A UBIQUITOUS GIS: FRAMEWORK, SERVICES AND ALGORITHMS DEVELOPMENT

By
Jiehua Yi

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

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario
January 2009

© Copyright
2009, Jiehua Yi

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-52081-9
Our file Notre référence
ISBN: 978-0-494-52081-9

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

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

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent 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 thesis.

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

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

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

■ ■ ■
Canada

Abstract

In this thesis, we focus on the development of a GIS system, called Intelligent Map Agent system (IMA). We implemented the system framework, designed and implemented several system services, and designed efficient algorithms for the services.

The main focus of the IMA is to provide access to services, which require the usage of spatial data and spatial information, to mobile users regardless of their location. The IMA achieves this via Services Oriented Architectures [8] which supports a distributed set of services instead of a single monolithic spatial information system. The IMA system can accommodate a large number of mobile users and services, which may be distributed across a wide geographical area. The services, as well as users, are represented by agents using JADE/LEAP technologies. The first part of the thesis is dedicated to the IMA system including a description of its architecture design and implementation. In the second part, we present algorithmic solutions to two services, intended for the IMA system. These are: (i) meeting scheduling services and (ii) shortest path services. The services are using data having temporal and geometrical constraints.

We present an algorithmic solution to one variation of the meeting scheduler service, where each participant (traveling in Euclidean space) is associated with a meeting schedule, and the problem is to find a location and a time where all the participants can meet for the longest possible duration. For this problem, we present two algorithms, a geometric algorithm has an $O(n \log n)$ execution time (participants travel at the same speed) and a linear programming algorithm with an $O(n)$ expected time (participants may travel at different speeds). When no new meeting can

be scheduled among all the participants, we give an $O(n^4)$ algorithm to find the maximum number of participants. We also consider the case that all participants travel in a road network (graph).

We also present an algorithmic solution to one variation of the shortest path service, which is to compute a shortest path among a set of growing disc obstacles. For the case where the disc radii grow at the same constant speed, our algorithm runs in $O(n^2 \log n)$ time, which improves upon an $O(n^3 \log n)$ time solution. When the discs grow at different speeds, we are able to compute a shortest path in $O(n^3 \log n)$ time.

Acknowledgements

I am lucky that I have worked with several excellent researchers in GIS/computational geometry during my Ph.D. study in Carleton University. All of them are of great help to me in a way or another. I have benefited a lot from their wisdom and research experiences.

First of all, I would like to thank my two supervisors, Jörg-Rüdiger Sack and Doron Nussbaum, for their financial support without which I could not have completed the work presented here. Jörg-Rüdiger Sack is not only strict with researches, but also has a lot of influences in soft skills on the students. He stands high and shows great patience to the students. Doron Nussbaum would like to sit with students together, and work out the details. This is a great guide especially when I started to do the research.

One of the co-authors, Anil Maheshwari, I would like to express gratitude on his comments and questions in all of the drafts during constructing a paper. The “ability to ask the right questions and then listen quietly” working style gives me great joy and sense of achievement.

Michael Smid, with whom I show my great respect. I have benefited from his strong mathematics background since he was my course instructor.

My another two great co-authors from University of Bonn, German, Rolf Klein, and Florian Berger, have been excellent fellow researchers to work with.

Also I would like to thank Prosenjit Bose, Pat Morin and the world class computational geometry research group, for their great help to open my mind through those interesting research presentations.

I would like to thank the members on my examining board: Leila De Floriani, Paola Flocchini, Brett Stevens, Anil Maheshwari and Michel Gaulin. I do appreciate their willingness to be on the committee, and the time they spent in reading my thesis.

I have been always saying that finishing my Ph.D. study is the result of my whole family's contribution. I would like to express my upmost appreciation to my husband Yongzhong, and three years old son Darren. Without their endless love and ultimate support, I could not have this thesis done. When Darren was only one and a half month old, I went back to school, and Yongzhong has been taking care of him all the time since then. It has been a pain in my heart for Darren, as when he needs me mostly but I could not be on his side. He had not so much time to enjoy playing with his mom, and to be huddled by his mom, when he was an infant and a toddler. My parents and sisters also deserve my gratitude for their support and their love.

