodic. Thus Algorithm 5 solves the mobile agent rendezvous problem.

If each mobile agent has memory $O(k \log \log n)$, then Algorithm 5 correctly determines whether rendezvous is possible and instructs the MAs to stop or rendezvous as appropriate. In the worst case, rendezvous is impossible and the MAs must complete all r rounds of Algorithm 5, where r is the smallest number of prime numbers such that $\prod_{i=1}^r p_i > n$. Each of the r rounds takes n steps so the time complexity is $O(rn)$. Earlier we proved that $r \in O(\frac{\log n}{\log \log n})$, so the time complexity of Algorithm 5 is $O(\frac{n \log n}{\log \log n})$. This completes the proof of Theorem 4.5. ■

4.5 Lower Bound on Memory

In Theorem 4.1, we prove that the mobile agent rendezvous problem can only be solved in the deterministic case if either k or n is known. When $k = 2$, we earlier

proved that $\Omega(\log \log n)$ memory is required to solve the rendezvous problem when k is known.

Theorem 4.6 *Solving the mobile agent rendezvous problem for $k > 2$ MAs requires $\Omega(\log \log n + \log k)$ memory.*

4.6 Conditional Solutions

In the preceding two sections, we proved that the mobile agent rendezvous problem is unconditionally solvable if each MA has memory $\Omega(\log \log n + \log k)$. In this section, however, we explore conditional solutions, i.e., solutions for cases where k and n satisfy various conditions such as primality. A network designer or administrator may be able to choose the values of k and n so as to meet these conditions.

The mobile agent rendezvous problem can be solved correctly whenever the sequences $\mathcal{S}_i$ are aperiodic for all $i \leq k$, e.g., n is prime or n is the product of two or more primes larger than k . A network designer or administrator, however, probably cannot directly dictate that $\mathcal{S}_i$ is aperiodic for all i but they are able to choose k and n . If $\gcd(k', n) = 1$, for all $k' \leq k$, e.g., n is prime or is the product of two primes greater than k , then $\mathcal{S}_i$ is aperiodic for all $i \leq k$.

The following algorithm assumes that the MAs know k , they share a common orientation, and $\gcd(k', n) = 1$ for all $k' \leq k$. An *active* token in unoccupied while an *inactive* token has a MA residing on it.

Algorithm 6

1. Release the token at the starting node.
2. Set $active = 1$.
3. Set $count = 0$.
4. Begin to walk around the ring in the clockwise direction.
5. Compute the intertoken distances to the next three active tokens, i.e., d_1, d_2, d_3 , and increment $count$ for each inactive token passed.
6. If $count == k - 1$, arrange rendezvous. (Only active MA remaining.)
7. If $d_2 > d_1$ and $d_2 \geq d_3$, then remain active.
8. Else become inactive, i.e., set $active = 0$, continue in current direction to starting node, and wait for further instructions.
9. Repeat from step 3.

Theorem 4.7 *When the MAs have $O(\log n)$ memory, share a common orientation, and $\gcd(k', n) = 1$, for all $k' \leq k$, then the mobile agent rendezvous search problem can be solved in time $O(n)$.*

Proof of Theorem 4.7.

First we prove that Algorithm 6 solves the rendezvous problem. If, for all $i \leq k$, $\mathcal{S}_i$ is aperiodic, then at least one MA will become inactive and at least one MA will remain active for each iteration of step 7. Thus after at most $\log k$ rounds, one MA will remain active and can arrange the rendezvous. If $\mathcal{S}_i$ is periodic for some i , however, then there is a round in which all MAs will become inactive and

so rendezvous is impossible. Thus Algorithm 6 solves the mobile agent rendezvous problem.

To prove that the time complexity of Algorithm 6 is $O(n)$ rather than $O(kn)$, we show that in each round unique ids can be assigned to the MAs such that the MA whose assigned id equals the intertoken distance at the beginning of the lexicographically maximum rotation of $\mathcal{S}$ always remains active. This implies that an instance of the execution of Algorithm 6 can be mapped to an instance of Peterson's leader election algorithm with unique ids and thus Algorithm 6 requires time $O(n)$.

To prove that the required mapping exists, we use induction on the number of rounds of Algorithm 6 and induction on the number of active MAs at the beginning of a round.

Base case (last round): Let z denote the number of the last round of Algorithm 6. Let m_z denote the number of MAs that are active at the beginning of round z . Let $\mathcal{S}_z$ denote the sequence of intertoken distances for the m_z MAs that are active at the beginning of round z . Let MA_{max} denote the MA at the start of the lexicographically maximum rotation of $\mathcal{S}_z$.

We prove via induction that unique ids can be assigned to the m_z MAs that are active at the beginning of round z such that the MA at the beginning of the lexicographically maximum rotation of $\mathcal{S}_z$ has the maximum id and remains active at the end of Algorithm 6 and is elected under Peterson's leader election algorithm.

Base case ($m_z = 2$): With $m_z = 2$ active MAs , there are only two intertoken dis-

tances, d_1, d_2 . Since $\mathcal{S}_z$ is aperiodic, $d_1 \neq d_2$. Without loss of generality, let $d_1 > d_2$. Assign $id = d_1$ to the MA at the head of the intertoken segment with distance d_1 and assign $id = d_2$ to the other MA .

Inductive hypothesis ($m_z = q$): Assume that unique ids can be assigned to the $m_z = q$ active MAs at the beginning of round z such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_z$ remains active.

$m_z = q + 1$: Let MA' denote an MA that does not occupy the node at the beginning of the lexicographically maximum rotation of $\mathcal{S}_z$ and whose distance to the next token, d' , can be decreased by one without affecting the MA at the start of the lexicographically maximum rotation of $\mathcal{S}_z$. Consider all the MAs except for MA' . The inductive hypothesis implies that these q MAs can be assigned unique ids such that the MA at the start of the lexicographically maximum rotation of $\mathcal{S}_z$ remains active. If we double these ids and assign $id = d'$ to MA' , then the last active MA in Algorithm 6 will be exactly the MA that would be elected by the Peterson leader election algorithm with unique ids.

Inductive hypothesis (rounds between the first and last rounds): Assume that for any round between the first and the last round of Algorithm 6, unique ids can be assigned to the active MAs at the beginning of the round such that the last active MA in Algorithm 6 will be exactly the MA that elected by the Peterson leader election algorithm with unique ids.

First round of Algorithm 6: The MAs that are active at the beginning of

round 2 are exactly those MAs that are active at the end of round 1. Thus the inductive hypothesis implies that the MAs that are active at the end of round 1 can be assigned unique ids and this includes the MA whose assigned id equals the intertoken distance at the beginning of the lexicographically maximum sequence of $\mathcal{S}_1$, the sequence of intertoken distances for round 1. Given these ids, we need to assign additional unique ids to the MAs that become inactive at the end of round 1. Let ID_{active} represent the set of unique ids for the MAs that remain active at the end of round 1.

Let m_1 denote the number of MAs that become inactive at the end of round 1. We prove via induction that we can assign unique ids to the m_1 MAs that are inactive at the end of round 1 such that the MA at the beginning of the lexicographically maximum rotation of $\mathcal{S}_1$ has the maximum id and thus remains active at the end of round 1 and is elected under Peterson's leader election algorithm.