Table of Contents

Abstract	ii
Acknowledgements	iv
Table of Contents	vi
List of Figures	ix
1 Introduction	1
1.1 Motivation	1
1.2 Summary of thesis results	4
1.2.1 IMA system architecture design and implementation	4
1.2.2 IMA system services	6
1.3 Organization	7
2 Background	9
2.1 Related applied work	9
2.1.1 Agent technologies	9
2.1.2 Systems using agent technologies	10
2.1.3 Software architectures	12
2.2 Related geometric concepts and algorithms	14
2.2.1 Voronoi diagrams	14
2.2.2 Shortest paths among growing discs	34
2.2.3 3SUM-hard problems	37

3	IMA System	40
3.1	Intelligent map agents architecture	40
3.1.1	Architecture overview	40
3.1.2	System components	44
3.1.3	Interaction and collaboration	49
3.2	IMA implementation and applications	51
3.2.1	IMA implementation	51
3.2.2	Services	53
3.3	Conclusions and future directions	61
4	Algorithms on Meeting Scheduling	63
4.1	Introduction	63
4.2	Organization	65
4.3	General problem definition	66
4.4	Euclidean meeting scheduling	67
4.4.1	Geometric model	70
4.4.2	Linear programming model	82
4.5	Maximizing the number of participants in Euclidean space	92
4.6	Meeting scheduling in a transportation network	97
4.7	Conclusion	99
5	Shortest Paths among Growing Discs	100
5.1	Introduction	100
5.2	Preliminaries	102
5.2.1	Departure curves computation	109
5.2.2	Nearest-block disc problem	114
5.3	An overview of our approach	119
5.4	Valid tangent computation	122
5.5	Valid buffer computation for various-speed discs	126
5.5.1	Algorithms	129
5.5.2	Time complexity analysis	130

5.6	Valid buffer computation for same-speed discs	136
5.7	Detecting the validity of a spiral	144
5.8	Executing Dijkstra's shortest path	145
5.9	Conclusions	146
6	Summary and Future Work	154
A	Computing the function of a departure curve	158
A.1	Compute time range for a departure curve	158
A.1.1	The time range for a right-right departure curve	158
A.1.2	The time range for a right-left departure curve	162
A.2	Compute the formula for a departure curve in various-speed case . . .	163
A.3	Compute the formula for a departure curve with the same-speed discs	169
	Bibliography	171

List of Figures

2.1	(a): Voronoi diagram of a set of points (b) order-2 Voronoi diagram of the set of points (c) order-3 Voronoi diagram of the set of points (d) furthest Voronoi diagram of the set of points	14
2.2	When an edge e in (a) is flipped, the topology is changed to e' in (b).	18
2.3	(a): Voronoi diagram of a set of discs (b) furthest Voronoi diagram of a set of discs	19
2.4	An example of the bisector $J(C_i, C_j)$ with function $\ p - o_i\ - \frac{w_i}{w_j}\ p - o_j\ = w_j r_i - w_i r_j$, $p \in J(C_i, C_j)$. $\frac{w_i}{w_j} = 1.5$, $\ o_i - o_j\ = 10$, the numbers on the curves represent the possible values of $w_i r_j - w_j r_i$	21
2.5	Two cones intersect with a hyperbola in 3- d . Two discs are formed by cutting the cones with a horizontal plane. The projection of the hyperbola onto $\mathbb{R}^2$ is a Voronoi edge between the two discs.	22
2.6	H is a plane parallel to XY plane with height E . disc $C_i = H \cap \text{cone}_i$.	23
2.7	The dashed discs are the determining discs of the minimum spanning circle (MSC) g of the set of discs. (a): Two determining discs a and b . The center of a , b and g are on a line. (b) Three determining discs a , d , and e . (c): When the radii of the discs grow at the same speed, discs a , d and e intersect at a common point p , and all other discs include p .	24
2.8	(a) No minimal spanning circle of C_1 , C_2 and C_3 . (b) The minimum spanning circle C of three circles C_1 , C_2 and C_3	25

2.9	(a) : The intersection of three cones $cone_1$, $cone_2$ and $cone_3$. (b) : The discs obtained by cutting the cones at the time plane t . The intersection of the discs are three arcs a_1 , a_2 and a_3	28
2.10	The stabbing lines for a set of discs. The stabbing region is composed of bold discs. The bold discs determine the stabbing lines.	30
2.11	Cited figure in [1]. The furthest color Voronoi diagram for three colors and seven sites. The doubly connected unbounded area belongs to the site in the middle while the dark (resp. light) shaded areas belong to the closest dark (resp. light) sites (following the arrows).	32
2.12	(a): Weighted distance functions of sites p , q and s . The <i>WVD</i> on a line is equivalent to the projection of the lower envelope of the weighted distance functions. (b): The weighted Voronoi regions on the edges (edges are bold). They are determined by different sites. The dashed lines are the lower envelope of the weighted distance functions. (c): When s is processed, r , p and q have been processed as they have less weights. The distance function of s is blocked by p and q 's.	33
2.13	A shortest path from s to g among the set of growing discs. It is composed of two tangent segments and a spiral segment. At the time when the spiral reaches tangent point q , disc C becomes the dashed disc.	35
2.14	An example of GEOMBASE.	39
3.1	The IMA architecture is user-centric. As such the IMA is aimed at providing each user with personal set of functionality capabilities, in- teraction patterns and visualization.	41
3.2	The IMA consists of three main tiers: user tier, services tier and data resources tier. Although the IMA permits each tier to interact with any other tier, it is envisioned that users will access the resource tier via the services tier. Note, that services can act as users and thus access other services to accomplish their task(s).	43

3.3	The figure illustrates the main components of the IMA and gives a possible snapshot of the IMA during operation. The double ended arrows represent communication between components (a Service Access Point, a Service Worker, and a Resource Provider are represented by SAP, SW and RP respectively).	45
3.4	The figure shows an example of an invocation sequence of service and interaction between the components. The solid double ended arrows represent ongoing communication between components while dashed double ended arrows represent temporary invocation which occurs during invocation of the different services and applications (a Service Access Point, a Service Worker, and a Resource Provider are represented by SAP, SW and RP respectively).	46
3.5	The IMA can operate in a distributed environment. Although, users are handled by a single IMA cluster (depicted in Figure 3.3), they have access to all services via a federated network of IMA clusters. The figure shows an example of a three federated IMA clusters which are located across Canada (Halifax, Ottawa, and Vancouver). The three clusters are connected to each other via the IMA DFB. Note that the services registered at the various locations would typically be different.	50
3.6	The figure shows a set up of a meeting scheduling application. The set up mimics a real-life distributed users and services which are registered with the IMA clusters (locations are across Canada). The arrows between the components represent the registration and knowledge relationships between the components.	54
3.7	This snapshot shows the status of the users and services after the users and services have registered with the IMA system.	56
3.8	Linda, who is the meeting initiator, sets up the meeting subject, duration, location, date, and invites persons for the new meeting.	57

3.9	The figure presents an interaction diagram between the different components of the IMA. For brevity we omitted the interaction with the SAPs.	58
3.10	The figure presents the dynamic agents representing users and services that are produced during the processing of a service in the IMA system. After the service requests have been fulfilled, the dynamic agents will be deleted by the IMA.	59
3.11	Each user receives a map with the new meeting location and two shortest paths. The first path, shown in red, is the path from the current meeting to the new meeting location. The second path, shown in blue, is the path from the new meeting location to the next meeting location. (Note, the blue path travels along the river until it can cross the Rideau river at a bridge.)	60
4.1	(a): Two vertical cones $PRE_{1,2}$ of opening angle 90° with the projection of apex on the XY plane $l_{1,2}^{pre}$. $PRE = PRE_1 \cap PRE_2$. A vertical line through l intersects PRE_1 and PRE_2 . (b): Two vertical cones $SUB_{1,2}$ of opening angle 90° with the projection of apex on the XY plane $l_{1,2}^{sub}$. $SUB = SUB_1 \cap SUB_2$. (c) An example of three cones intersect. . .	69
4.2	Illustrating the proof of Lemma 4.1.	72
4.3	Shape S' expanding over the furthest site Voronoi diagram. In the Voronoi cell V_{C_5} of the disc C_5 , the arc on the boundary of S' belongs to the dashed disc expanded from disc C_5	74
4.4	(a): Touching point l moves from the Voronoi edge between PRE_i and PRE_j to the interior of the arc of PRE_i . The arc of PRE_i is tangent to the arc of SUB_k at l . The dotted curves are the Voronoi edges between the left (resp., right) two discs, respectively. The Voronoi edge between PRE_i and PRE_j intersects $l_i^{pre} l_k^{sub}$ in l . (b): A type 3 event.	77

- 4.5 (a) The trace of the touching points of S' and S'' . The new meeting point is located on a Voronoi edge. The left Voronoi diagram is V^{pre} , and the right Voronoi diagram is V^{sub} . (b) The trace starts from the earliest meeting point m' . (c) The trace along a Voronoi edge from V^{pre} . (d) The trace is determined by two circles, one from V^{pre} , and one from V^{sub} . (e) The trace along a Voronoi edge from V^{sub} , and ends at the latest meeting point m'' 80
- 4.6 Axiom 2 does not hold for the participants M_1, M_2, M_3 , if we use $x \mapsto f(x)$ as objective function. For the sets $Q := \{M_1\}$, $G := \{M_1, M_2\}$ and $h := M_3$, we obtain $w(Q) = w(G) = 1$ and $w(G \cup \{h\}) < w(G)$, but $1 = w(Q \cup \{h\}) = w(Q)$ 84
- 4.7 This is a 3-d figure. The four vertices are located on the convex body F intersected by the set of cones. Every vertical segment of maximum length contained in this parallelogram corresponds to a meeting of longest duration. 85
- 4.8 The longest meeting duration that is possible for the participants in Z is the same as for the participants in B . The dotted curves correspond to downward cones in $B \setminus Z$ 89
- 4.9 Set of 4 participants $\{M_1, M_2, M_3, M_4\}$ and their way to the optimum meeting location. The start of their optimum meeting is determined by participants M_1 and M_3 , while the end is determined by M_2 and M_4 . Removing any participant enables a meeting with longer duration. If one, for example, removes participant M_1 , the optimum meeting location becomes $(1, 1)$ and the duration of the optimum meeting increases by $\sqrt{2} - 1$ 90
- 4.10 The longest meeting duration that is possible for the participants in Z is the same as for the participants in B . The dotted curves d_1, d_2 correspond to downward upward cones in $B \setminus Z$, while the dotted curve u_1 corresponds to an upward cone. 91