Base case ($m_1 = 1$): If $\min(ID_{active}) > 1$, we can assign $id = \min(ID_{active}) - 1$ to the MA that becomes inactive at the end of round 1. Otherwise, increase each id in ID_{active} by one and assign $id = 1$ to the MA that becomes inactive at the end of round 1.

Inductive hypothesis: Assume that unique ids can be assigned to the $m_1 = q$ MAs that become inactive at the end of round 1 such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_1$ has the maximum id.

$m_1 = q + 1$: Let MA' denote an MA that becomes inactive at the end of round 1 and

whose distance to the next token, d' , can be decreased by one without affecting which MA starts the lexicographically maximum rotation of $\mathcal{S}_1$. Consider the MAs that become inactive after round 1 excluding MA' . The inductive hypothesis implies that these q MAs can be assigned unique ids such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_1$ has the maximum id and remains active at the end of round 1. If we double the ids assigned to all MAs , active and inactive excluding MA' , and assign $id = d'$ to MA' , then the MA with the maximum assigned id in round 1 will remain active at the end of round 1 in Algorithm 6 and will be elected in the Peterson leader election algorithm with unique ids.

Thus unique ids can be assigned to all MAs at the start of Algorithm 6 such that the execution of Algorithm 6 can be mapped to an execution of the Peterson leader election algorithm with unique ids. As a result, Algorithm 6 has the same time complexity as the Peterson leader election algorithm with unique ids, i.e., $O(n)$. This completes the proof of Theorem 4.7. ■

When k is prime, the ring is oriented, and $\gcd(k', n) = 1$, for all $k' \leq k$, then a variation of Algorithm 6 solves the mobile agent rendezvous problem with $O(\log k)$ memory and $O(n)$ time.

Algorithm 7

1. Release the token at the starting node.
2. Set active = 1.

3. Begin to walk around the ring in a clockwise direction.
4. Execute round 1 of Algorithm 6 but calculate the intertoken distances mod k . (All MAs will return to their starting nodes and those that became inactive have set $active = 0$.)
5. Execute Algorithm 6 as if on a ring of size k . The distances of interest are now the number of inhabited tokens between pairs of empty tokens.
6. Compute the number of inhabited tokens, i.e., tokens hosting inactive MAs , met on the path to the next three uninhabited tokens, i.e., m_1, m_2, m_3 .
7. If $m_1 == k - 1$, arrange the rendezvous. (Only one active MA left.)
8. If $m_2 > m_1$ and $m_2 \geq m_3$, then remain active.
9. Else become inactive, i.e., set $active = 0$, continue in current direction to starting node, and wait for further instructions.
10. Repeat from step 5.

Theorem 4.8 *When the MAs have $O(\log k)$ memory, share a common orientation, k is prime, and $\gcd(k', n) = 1$, for all $k' \leq k$, then the mobile agent rendezvous problem can be solved in time $O(n)$.*

Proof of Theorem 4.8. The execution of round one takes n steps. The remaining steps are as those in Algorithm 6 except that instead of counting nodes, the MAs count tokens. As in Algorithm 6, those remaining steps are executed in $O(n)$ time. Memory $O(\log k)$ is sufficient since an MA need only keep track of three distances at

a time. This completes the proof of Theorem 4.8. ■

The following algorithm solves the rendezvous problem when $\gcd(k', n) = 1$, for all $k' \leq k$, but k is not prime.

Algorithm 8

1. Release the token at the starting node.
2. Set $active = 1$ and $count = 0$.
3. Begin to walk around the ring in the clockwise direction.
4. Compute the intertoken distances mod k to the next three active tokens, i.e., $d_1, d_2, d_3 \bmod k$, and increment $count$ for each inactive token passed.
5. If $count == k - 1$, arrange rendezvous. (Only active MA remaining.)
6. If $d_2 > d_1 \bmod k$ and $d_2 \geq d_3 \bmod k$, then remain active.
7. Else become inactive, i.e., set $active = 0$, and wait for further instructions.
8. Repeat from step 4.

Theorem 4.9 *When the MAs have $O(\log k)$ memory, share a common orientation, and $\gcd(k', n) = 1$, for all $k' \leq k$, the mobile agent rendezvous search problem can be solved in time $O(n \log k)$.*

Proof of Theorem 4.9.

Since $\gcd(k', n) = 1$, for all $k' \leq k$, rendezvous is possible but the MAs do not have enough memory to identify the lexicographically maximum rotation of the sequence

of intertoken distances in a single traversal of the ring.

In each execution of steps 4 to 7 of Algorithm 8, at least one-third of the active MAs become inactive. Since $\gcd(k', n) = 1$, for all $k' \leq k$, the sequence of intertoken distances in each round is aperiodic so that rendezvous remains possible as the number of active MAs decreases. After at most $\log k$ rounds, the only MA that is still active is the one at the start of the lexicographically maximum rotation of the sequence of intertoken distances. Each round takes time n , so Algorithm 8 ensures a rendezvous in time $O(n \log k)$. This completes the proof of Theorem 4.9. ■

4.7 Leader Election and Rendezvous

The relationship between the rendezvous problem and the leader election problem among the k MAs depends on whether the MAs share a common orientation.

Theorem 4.10 *If the MAs share a common orientation, then the leader election problem among k MAs is equivalent to the rendezvous problem for those MAs . If the MAs do not share a common orientation, however, then the leader election problem is strictly more complex than the rendezvous problem.*

Proof of Theorem 4.10. If the leader election problem can be solved for the model described in section 4.1, then the elected MA can travel around the ring and instruct all other MAs to meet at a particular node. Thus a solution to the leader election problem for MAs always implies that there is a solution for the corresponding mobile agent rendezvous problem.

However, a solution to the mobile agent rendezvous problem does not necessarily imply a solution to the leader election problem. Consider the case where $k = 2$ *MA*s reside on $n = 3$ node ring, k is known but the *MA*s do not share a common orientation. If the rendezvous algorithm instructs the *MA*s to:

1. Choose a direction and begin to walk.
2. Walk until either:
 - (a) the other *MA* is met at a node, i.e., rendezvous occurs, or
 - (b) a token is discovered.
3. If a token is discovered, change direction and continue walking until rendezvous occurs.

If the *MA*s choose opposite directions initially, they will rendezvous on the node that has no token in at most two steps yet, given the symmetry of the problem, one *MA* cannot be distinguished from the other. Their memory contents are identical since they executed exactly the same steps of the algorithm and were on comparable nodes at the same time. Thus when the *MA*s do not share a common orientation, the leader election problem for *MA*s is strictly more difficult than the corresponding rendezvous problem.

If the *MA*s do share a common orientation, however, then a solution to the rendezvous problem always implies a solution to the corresponding leader election problem. In Algorithm 3 of section 4.4, the last *MA* that remains active organizes the

rendezvous and that MA can be elected the leader. This completes the proof of Theorem 4.10. ■

Chapter 5

Mobile Agent Rendezvous when Tokens Fail

5.1 The Network Model

The network model of Chapter 4 is extended to allow for token failure, i.e., when a token is no longer visible to any *MA*. For example, a token could fail if it is placed on a hostile node or if a malicious *MA* is able to flip the bit that represents the token. Tokens may fail either upon release or later in a given algorithm.

5.2 Overview of Chapter 5

Three cases of token failure are investigated. First, we assume tokens can fail only upon release. We prove that if the *MA*s have $O(k \log n)$ memory, $\gcd(k', n) = 1$ for

all $k' \leq k$, and at most $k - 1$ tokens fail, then the mobile agent rendezvous problem can be solved in time $O(kn)$.