5.1	Right-right, right-left, left-right, left-left tangents from disc $C_i(T_0)$ to disc C_j , which is originally $C_j(T_0)$	104
5.2	For the proof of Lemma 5.2. Disc C_i and C_j grow at the same speed. Tangent $l_{ij}^{rr}(T_0)$ and $l_{ij}^{rr}(t)$ with $t > T_0$	105
5.3	The Voronoi diagram of a set of discs growing at the same speed. $DC(l_{ij}^{rr})$, $AC(l_{ij}^{rr})$ and tangent $l_{ij}^{rr}(T_0)$. Tangent $l_{ij}^{rr}(T_p)$ intersects V_{C_i} with p . Tangent $l_{ij}^{rr}(T'_p)$ intersects V_{C_j} with point q . $p \in DC(l_{ij}^{rr})$, $p' \in DC(l_{ij}^{rr})$, $q \in AC(l_{ij}^{rr})$. The bold segment on $DC(l_{ij}^{rr})$ is the valid buffer on $DC(l_{ij}^{rr})$, determined by disc C_R	107
5.4	Tangent l from C_i to C_j . (a) C_k projects on l , and C_k and $C_{k'}$ are lower than C_j along l . (b) C_k and C_r are two right-blocks of l . C_f is the left-closest-block of l	108
5.5	Two discs C_i and C_j , with $ o_i o_j = s$. The speeds of C_i , C_j and the robot are $V_i (= 1)$, V_j , and V_R , respectively. The robot starts to travel from C_i to C_j along the bold tangents at time $\bar{t}$. In (a) the bold tangent is a right-right tangent, and in (b), it is a right-left tangent.	110
5.6	The logarithmic spiral around a growing disc which grows at speed 1. The speed of the spiral is V_R . A point on the spiral is specified as $(r(t), \Theta(t))$, and $r(t) = t$	111
5.7	L is tangent to C_0 , $p \in C_0$, $o_i q \perp L$. The ending point w of $l_{0i}^{rl}(t_1)$ is located inside V_{C_i} , where $ l_{0i}^{rl}(t_1) = L $. The dotted curves are Voronoi edges. The dashed disc is centered at w , and is tangent to C_i	115
5.8	(a): $slope(l_{0j}^{rl}) > slope(l_{0i}^{rr})$, and C_j is lower than C_i along l_{0i}^{rl} . (b): $slope(l_{0j}^{rl}) < slope(l_{0i}^{rr})$, and C_j is lower than C_i along l_{0i}^{rr} . (c): $slope(l_{0j}^{rl}) < slope(l_{0i}^{rr})$, and C_j is higher than C_i along l_{0i}^{rr}	117
5.9	(a) $slope(l) < slope(l_{0j}^{rr})$. (b) Tangent $l = l_{0i}^{rl}$ is located between l_{0j}^{rr} and l_{0j}^{rl} in CCW order, C_j is lower than C_i along l , l is invalid. The growth of the discs is not shown in the figure.	122