Second, we assume tokens can fail at anytime. We prove that if the *MA*s have $O(k \log n)$ memory, $\gcd(k', n) = 1$ for all $k' \leq k$, and at most $k - 1$ tokens fail, then the mobile agent rendezvous problem can be solved in time $O(k^2n)$.

Next, we assume tokens can fail at anytime, $\gcd(k', n) = 1$ for all $k' \leq k$, and the *MA*s know n , the number of nodes in the ring. We prove that if the *MA*s have $O(k \log n)$ memory, then the mobile agent rendezvous problem can be solved in time $O(kn)$.

We conclude by comparing the time and memory requirements of rendezvous when tokens can fail to those when tokens cannot fail.

5.3 The Feasibility of Rendezvous

If all tokens fail or if, as shown in the previous chapter, k , the number of tokens, is such that $\gcd(k', n) \neq 1$ for some $k' \leq k$, then rendezvous is unsolvable in the given model.

5.4 Rendezvous When Tokens Fail Upon Release

First, we assume that a token can fail only upon release, i.e., in the first step of a given algorithm. The *MA* that released the token is not aware that it failed. The

MA s know k so, with adequate memory, they can walk around the ring and calculate the sequence $S = d_1, \dots, d_{3k}$, i.e., the sequence of the first $3k$ intertoken distances. Consider S^R , which is the reverse of S . Since the tokens fail only upon release, S^R can be partitioned as follows:

$$S^R = Q^q + d_1, \dots, d_\gamma \quad (5.1)$$

where Q^q is the concatenation of q copies of a unique aperiodic subsequence Q , $+$ is the concatenation operator, and $d_1, \dots, d_\gamma$ is a subsequence such that $\gamma < |Q|$. Upon identifying the subsequence Q , the MA s can identify a unique node upon which to rendezvous.

Algorithm 9

1. Release the token at the starting node.
2. Choose a direction and start walking.
3. Compute the sequence of $3k$ intertoken distances i.e., $S = d_1, d_2, \dots, d_{3k}$.
4. Let S^R be the reverse of S .
5. Find the shortest aperiodic subsequence Q that starts with the first element of S^R and is repeated such that $S^R = Q^q + d_1, \dots, d_\gamma$ where $\gamma < |Q|$.
6. Let Q^R be the reverse of Q .
7. Let $lexi(someSequence)$ denote the lexicographically maximum rotation of $someSequence$.

8. Set $forward = lexi(Q)$ and $reverse = lexi(Q^R)$.
9. If $forward$ and $reverse$ differ, then determine which of these sequences is the lexicographic maximum and rendezvous at the node where this sequence begins.
10. Else let MA_i and MA_j denote the MAs at the beginning of $forward$ and $reverse$ respectively.
11. If MA_i and MA_j are the same MA , then rendezvous at the node where MA_i resides.
12. If MA_i and MA_j are distinct MAs , then look at the two paths between MA_i and MA_j in the ring.
 - i) If only one of the paths had an odd number of nodes, then rendezvous at the node in the midpoint of that path.
 - ii) If both paths have an odd number of nodes, then
 - a) if the paths differ in length, rendezvous at the midpoint of the shorter path,
 - b) else compare the sequences of intertoken distances for the two paths and rendezvous at the node in the midpoint of the path that is the lexicographic maximum.
 - iii) If both paths have an even number of nodes, then rendezvous at the node in the midpoint of the path that contains an odd number of MAs .

Theorem 5.1 *When the MAs have $O(k \log n)$ memory, $\gcd(k', n) = 1$ holds for all $k' \leq k$, $f \leq (k - 1)$ tokens fail, and tokens can fail only upon release, then the mobile agent rendezvous problem can be solved in $O(kn)$ time.*

Proof of Theorem 5.1.

Let a be the number of tokens that do not fail, i.e., $a = k - f$. Let $A = d_1, \dots, d_a$ be the sequence of intertoken distances that exist after the f tokens have failed such that $\sum_{i=1}^a d_i = n$. Let S be the sequence of $3k$ intertoken distances calculated by a given MA in step 3 of Algorithm 9. Let S^R be the reverse of S . With a renumbering of the intertoken distances in S ,

$$S_R = A^\alpha + d_1, \dots, d_\gamma = (d_1, \dots, d_a)^\alpha + d_1, \dots, d_\gamma. \quad (5.2)$$

where $\gamma < a$. Thus there exists at least one aperiodic subsequence, namely A , that satisfies equation 5.1. If A is the shortest subsequence that satisfies equation 5.1, the MAs discover A in step 5 of Algorithm 9. Otherwise, the MAs discover a shorter aperiodic subsequence, Q , that satisfies equation 5.1.

The subsequence discovered in step 5 of Algorithm 9 is unique. If the shortest subsequence has z elements, these elements are the first z elements of S^R . Any other subsequence of the same length that satisfies equation 5.1 is also comprised of the first z elements of S^R and thus the subsequence discovered in step 5 is unique. This implies that all the MAs identify the same rendezvous node in the remaining steps of Algorithm 9 and rendezvous occurs.

Calculating S , the sequence of $3k$ intertoken distances, requires $O(k \log n)$ memory and requires $O(kn)$ time. Identifying the appropriate subsequence in step 5, determining the rendezvous node, and walking to the rendezvous node can be done in $O(kn)$ time as well, so the overall time requirement is $O(kn)$. This completes the proof of Theorem 5.1. ■

5.5 Rendezvous When Tokens Fail After Release And the Ring Size is Unknown

Suppose that token failures occur after release. In the following algorithm, if more than one but fewer than k MA s meet on a given node, then a partial rendezvous occurs, i.e., the MA s *merge* and act as one MA for the remainder of the algorithm.

Algorithm 10

1. Release token.
2. Set $r = 0$, where r denotes a round of the algorithm.
3. Choose a direction and begin walking.
4. Upon meeting another MA , merge with that MA .
5. Calculate the first $k - r$ intertoken distances, i.e., $S = (d_1, \dots, d_{k-r})$.
6. Estimate n as $\hat{n} = \sum_{i=1}^{k-r} d_i$.
7. Calculate S_{LMR} , the lexicographically maximum rotation of S .
8. Set $h = 0$.

9. Walk to the node that starts S_{LMR} and increment h for each node travelled.
10. Wait $2\hat{n} - h$ clock ticks.
11. If there are k MAs or their merged equivalent on the current node, stop.
- /* Rendezvous has occurred. */
12. Else if there are $1 < v < k$ MAs on the current node, then merge.
13. Set $r = r + 1$ and repeat from step 3.

The following three lemmata are used in the proof of Theorem 5.2.

Lemma 5.1 *Given a mobile agent MA^* and $0 \leq r \leq f - 1$, all other MAs will either finish round r or merge with MA^* by the time MA^* finishes step 5 of round $r + 1$ in Algorithm 10.*

Proof of Lemma 5.1.

Base case: $r = 0$.

Let MA_i be an arbitrary MA , other than MA^* , that finishes round $r = 0$ after MA^* does so. Let $\hat{n}$ and $\tilde{n}$ denote the estimates for n calculated by MA^* and MA_i respectively in round $r = 0$. The two MAs will meet and merge if:

- *Case 1:* MA^* 's wait in step 10 of round $r = 0$ overlaps MA_i 's walk in step 5 of round $r = 0$ by at least n steps, or
- *Case 2:* MA_i 's wait in step 10 of round $r = 0$ overlaps MA^* 's walk in step 5 of round $r = 1$ by at least n steps.

If the two MA s do not meet and merge in round $r = 0$, then

- *Case 3*: MA_i must finish round $r = 0$ on or before the time that MA^* finishes step 5 of round $r = 1$.

In round $r = 0$ of Algorithm 10, MA^* begins waiting no later than $\frac{3}{2}\hat{n}$, thus *Case 1* requires that

$$\frac{3}{2}\hat{n} + n \leq \tilde{n}. \quad (5.3)$$

Suppose that *Case 1* does not hold. *Case 3* requires that

$$3\tilde{n} \leq 3\hat{n} + n^* \quad (5.4)$$

where n^* is the estimate for n calculated by MA^* in step 5 of round $r = 1$.

Suppose that *Case 3* does not hold either. The final case, *Case 2*, requires that

$$\frac{3}{2}\tilde{n} \leq 3\hat{n}. \quad (5.5)$$

Since *Case 3* does not hold, equation 5.4 implies that

$$3\hat{n} + n^* < 3\tilde{n} \quad (5.6)$$

and thus MA_i waits in round $r = 0$ while MA^* walks at least n steps in step 5 of round $r = 1$ and the two MA s subsequently meet and merge.

Suppose that, like the previous cases, *Case 2* does not hold. Since equation 5.5 does not hold, then

$$3\hat{n} < \frac{3}{2}\tilde{n}. \quad (5.7)$$

This implies, however, that $2\hat{n} < \tilde{n}$ and thus contradicts the fact that *Case 1* does not hold. Therefore one of the cases must hold and thus, on or before MA^* finishes step 5 of round $r = 1$, either MA^* and MA_i meet and merge or MA_i finishes round $r = 1$.

Inductive Hypothesis: The lemma holds for $r = q$.

$r = q + 1$:

The inductive hypothesis implies that in round $r = q$, the two MAs either meet and merge, or MA_i finishes round $r = q$ on or before MA^* finishes step 5 of round $r = q + 1$. MA_i only starts round $r = q + 1$ if the two MAs did not meet in round $r = q$, i.e., MA_i finished round $r = q$ on or before MA^* finished step 5 of round $r = q + 1$.

Let t_0 and t_1 denote the respective times that MA^* and MA_i begin round $r = q + 1$ and let $\delta = t_0 - t_1$. The inductive hypothesis implies that $\delta > 0$ and that MA_i begins round $r = q + 1$ no later than $t_0 + \hat{n}$, where $\hat{n}$ is the estimate for n calculated by MA^* in round $r = q + 1$.

The two MAs will meet and merge in round $r = q + 1$ if

- *Case 1'*: MA^* 's wait in step 10 of round $r = q + 1$ overlaps MA_i 's walk in step

5 of round $r = q + 1$ by at least n steps, or

- *Case 2'*: MA_i 's wait in step 10 of round $r = q + 1$ overlaps MA^* 's walk in step 5 of round $r = q + 2$ by at least n steps.

If the two MAs do not meet and merge in round $r = q + 1$, then

- *Case 3'*: MA_i must finish round $r = q + 1$ on or before the time that MA^* finishes step 5 of round $r = q + 2$.

Case 1' occurs if

$$t_0 + \frac{3}{2}\hat{n} + n \leq t_0 + \delta + \tilde{n} \quad (5.8)$$

where $\hat{n}$ and $\tilde{n}$ are MA^* and MA_i 's respective estimates for n in round $r = q + 1$.

Suppose that *Case 1'* does not hold. *Case 3'* occurs if MA_i finishes round $r = q + 1$ on or before MA^* finishes step 5 of round $r = q + 2$, i.e.,

$$t_0 + \delta + 3\tilde{n} \leq t_0 + 3\hat{n} + n^* \quad (5.9)$$

where n^* is MA^* 's estimate for n in round $r = q + 2$.

Suppose that *Case 3'* does not hold. *Case 2'* occurs if

$$\delta + \frac{3}{2}\tilde{n} \leq 3\hat{n}. \quad (5.10)$$

However, since *Case 3'* does not hold, then

$$3\hat{n} < 3\hat{n} + n^* < \delta + 3\tilde{n}. \quad (5.11)$$

and thus MA_i waits in round $r = q + 1$ while MA^* walks at least n nodes in step 5 of round $r = q + 2$. This contradicts the fact that *Case 1'* did not occur. Thus one of three cases must occur when MA_i is in round $r = q + 1$. This ends the proof of Lemma 5.1. ■

Lemma 5.2 *The MAs that see the same sequence, up to a rotation, of intertoken distances S in a given round will rendezvous at the end of that round.*

Proof of Lemma 5.2.

The MAs that see the same S , up to a rotation, will have the same estimate for $\hat{n}$, and will identify the same rendezvous node. Let t_0 denote the time that the first MA finishes calculating S . Lemma 5.1 implies that the remaining MAs that see rotations of S will start calculating those rotations no later than t_0 . The first MA will wait at the rendezvous node from no later than $t_0 + \frac{\hat{n}}{2}$ until exactly $t_0 + 2\hat{n}$. The remaining MAs that see rotations of S arrive at the rendezvous node no later than $t_0 + \frac{3\hat{n}}{2}$, and thus all MAs that saw rotations of S in a given round will rendezvous by the end of that round. This completes the proof of Lemma 5.2. ■

Lemma 5.3 *A mobile agent MA_i will not execute round $r = f$ if fewer than f tokens have failed by the time that all MAs finishes step 5 of round $f - 1$.*

Proof of Lemma 5.3.

Base case: $f = 1$. If no tokens fail before all MA s, including MA_i , complete step 5 of round $r = 0$ then the MA s see, up to a rotation, the same sequence of intertoken distances S and thus identify the same rendezvous node. Since rendezvous occurs in round $r = 0$, round $r = 1$ is not executed.

Inductive hypothesis: The theorem is true for $f \leq q$.

Case $r = q + 1$:

If exactly q of $q + 1$ token failures have occurred, then the MA s are in rounds $r \leq q$. If all the MA s do not merge or rendezvous in one of the first $q - 1$ rounds, then some MA s execute round q . If no more tokens fail before these MA s finish step 5 of round q , then all the MA s in round q calculate, up to a rotation, the same sequence of intertoken distances, S , and thus rendezvous occurs in round $r = q$. If an additional token fails before all the MA s finish step 5 of round $r = q$, then the MA s may calculate sequences of intertoken distances that differ by more than a rotation and thus rendezvous may not occur. Those MA s that continue to round $r = q + 1$, however, all calculate the same sequence, up to a rotation, of intertoken distances and thus rendezvous occurs in round $r = q + 1$. This completes the proof of Lemma 5.3.

■

Theorem 5.2 *When the MA s have $O(k \log n)$ memory, $\gcd(k', n) = 1$ holds for all $k' \leq k$, at most $(k - 1)$ tokens fail, and token failures occur after release, then the mobile agent rendezvous problem can be solved in $O(k^2 n)$ time.*

Proof of Theorem 5.2.