5.10	Sweep circle C passes through event point p , which is the end point of l_{0i}^l . $C \cap C_i = \{p, q\}$. Segment $\hat{l}$ is tangent to C_0 , $ \hat{l} = d$, with endpoint q . Sweep circle C' passes through the ending point of l_{0i}^{rr}	125
5.11	An arrangement of the departure curves on disc C_0 is illustrated in the polar coordinate system, with the various-speed discs. The vertical line is $DC(l_{0i}^{rr})$, with $V_0 = V_i$. The rests are right-left or right-right departure curves, each of which is part of a sine curve.	127
5.12	For the proof of Property 5.2. The intersection point q of two departure curves using Cartesian coordinates. $DC(l_{0i}) \cap DC(l_{0j}) = q$. Tangent $l_{0i}(t_q)$ and $l_{0j}(t_q)$ are co-linear.	128
5.13	An example of $\Theta(n^2)$ intersection points in the arrangement of the departure curves around disc C_0 . The speed of each disc C_i in the middle is $V_i \rightarrow 0$. $V_R \rightarrow \infty$. The speeds of other discs C_j , $V_j = V_0$. V_0 is big enough such that each C_i will touch l_{0j}^{rr} once at different times without intersecting other C_i s. Imagining that the discs grow at the given speeds. For a specific C_j in the figure, l_{0j}^{rr} is parallel to each other at different times, as $V_0 = V_j$. Since discs C_i grows at a very small speed, l_{0j}^{rr} may be tangent to a C_i from the left side of C_i , and then is tangent to C_i from the right side of C_i	131
5.14	$cone_0$ intersects $cone_1$ and $cone_2$ with two Descartes curves. Project the Descartes curves onto the x - y plane. It forms the Voronoi cell V_0 for the corresponding disc C_0 , V_1 for C_1 , and V_2 for C_2	134
5.15	Doubly connected edge list data structure for a polygon with holes. .	134
5.16	An invalid right-left tangent becomes valid again. $l_{0i}^l(T_0)$ is invalid, while $l_{0i}^l(t)$ is valid. (a) $l_{0i}^l(T_0)$ is invalid because of C_j . (b) The departure curve on C_0 . The buffer candidates on the departure curve are located inside V_{C_0}	137
5.17	For the proof of Lemma 5.18 (2). Assume that $DC(l_{0i}^{rr})$ intersects the Voronoi edge between C_0 and C_j with p and q	139

A.1	(<i>a</i>) for the right-right function computation. (<i>b</i>) for the right-left function computation.	159
-----	--	-----

Chapter 1

Introduction

1.1 Motivation

A Geographic Information System (GIS) [53] is a computer system capable of integrating, storing, editing, analyzing, sharing, and displaying geographically-referenced information.

GIS technology is widely used by businesses, governments and individuals. Over the years, through augmentation of functionality, these systems grew in scope and have become powerful tools for operational planning and decision making. However, the systems have been monolithic and as such are often cumbersome, hard to use and to customize. Frequently, it takes several years to deploy them (including data integration), and requires large amounts of computing resources. To meet today's demands posed by a dynamic, mobile, and fast moving environment, new ideas must be developed.

Ubiquitous computing is an important concept proposed by Weiser [117] in the early 1990s. Through ubiquitous computing, people are connected everywhere, at any time using embedded and hidden devices. Now, after more than a decade of research and development, ubiquitous computing becomes possible: (*i*) The growth in communication capabilities, especially in the wireless domain, enables better access to information, data, as well as easy and independent interaction between people

within and outside their community and geographic vicinity. (ii) The size of electronic devices has been reduced, and even more functionality has been integrated into a single device. Most new devices, including cellular phones, are capable of capturing, transmitting and receiving digital data (including images and video) as well as traditional voice data (for telephone calls). (iii) The cost of electronic devices, e.g., mobile phones and PDAs, as well as communication network access/usage fees has been constantly decreasing. GIS technology has adapted to these trends, and the traditional usage of maps and GIS has been extended to real-time and seamless access and/or usage of spatial information. Examples are: one uses an Internet mapping program (such as GoogleMap or MapQuest) to seek directions, or one uses a navigation system in a car instead of maps. As a result, GIS is becoming an integral part of our lives, and our dependency on spatial data and spatial data manipulation is growing, as well as the demand for spatially referenced services.

The intelligent map agents system (IMA), proposed here, uses agent technologies to introduce new, generalized concepts for working with spatial and geographic data. It aims at creating a mobile user-centric ubiquitous environment where users can access, use and manipulate spatial data explicitly as well as implicitly, e.g., by requesting services requiring spatial data. User interaction with, or within, the system is facilitated through agents. An *intelligent map agent* is a software agent [84], situated in a system that consists of users, geographic data, networks, possible other agents, data sources, and distributed geographic services, all of which are ready to serve the user by taking into account the user's needs and state. Thus, the objective of our work on the IMA system is to develop a framework which permits the proliferation of geographic data within our society through seamless connection, collaboration between users, and interaction with distributed geographic services and data centres. This is achieved by moving away from monolithic geographic information systems into a lightweight environment where users can access and select data, functionality and visualization according to their needs. The research focuses on design and implementation of the framework of the IMA system, and providing efficient algorithms

for the services in the framework (or similar services).

We focus on two services: a meeting scheduler service and a shortest path service. The meeting scheduler service aims to schedule a new meeting among a group of people distributed in different locations. Traditional meeting scheduler service, for example, the palm computer, only considers time. However, the time spent on traveling to and from the new meeting location has effect on the determination of the new meeting location. Thus, the novelty of the meeting scheduler service is scheduling a new meeting based on time and location information. We provided the first meeting scheduler algorithms with spatial and temporal information.

As we know, computing an optimal path in a geometric domain is a fundamental problem in computational geometry, having many applications in robotics, GIS, wire routing etc. [79]. The computation of optimal paths has been studied in GIS mainly for road networks (see De Florian et al. [37]). For example, one of the applications is a routing problem: delivering packages to and from different locations, with time restriction. Each path to be traveled should be as short as possible to cut down on the cost of distribution. Another example is that an optimal path construction is applied to some emergency systems such as 911 fire alarms. These systems require that the computation of an optimal path be almost 1-3 seconds. Furthermore, during travel, the next path of the vehicles should be calculated in time. The shortest path service in our IMA system aims at providing the minimum traveled path in a spatial environment, for example, in a road network, or on a terrain, between start and end locations. We study a shortest path problem which can be applied in the following scenario: evacuating residents rapidly from one location to a safe place while passing through several locations on fire, where, the fire expands outwards fast (modeled by growing discs). To be concise, the problem we studied is seeking a shortest path among growing discs for a point robot.

The research related to the two services in the IMA system are introduced separately in the next section.

1.2 Summary of thesis results

In this section, we summarize the results obtained in this thesis. The thesis is divided into two parts. The first part is dedicated to the IMA system including a description of its architecture design and implementation. In the second part, we present algorithmic solutions to two services which can be used as services within the IMA system. These are: (i) meeting scheduling services, aiming at scheduling a new meeting among a set of participants, based on temporal-spatial information, and (ii) shortest path services, specifically dealing with finding a shortest path among obstacles which are growing discs, and a constant speed on the shortest path is considered. Next, we discuss the results of each chapter in detail.

1.2.1 IMA system architecture design and implementation

The main focus of the IMA is to provide access to services, which require the usage of spatial data and spatial information, for mobile users regardless of their location. The IMA achieves this via an open architecture which supports a distributed set of services instead of a single monolithic spatial information system. The open architecture can accommodate a large number of mobile users and services which may be distributed across a wide geographical area. With respect to this, the key component of the IMA is its services oriented technology. The system adapts the Services Oriented Architectures (SOA) [8]. SOA has three parties: a provider, a broker, and a requester. A provider offers a business service(s) whose implementation is not directly visible to the requester. A requester is a consumer of services. It learns from the broker the information structure that it has to send and receive from the provider and what protocol to use to access that service. A broker is a match maker where providers publish their services, and where requesters discover the services.

The services, described in the following section, as well as users, are facilitated by agents using JADE/LEAP technologies. We choose the JADE agent technology, for the following reasons: (1) to remedy the deficiencies of mobile users' communication

and computation abilities, (2) to reduce the users' burden on multitasking performance, (3) to improve the scalability of services: the system needs to provide more services, and each service assists more users, (4) to support communication fault tolerance: an agent deals with the situation when the connection between a mobile user and the system is broken, (5) to improve the inter-operability of users, regardless of the devices and environment the users are operating in. The IMA system has three basic tiers: user, functionality service and data service tiers. The design & development of each tier, is guided by particular design principles. For example, for the user tier, we design a context manager to maintain each user's personal information. For the functionality service tier, we use the broker provided by JADE agent to register the users and publish the services, and use service worker agents to improve the scalability of the services. The data service tier is responsible for the connection with geographic data centers.

The IMA system development is a collaborative effort of a group of researchers [42, 43, 44, 83]. I took over the responsibility for building the framework of the IMA system in 2004. I have achieved the following:

1. Built the framework based on an incomplete Master's thesis (by Hongsheng Liu): redesigned the user interfaces, and the interface between service domain and resource domain.
2. Achieved a distributed system, i.e., the services and the users are located in a distributed environment: applied JADE's mechanism of distributed system into the IMA system.
3. Tested the framework by incorporating a simplified meeting scheduler service, which in turn calls for a shortest path service (Dijkstra's shortest path algorithm is applied): fulfilled the meeting scheduler user and service agents, and modified the shortest path interface in C.

This work has been described in [43].

1.2.2 IMA system services

According to the definition given in [11], *software as a service* is a software application delivery model where a software vendor develops a web-native software application and hosts and operates (either independently or through a third-party) the application for use by its customers over the Internet. The IMA system design enables a large number of services to be offered simultaneously. A simple example is the meeting scheduler service, which in turn calls for a shortest path service. Both services are related to theoretical problems in Computational Geometry. We designed efficient algorithms for solving these problems which are introduced next.