Let $f \leq k - 1$ be the number of tokens that actually fail. Lemma 5.3 implies that no MA will execute more than f rounds of Algorithm 10. Suppose that rendezvous has not occurred by the end of round $r = f - 1$. Let MA^* denote the first MA that begins rounds $r = f$ and let t_0 denote the time when MA^* starts round $r = f$ of Algorithm 10. Since f tokens have failed, MA^* 's estimate for n will be correct, i.e., $\hat{n} = n$. The remaining MAs see the same sequence, up to a rotation, of intertoken distances as MA^* , and thus Lemma 5.2 implies that rendezvous occurs at the end of round f .

The number of failures, f , is at most $k - 1$ so at most $k - 1$ rounds of Algorithm 10 are executed. A round takes at most $k(n - 1)$ time, i.e., the product of the number of intertoken distances measured and the maximum intertoken distance possible. The resulting time required is $O(k^2n)$. Because at most k intertoken distances are calculated and the maximum intertoken distance is $(n - 1)$, the resulting memory complexity is $O(k \log n)$. This completes the proof of Theorem 5.2. ■

5.6 Rendezvous When Tokens Fail After Release and the Ring Size is Known

The following algorithm is useful when the MAs know not only k , the number of MAs , but also know n , the number of nodes in the ring. With adequate memory,

the MA s can walk around the ring and calculate the sequence $S = d_1, \dots, d_{3n}$ of $3n$ intertoken distances. If S^R , the reverse of S , begins with an aperiodic subsequence A that is repeated at least twice, the MA s can identify a unique node upon which to rendezvous. Otherwise, the MA s restart the algorithm. A MA that identifies a node for rendezvous would wait at that node until all MA s that could have seen the same view have arrived at that node. If the equivalent of k MA s arrived at the node, rendezvous would occur. Otherwise, the MA s restart the algorithm.

Algorithm 11

1. Release the token at the starting node.
2. Choose a direction and start walking.
3. Compute the sequence of intertoken distances for $3n$ steps, i.e., $S = d_1, d_2, \dots, d_x$ such that $\sum_{i \in S} d_i = n$.
4. Let S^R be the reverse of S .
5. Attempt to find an aperiodic subsequence A that starts at the first element of S^R and is repeated at least twice.
6. If no such subsequence exists, wait $n/2$ clock ticks and repeat from step 3.
7. Else let A^R be the reverse of A .
8. Let $lexi(someSequence)$ denote the lexicographically maximum rotation of $someSequence$.
9. Set $forward = lexi(A)$ and $reverse = lexi(A^R)$.

10. If *forward* and *reverse* differ, then determine which of these sequences is the lexicographic maximum and rendezvous at the node where this sequence starts.
11. Else let MA_i and MA_j denote the MA s at the beginning of *forward* and *reverse* respectively.
12. If MA_i and MA_j are the same MA , then rendezvous at the node where MA_i resides.
13. If MA_i and MA_j are distinct MA s, then look at the two paths between MA_i and MA_j in the ring.
 - i) If only one of the paths had an odd number of nodes, then rendezvous at the node in the midpoint of that path.
 - ii) If both paths have an odd number of nodes, then
 - a) if the paths differ in length, rendezvous at the midpoint of the shorter path,
 - b) else compare the sequences of intertoken distances for the two paths and rendezvous at the node in the midpoint of the path that is the lexicographic maximum.
 - iii) If both paths have an even number of nodes, then rendezvous at the node in the midpoint of the path that contains an odd number of MA s.
14. Wait $n/2$ minus the distance travelled in step 13.
15. If k MA s have arrived at the current node, rendezvous is complete.

16. Else repeat from step 3.

Theorem 5.3 *When the MAs have $O(k \log n)$ memory, know k and n , $\gcd(k', n) = 1$ for all $k' \leq k$, at most $k - 1$ failures occur, and tokens failures occur at anytime, then the mobile agent rendezvous problem can be solved in $O(kn)$ time.*

Proof of Theorem 5.2 Rendezvous fails when at least two MAs see different views during a given $3n$ node walk around the ring or when at least one MA does not see a repeated aperiodic sequence. Both of these situations arise when, in a given $3n$ step walk around the ring, at least one token fails. Since at most $k - 1$ tokens can fail, then at most $k - 1$ of the $3n$ step walks can be completed before all MAs see the same view. Thus rendezvous requires $O(kn)$ time. This completes the proof for Theorem 5.3. ■

5.7 The Cost of Token Failure

When tokens fail, the time and memory requirements of the mobile agent rendezvous problem increase. In Table 3, we compare the memory and time requirements for rendezvous with and without token failure.

In Chapter 4, we proved that when k is known and tokens cannot fail, the $k \geq 2$ mobile agent rendezvous problem can be solved with $O(k \log n)$ memory and $O(n)$ time. When tokens can fail upon release, however, the time required for rendezvous increases to $O(kn)$. If tokens can fail at anytime, the time required for rendezvous

<i>Knowledge</i>	<i>Memory</i>	<i>TokensFail</i>	<i>Time</i>
k	$O(k \log n)$	no	$O(n)$
	$O(k \log n)$	upon release	$O(kn)$
	$O(k \log n)$	anytime	$O(k^2n)$
k, n	$O(k \log n)$	anytime	$O(kn)$

Table 5.1: The Cost of Token Failure

increases $O(k^2n)$. However, when tokens can fail at anytime and the *MA*s know n , the time required for rendezvous can be held $O(kn)$.

Chapter 6

Mobile Agent Rendezvous with Randomization

Until now, only deterministic solutions to the mobile agent rendezvous search problem have been considered. In this chapter, however, we analyze randomized rendezvous algorithms so that we may compare their memory and time requirements with those of the previously analyzed deterministic algorithms. We also explore the time and memory requirements of rendezvous algorithms that use both randomization and tokens.

6.1 The Network Model

We return to the two *MA* model of chapter 3 but the *MA*s can run randomized as well as deterministic algorithms. The two *MA*s on a given ring are still restricted to

running the same algorithm, whether deterministic or randomized.

6.2 Overview of Chapter 6

First, we investigate the use of random walks to solve the mobile agent rendezvous search problem, i.e., in each step of a random walk algorithm, the *MA*s randomly choose the direction of travel and then walk a fixed number of nodes, m , in that direction.

We prove that if the *MA*s have memory $O(\log m)$ and move m nodes after randomly choosing a direction, then the mobile agent rendezvous problem can be solved in expected time $O(\frac{n^2}{m})$.

Next, we investigate how tokens and random walks can be combined to solve the mobile agent rendezvous problem. We prove that if the *MA*s know neither n nor d , have memory $O(1)$, and change direction with probability p whenever they count $0 \bmod 2$ tokens, then the mobile agent rendezvous problem can be solved in expected time $O(n)$.

We conclude by comparing the time and memory requirements of the randomized algorithms in this chapter with the corresponding deterministic algorithms of Chapter 3.

6.3 Rendezvous with Randomization

Assume that the two *MA*s, initially d nodes apart and with $O(\log m)$ memory, undertake the following random walk algorithm.

Algorithm 12

1. With probability $0 < p < 1$, move m nodes in the clockwise direction.
2. Else move m nodes in the counterclockwise direction.
3. Repeat from step 1 until rendezvous occurs.