1.2.2.1 Algorithms for meeting scheduler services

Meeting scheduling services aid users to schedule a new meeting among them satisfying particular objectives. In this work, we study the problem where a group of people, each associated with their own meeting calendars, would like to find a meeting location with largest meeting time that can fit into each participant's schedule. More specifically, a set of n participants, each participant P_i has a previous meeting $M_1^i(l, t_1, t_2)$ and a subsequent meeting $M_2^i(l, t_1, t_2)$, where, l is the meeting location, t_1 is the meeting starting time, and t_2 is the meeting ending time. The problem is to schedule a new meeting $M^*(l, t_1, t_2)$ with the longest duration ($\max ||t_2 - t_1||$), such that the new meeting can be added to each participant's schedule, without violating the participant's original meeting schedule. I.e., the latest participant arriving at the new meeting location $M^*(l)$ should be at time t_1 after his/her previous meeting, and the earliest participant leaving $M^*(l)$ should be at time t_2 such that he/she can arrive at his/her subsequent meeting location on time. The novelty of the meeting scheduler service is that not only time is considered, as in current commercial products, but also geographic locations. By taking into consideration both time and space limitations of the users, we distinguishes our problem from the current practice of scheduling meetings based only on time limitations of its participants [17, 28, 48, 54, 100, 101, 107, 109, 116]. We offer several algorithms to tackle the

meeting scheduler problem. The first algorithm provides a geometrical solution to the problem based on geometrical properties. The second algorithm, uses linear programming techniques. The third algorithm focuses on the case when no new meeting can be scheduled for all of the participants, i.e., violations exist. For that problem instance, we find a new meeting location allowing the maximal number of participants to take part in. Finally, we briefly consider a graph-based model and derive the computational complexity of finding the optimal meeting time/location based on this model.

This work resulted in the conference papers [18] and [62], and a journal paper [19].

1.2.2.2 Algorithms for shortest paths services

Shortest path services are fundamental services provided by the IMA system. In this thesis, we propose an algorithmic solution which will lead to a dynamic shortest path service. This research investigates the problem of finding a shortest path among moving obstacles, and the moving directions of the obstacles are uncertain. The obstacles are thus modelled as growing discs. More specifically, given a set of n discs in the plane, the goal is to find a collision free shortest path, for a point robot with a steady speed V_R starting from a source point s and ending at a destination point g . In [114, 115], van den Berg and Overmars studied the dynamic shortest path problem when all discs grow at the same speed. They provide a solution with $O(n^3 \log n)$ running time, where, n is the number of discs. We improved the bound by a factor of n to $O(n^2 \log n)$ [70]. When the discs grow at various speeds, there is no time complexity applied in their algorithm. We give an $O(n^3 \log n)$ time algorithm for that [71].

1.3 Organization

This thesis is organized as follows: Chapter 2 reviews the related literature. Chapter 3 focuses on the IMA system architecture and its implementation. Chapter 4 presents algorithms related to the meeting scheduler service. Chapter 5 describes the dynamic

shortest path algorithms. Conclusions and future work are listed in Chapter 6.

Chapter 2

Background

This chapter gives an overview of existing results in the areas of research we address. Section 2.1 discusses previous work related to our applied work on the IMA system. Section 2.2 discusses existing results related to our more theoretical research. For the meeting scheduler service in the IMA system, the related theoretical research is presented in Chapter 4. The theoretical research on the special shortest path problem as related to our shortest path service is presented in Chapter 5.

2.1 Related applied work

2.1.1 Agent technologies

A *software agent* is software which can act autonomously or semi-autonomously in order to accomplish a task on behalf of a user. Software agents do not operate on their own, rather, they operate in an agent platform. A popular and still growing agent platform is JADE (Java Agent DEvelopment Framework). We cite [39]: “JADE is a software framework fully implemented in Java. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA (Foundation of Intelligent Physical Agent) specifications and through a set of graphical tools that supports debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time

by moving agents from one machine to another one, as and when required". Other agent platforms include TACOMA [56] and Aglets [64]. TACOMA (Troms And Cornell Moving Agents) is a lightweight distributed computing middleware platform. It provides a secure message passing mechanism and a simple framework for implementing application specific message handlers. It is a generic toolkit for building closed distributed communication infrastructures. Aglets is a Java-based mobile agent platform and library that eases the development of agent-based applications. All agent platforms support mobile agents.

Agents can be pro-active or reactive, static or dynamic, and/or mobile. Agent technology has been used in different areas, for example, searching for the cheapest flights to a destination or locating a restaurant that is closest to the user.