Theorem 6.1 *If the two *MA*s have $O(\log m)$ bits of memory, then the mobile agent rendezvous search problem can be solved in expected time $O(\frac{n^2}{m})$.*

Before proceeding with the proof of Theorem 6.1, consider two mobile agents at distance d from each other and assume that $m = 1$. In each step of the algorithm, one of the following cases occurs. The mobile agents

1. move in the same direction with probability $p^2 + (1 - p)^2$,
2. move in opposite directions towards each other with probability $p(1 - p)$, or
3. move in opposite directions away from each other with probability $p(1 - p)$.

This implies that E_d , the expected number of rounds until rendezvous, is

$$E_d = 1 + (p^2 + (1 - p)^2)E_d + p(1 - p)E_{d+2} + p(1 - p)E_{d-2}. \quad (6.1)$$

It follows that for $d \geq 4$,

$$\begin{aligned} E_d &= \frac{1}{p(1-p)}((1-p^2 - (1-p)^2)E_{d-2} - p(1-p)E_{d-4} - 1) \\ &= 2E_{d-2} - E_{d-4} - \frac{1}{p(1-p)}. \end{aligned}$$

Given the initial condition $E_0 = 0$, it follows from equation 6.1 that

$$\begin{aligned} E_1 &= 1 + (p^2 + (1-p)^2)E_1 + p(1-p)E_3, \text{ and} \\ E_2 &= 1 + (p^2 + (1-p)^2)E_2 + p(1-p)E_4, \end{aligned}$$

such that

$$\begin{aligned} E_3 &= 2E_1 - \frac{1}{p(1-p)} \\ E_4 &= 2E_2 - \frac{1}{p(1-p)} \end{aligned}$$

Consider the following Lemma.

Lemma 6.1

$$\begin{aligned} E_{2k+1} &= E_1 - \frac{k+k^2}{2p(1-p)}, \text{ for } 2k+1 \leq n/2 \\ E_{2k} &= kE_2 - \frac{-k+k^2}{2p(1-p)}, \text{ for } 2k \leq n/2. \end{aligned}$$

Proof of Lemma 6.1

Consider the first expression. Using induction, it can be shown that there is a sequence

a_k such that

$$E_{2k+1} = E_1 - \frac{a_k}{p(1-p)}.$$

In fact,

$$\begin{aligned} E_{2k+1} &= 2E_{2k-1} - E_{2k-3} - \frac{1}{p(1-p)} \\ &= 2 \left(E_1 - \frac{a_{k-1}}{p(1-p)} \right) - \left(E_1 - \frac{a_{k-2}}{p(1-p)} \right) - \frac{1}{p(1-p)} \\ &= E_1 - \frac{2a_{k-1} - a_{k-2} + 1}{p(1-p)}, \end{aligned}$$

where a_k satisfies the recurrence relation:

$$a_k = 2a_{k-1} - a_{k-2} + 1$$

with initial conditions $a_0 = 0, a_1 = 1$. A special solution for this relation is $\frac{1}{2}k^2$. The characteristic polynomial is $x^2 = 2x + 1$ and a general solution of the homogeneous recurrence $a_k = 2a_{k-1} - a_{k-2}$ is $c_1 + c_2k$, for some constants c_1, c_2 . The general solution of $a_k = 2a_{k-1} - a_{k-2} + 1$ is then $a_k = c_1 + c_2k + \frac{1}{2}k^2$. Therefore the solution of the recurrence is $a_k = \frac{1}{2}k + \frac{1}{2}k^2$. This completes the proof of the first expression

Consider the second expression. Using induction, it can be shown that there exist sequences a_k, b_k such that

$$E_{2k} = a_k E_2 - \frac{b_k}{p(1-p)}.$$

In fact,

$$\begin{aligned}
E_{2k} &= 2E_{2k-2} - E_{2k-4} - \frac{1}{p(1-p)} \\
&= 2 \left(a_{k-1}E_2 - \frac{b_{k-1}}{p(1-p)} \right) - \left(a_{k-2}E_2 - \frac{b_{k-2}}{p(1-p)} \right) - \frac{1}{p(1-p)} \\
&= (2a_{k-1} - a_{k-2})E_2 - \frac{2b_{k-1} - b_{k-2} + 1}{p(1-p)},
\end{aligned}$$

which gives rise to the recurrences $a_k = 2a_{k-1} - a_{k-2}$ and $b_k = 2b_{k-1} - b_{k-2} + 1$ with initial conditions $a_0 = b_0 = 0, a_1 = 1, b_1 = 0$. Solving the recurrences as before results in $a_k = k$ and $b_k = -\frac{1}{2}k + \frac{1}{2}k^2$. This completes the proof of the second expression and thus completes the proof of Lemma 6.1. ■

Lemma 6.1 is used in the following proof of Theorem 6.1.

Proof of Theorem 6.1.

Assume that $m = 1$. If $d = n/2$ then equation 6.1 becomes

$$E_{\frac{n}{2}} = 1 + (p^2 + (1-p)^2)E_{\frac{n}{2}} + 2p(1-p)E_{\frac{n}{2}-2}$$

such that

$$E_{\frac{n}{2}} = E_{\frac{n}{2}-2} + \frac{1}{2p(1-p)}. \tag{6.2}$$

Lemma 6.1 then implies that

$$E_{n/2} = (n/4)E_2 - \frac{n/4(n/4-1)}{2p(1-p)}, \text{ and}$$

$$E_{n/2-2} = ((n/4 - 1)E_2 - \frac{(n/4 - 1)(n/4 - 2)}{2p(1 - p)}).$$

Substituting these expressions into equation 6.2 and solving for E_2 results, after simplifications, in

$$E_2 = \frac{n/2(n/2 - 2)}{8p(1 - p)} - \frac{(n/2 - 2)(n/2 - 4)}{8p(1 - p)} + \frac{1}{2p(1 - p)} = \frac{n/2 - 1}{2p(1 - p)}.$$

Finally, using the formula in part 2 of Lemma 6.1 results in

$$E_d = \frac{d}{2}E_2 - \frac{d(d - 2)}{8p(1 - p)} = \frac{d}{2} \cdot \frac{n/2 - 1}{2p(1 - p)} - \frac{d(d - 2)}{8p(1 - p)} = \frac{d(n - d)}{8p(1 - p)}, \quad (6.3)$$

i.e., the expected number of rounds until rendezvous when the MA s begin d nodes apart. With $m = 1$, a MA walks one node per round so the time per round is 1. As a result, the expected time until rendezvous is $\frac{d(n-d)}{8p(1-p)}$ which is $O(n^2)$. Therefore rendezvous occurs in expected time $O(n^2)$ when each mobile agent has memory $O(1)$.

Now consider the case where $m > 1$, each MA has memory $O(\log m)$ and can walk m nodes in each round of Algorithm 12. Replacing d with $\frac{d}{m}$ and n with $\frac{n}{m}$, equation 6.3 implies that the expected number of rounds is

$$\frac{\frac{d}{m} \left(\frac{n-d}{m} \right)}{8p(1 - p)}.$$

Since each round takes m time, rendezvous occurs in expected time $O(\frac{n^2}{m})$ when each agent has memory $O(\log m)$.

This completes the proof of Theorem 6.1. ■

6.4 Rendezvous with Randomization and Tokens

Having seen how the mobile agent rendezvous problem can be solved using randomization, it is worthwhile investigating how tokens and randomization can be combined to solve the problem.

Suppose that the two *MA*s have memory $O(1)$ and know neither n nor d .

Algorithm 13

1. Release the token.
2. Set $\text{count} = 0$.
3. Choose a direction and walk until a token is reached.
4. At the token, set $\text{count} = \text{count} + 1$.
5. If $\text{count} = 0 \bmod 2$, change direction with probability $0 \leq p \leq 1$.
6. Otherwise, maintain the same direction.
7. Walk to the next token.
8. Repeat from step 4 until rendezvous occurs.

Theorem 6.2 *When each *MA*s has memory $O(1)$, the mobile agent rendezvous problem can be solved in $O(n)$ time.*

Proof of Theorem 6.2. The expected number of rounds of Algorithm 13 is

$$\sum_{i=1}^{\infty} (2p(1-p))(1-2p(1-p))^{i-1} i = \frac{1}{(2p(1-p))}.$$

All but the final round take time n . Once the MAs are travelling in opposite directions on the ring, the expected time to rendezvous is $\frac{n}{2}$. Thus the total expected time to rendezvous is

$$n\left(\frac{1}{2p(1-p)} - 1\right) + \frac{n}{2} = \frac{n(1-p(1-p))}{2p(1-p)}$$

which is $O(n)$. The expected time is minimized when $p = \frac{1}{2}$. This completes the proof of Theorem 6.2. ■

6.5 The Performance of Deterministic and Randomized Algorithms

Theorem 6.1 proves that if the MAs have memory $O(\log n)$, then Algorithm 12, a randomized rendezvous algorithm, solves the two mobile agent rendezvous problem in expected time $O(n)$. Theorem 3.3 proves that there is a deterministic algorithm using tokens that requires memory $O(\log n)$ and time $O(n)$.

Theorem 6.2 proves that if the MAs are restricted to $O(1)$ memory, the combined use of randomization and tokens in Algorithm 13 solves the mobile agent rendezvous

problem in expected $O(n)$ time. Theorem 3.5, however, proves that none of the deterministic algorithms that use identical, stationary tokens can match this performance, since such algorithms have a lower bound on required memory of $O(\log \log n)$.

Chapter 7

Conclusion

Table 7.1 lists the time required for rendezvous when the MAs have $O(\log n)$ memory in the $k = 2$ case and $O(k \log n)$ memory in the $k > 2$ case. Unless stated otherwise, the MAs run one of the deterministic algorithms presented in this thesis. The table also lists the time required for rendezvous when two MAs have $O(1)$ memory and they run Algorithm 13, the randomized algorithm that uses tokens.

When $k = 2$, $O(\log n)$ memory allows the MAs to rendezvous in $O(n)$ time. If $k \geq 2$ and memory is increased to $O(k \log n)$, rendezvous can still occur in $O(n)$ time. However, when $k \geq 2$, tokens can fail upon release, and the MAs have $O(k \log n)$ memory, the rendezvous time can increase to $O(kn)$. If the tokens can fail at anytime, the rendezvous time may increase to $O(k^2n)$. Finally, when $k = 2$, the MAs have $O(1)$ memory, and the MAs run Algorithm 13 which uses randomization and tokens, rendezvous occurs in expected time $O(n)$.

Thus, given sufficient memory, the MAs can rendezvous in $O(n)$ time when they

<i>Knowledge</i>	<i>Memory</i>	<i>Time</i>	<i>Note</i>
$k = 2$	$O(\log n)$	$O(n)$	deterministic algorithm
$k \geq 2$	$O(k \log n)$	$O(kn)$	deterministic algorithm
$k \geq 2$	$O(k \log n)$	$O(k^2 n)$	deterministic algorithm and tokens fail
$k = 2$	$O(1)$	$O(kn)$	randomized algorithm using tokens

Table 7.1: A Comparison of Rendezvous Times

use unfailing tokens and the same deterministic algorithms. Furthermore, for the $k = 2$ case, the MAs can rendezvous in $O(n)$ expected time when they use unfailing tokens and the same randomized algorithms. When tokens can fail, however, rendezvous time can increase by at least $O(k)$ for $k \geq 2$.

While the rendezvous search problem has been studied extensively, research has focused on domains that are quite different from network domains, e.g., the studied domains were usually continuous and did not reflect the topologies of computer networks. Randomized algorithms were used to combat any symmetry that arose in the problem. In contrast, we investigated the rendezvous search problem on a common network topology, the ring, and used tokens to break symmetry so that the MAs could run the same deterministic algorithms. We proved an impossibility theorem for the two MA case and then presented algorithms, with various time and memory requirements, that solved the rendezvous search problem for two MAs. After doing the same for the $k \geq 2$ MA case, we studied how token failure affected the time and memory requirements for rendezvous. We then explored the use of a random walk algorithm for the two MA case. Finally, we presented an algorithm that uses both tokens and randomization to solve the rendezvous search problem.

Several interesting questions about mobile agent rendezvous in the ring remain unanswered. For example, how do the time and memory requirements for rendezvous change if a *MA* on a given node is able to see not only tokens and other *MA*s on the same node but also tokens and *MA*s on adjacent nodes? What happens when a *MA* that is residing on the same node as a token only sees that token with probability $0 < p < 1$? Since tokens can be used successfully to break symmetry in the ring, it would be interesting to study the use of tokens in other network topologies that exhibit symmetry, e.g., the torus and the hypercube. Does rendezvous in these higher dimension networks require more than one token per *MA*? Without doubt, the mobile agent rendezvous problem holds many interesting questions that require further study.

Bibliography

- [1] S. Alpern, The Rendezvous Search Problem, SIAM Journal of Control and Optimization, 33, pp. 673-683, 1995. (Earlier version: LSE CDAM Research Report, 53, 1993.)
- [2] S. Alpern, Asymmetric Rendezvous on the Circle, Dynamics and Control, 10, pp. 33-45, 2000.
- [3] S. Alpern, Rendezvous Search in One or More Dimensions, LSE CDAM Research Report, No. 05, 2001.
- [4] S. Alpern, Rendezvous Search on Labelled Networks, Naval Research Logistics, 49, pp. 256-274, 2002.
- [5] S. Alpern, Rendezvous Search: A Personal Perspective, Operations Research, 50, No. 5, pp. 772-795, 2002.
- [6] S. Alpern and V. Baston and S. Essegiaier, Rendezvous Search on a Graph, Journal of Applied Probability, 36, No.1, pp. 223-231, 1999.

- [7] S. Alpern and A. Beck, Rendezvous Search on the Line with Bounded Resources: Expected Time Minimization, *European Journal of Operational Research*, 101, pp. 588-597, 1997.
- [8] S. Alpern and A. Beck, Rendezvous Search on the Line with Limited Resources: Maximizing the Probability of Meeting, *Operational Research* 47, No. 6, pp. 849-861, 1999.
- [9] S. Alpern and A. Beck, Asymmetric Rendezvous on the Line is a Double Linear Search Problem, *Mathematics of Operations Research*, 24, No. 3, pp. 604-618, 1999.
- [10] S. Alpern and A. Beck, Pure Strategy Asymmetric Rendezvous on the Line with an Unknown Initial Distance, *Operations Research*, 48, No. 3, pp. 498-501, 2000.
- [11] S. Alpern and S. Gal, Rendezvous Search on the Line with Distinguishable Players, *SIAM Journal Control and Optimization*, 33, pp. 1270-1276, 1995.
- [12] S. Alpern and S. Gal, Searching for an Agent Who May or May Not Want to be Found, *Operations Research*, 50, No. 2, pp. 311-323, 2002.
- [13] S. Alpern and S. Gal, *The Theory of Search Games and Rendezvous*, Kluwer Academic Publishers, Norwell, Massachusetts, 2003.
- [14] S. Alpern and J.V. Howard, Alternating Search at Two Locations, *Dynamic Control*, 10, pp. 319-339, 2000.

- [15] S. Alpern and W.S. Lim, Rendezvous of Three Agents on the Line, *Naval Research Logistics*, 49, pp. 224-255, 2002.
- [16] S. Alpern and D. Reyniers, The Rendezvous and Coordinated Search Problems, *Proceedings of the 33rd Conference on Decision and Control*, No. 1, pp. 513-517, Lake Buena Vista, FL, December 1994.
- [17] E.J. Anderson and S. Essegaiier, Rendezvous Search on the Line with Indistinguishable Players, *SIAM Journal of Control and Optimization*, 33, pp. 1637-1642, 1995.
- [18] E.J. Anderson and S. Fekete, Two-dimensional Rendezvous Search, *Operations Research*, 49, pp. 107-188, 2001.
- [19] E.J. Anderson and R.R.Weber, The Rendezvous Problem on Discrete Locations, *Journal of Applied Probability*, 28, pp. 839-851, 1990.
- [20] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility, *I.E.E.E. Transactions on Robotics and Animation*, 15, No. 5, pp. 818-828, 1999.
- [21] T.M. Apostol, *Introduction to Analytical Number Theory*, Springer Verlag, 1997.
- [22] L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro, Election and Rendezvous of Anonymous Mobile Agents in Anonymous Networks with Sense of Direction, *Proceedings of the 9th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pp. 17-32, 2003.

- [23] L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro, Can We Elect If We Cannot Compare?, Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 324-332, 2003.
- [24] V. Baston, Two Rendezvous Search Problems on the Line, Naval Research Logistics, 46, pp. 335-340, 1999.
- [25] V. Baston and S. Gal, Rendezvous on the Line When the Players' Initial Distance is Given by an Unknown Probability Distribution, SIAM Journal of Control and Optimization, 36, No.6, pp. 1880-1889, 1998.
- [26] V. Baston and S. Gal, Rendezvous Search When Marks Are Left at the Starting Points, Naval Research Logistics, 47, No. 6, pp. 722-731, 2001.
- [27] S. Benkoski, M. Monticino, J. Weisinger, A Survey of the Search Theory Literature, Naval Research Logistics, 38, pp. 469-494, 1991.
- [28] H. Buhrman, M. Franklin, J. Garay, J. Tromp, and P. Vitanyi, Mutual Search, JACM, Vol. 46, No. 4, pp. 517-536, 1999.
- [29] E. Chester and R. Tutuncu, Rendezvous Search on Finite Domains, Preprint, Department of Mathematical Sciences, Carnegie Mellon University, 2001.
- [30] T. Cormen, C. Leiserson, and R. Rivest, Introduction to Algorithms, MIT Press, Cambridge, Massachusetts, 1992.

- [31] V.P. Crawford and H. Haller, Learning to Cooperate: Optimal Play in Repeated Coordination Games, *Econometrica*, 58, No. 3, pp. 571-596, 1990.
- [32] A. Dessmark, P. Fraigniaud, and A. Pelc, Deterministic Rendezvous in Graphs, 11th Annual European Symposium on Algorithms (ESA), pp. 184-195, 2003.
- [33] W. Feller, An Introduction to Probability Theory and its Applications, Vol. II, Wiley, 1966.
- [34] P. Flocchini, E. Kranakis, D. Krizanc, and F. Luccio, Sorting Multisets in Anonymous Rings, to appear, *Journal of Parallel and Distributed Computing*.
- [35] P. Flocchini, E. Kranakis, D. Krizanc, F. Luccio, N. Santoro, and C. Sawchuk, Mobile Agent Rendezvous with Token Failure, preprint, 2003.
- [36] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Multiple Mobile Agent Rendezvous in the Ring, to appear, *Proceedings of Latin America Theoretical Informatics (LATIN)*, 2004.
- [37] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots, *ISAAC 1999, LNCS 1741*, pp. 93-102, 1999.
- [38] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, Gathering of Asynchronous Oblivious Robots with Limited Visibility, *STACS 2001, LNCS 2010*, pp. 247-258, 2001.

- [39] S. Gal, Rendezvous Search on the Line, *Operations Research* (47), No. 6, pp. 974-976, 1999.
- [40] N. Hanusse, D. Kavvadias, E. Kranakis, D. Krizanc, Memoryless Search Algorithms in a Network with Faulty Advice, *Proceedings of IFIP International Conference on Theoretical Computer Science, (TCS)*, pp. 206-216, 2002.
- [41] J. Howard, Rendezvous Search on the Interval and Circle, *Operations Research*, 47, No. 4, pp. 550-558, 1999.
- [42] F. Kramarz, Dynamic focal points in N-person coordination games, *Theory and Decision*, 40, pp. 227-313, 1996.
- [43] E. Kranakis, D. Krizanc, L. Kirousis, and C. Sawchuk, Randomization and Mobile Agent Rendezvous in the Ring, preprint, 2003.
- [44] E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Mobile Agent Rendezvous Search Problem in the Ring, *International Conference on Distributed Computing Systems (ICDCS)*, pp. 592-599, 2003.
- [45] D. Knuth, *The Art of Computer Programming, Volume 1, Fundamental Algorithms*, 3rd edition, Addison Wesley, 1997.
- [46] W.S. Lim, Rendezvous Search on the Line with Three Players, LSE CDAM Research Report, LSE-MDS-81, 1997.

- [47] W.S. Lim, A Rendezvous-Evasion Game on Discrete Locations with Joint Randomization, *Advances in Applied Probability*, 29, pp. 1004-1017, 1997.
- [48] W.S. Lim and S. Alpern, Minimax Rendezvous Search on the Line, *Journal of Control and Optimization*, 34, pp. 1650-1665, 1996.
- [49] W.S. Lim and A. Beck and S. Alpern, Rendezvous Search on the Line with More Than Two Players, *Operations Research*, 45, pp. 357-364, 1997.
- [50] M. Pikounis and L.C. Thomas, Many Player Rendezvous Search: Stick Together or Split and Meet? *Naval Research Logistics*, 48, No. 8, pp. 710-721, 2001.
- [51] G. Prencipe and M. Cieliebak, Gathering Autonomous Mobile Robots, *Proceedings of 9th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pp. 57-72, 2002.
- [52] S. Ross, *Probability Models for Computer Scientists*, Academic Press, 2002.
- [53] N. Roy and G. Dudek, Collaborative Robot Exploration and Rendezvous: Algorithms, Performance Bounds, and Observations, *Autonomous Robots*, 11, pp. 117-136, 2001.
- [54] T. C. Schelling, *The Strategy of Conflict*, Harvard University Press, Cambridge, MA, 1960.
- [55] L.C. Thomas and P.B. Hulme, Searching for Targets Who Want to be Found, *Journal of the Operations Research Society*, 48, Issue 1, pp. 44-50, 1997.

- [56] X. Yu and M. Yung, Agent Rendezvous: A Dynamic Symmetry-Breaking Problem, in Proceedings of ICALP '96, LNCS 1099, pp. 610-621, 1996.