2.1.2 Systems using agent technologies

Agent research took its origins from collaborative agents, which appear in Distributed AI and Multi-Agent Systems. Agents negotiate in order to resolve conflicts (e.g., meeting time). Some agents collaborate to integrate information. Agents wrap around legacy systems ("glue" to interconnect them), and provide solutions to inherently distributed problems, e.g., air traffic control and telecommunications network management. Strong forms for collaborative agents are: Georgeff's BDI (Beliefs-Desires-Intention) in the distributed Multi-Agent Reasoning System (dMARS) [33], and Shoham's AOP (Agent-Oriented Programming) in modal logic [68].

Research in [35] focuses on developing an agent-enhanced Work flow Management System (WfMS), called JBees, where the functional work associated with running a WfMS has been partitioned among various collaborating agents that are interacting with each other by following standard agent communication protocols (FIPA).

Moraitis et al. [81] combine the Gaia methodology [119] and the JADE development environment in order to implement a real multi-agent system (MAS). It focuses on the functionalities of the services, while exploiting existing web services of different service providers. The Gaia methodology is an attempt to define a complete and

general methodology that is specifically tailored to the analysis and design of MASs. By applying the Gaia methodology, the real multi-agent systems are composed of a number of autonomous interactive agents and each agent plays one or more specific roles. Gaia defines the structure of a MAS in terms of a role model. The model identifies the roles that agents have to play within the MAS and the interaction protocols between the different roles. An agent based system with focus on automated generalization has been described in [63].

Several other research groups are investigating the usage of agent technologies in conjunction with geographic data. However, these systems are special purpose systems each of which targets a particular application. As far as we know, the project: CReation of User-friendly Mobile services PErsonalised for Tourism (CRUMPET) [65, 97, 105, 120], seems to be one of the first system employing agent technology for implementation, validation and trial on GIS applications. The apparent objective of CRUMPET is the creation of user-friendly mobile services personalized for tourism, and using agent technology for fast creation of robust, scalable and seamless access to nomadic services. The CRUMPET project took the use of geographical information technology one step further by developing an integrated, spatially enriched information architecture through the intensive use of distributed intelligent agent technology based on the FIPA standard. Some of the key technologies used in CRUMPET are intelligent agent technology, embedded devices technologies (PDAs, Mobile phones), UML, XML and Java. The CRUMPET research work tackled the problems as follows: Using a handheld device, tourists can check out an array of information such as their nearest Italian restaurant, or the time of public transport in another city by using a digital map of that city. GPS is used to establish the user's position. While similar electronic services such as digital tour planners or restaurant guides are now on the market, they are not linked together. With the CRUMPET system, a central database has been established allowing all these services to be integrated into a single working network. Users, independently of their location, can access information elsewhere on the network. The CRUMPET project claims that the agent platform is

suitable for the Universal Mobile Telecommunications System (UMTS) and for local wireless networks (W-LAN).

Another research framework is the Deep Map Project [72]. Its objective was to develop a digital personal mobile tourist guide using GIS, databases, natural language processing, intelligent user interfaces and knowledge representation, conducted in EML (Europe Media Lab), which can be seen as pre-work to the CRUMPET project.

The Monads Research Project [78] has implemented an agent-based architecture providing adaptive services. It addresses the problem associated with adapting land line software applications for wireless networks, namely variations in Quality-of-Service (QoS). The research project studies the short-term predictability of available QoS, enabling applications to adjust their behavior dynamically to the forthcoming QoS. The results presented indicate that intelligent agents can learn the key characteristics quite quickly, including tempo-spatial variations of available QoS, so that reasonable predictions of available QoS in the near future are possible.

2.1.3 Software architectures

The architecture we adopted in the IMA system is the Service Oriented Architecture (SOA) [8]. This architecture has service providers and service consumers within a distributed computing network, and focuses on the basis for interaction between components on those same defined services, i.e., service requests, service responses, or service exceptions. SOA is widely used by the Web Services. A web service is a SOA with at least the following additional constraints:

1. interfaces must be based on Internet protocols such as HTTP, FTP, and SMTP.
2. except for binary data attachment, messages must be in XML.

SOAP-WSDL (Service Oriented Architecture Protocol - Web Service Definition Language) standard for “publishing, finding and binding” services of W3C (World Wide Web Consortium) [29] is the most popular web service.

OpenGIS Web Service (OWS) Architecture [69] is a common architectural framework for the design and implementation of Open Distributed Processing applications based on a Web Services platform. It defines a framework of guiding concepts, terminology, fundamental patterns and organizing principles for implementation and deployment. It establishes common interfaces, exchange protocols, and services that can be utilized by any application, to enable interoperation between instances of Web Services.

OpenGIS Location Service (OpenLS) architecture provides five core services: (1) a Directory Service provides access to an online directory for a specific location, (2) a Gateway Service finds the position of a known mobile terminal, (3) a Location Utility Service provides a transformation between the description of a location to a point with Cartesian coordinates, (4) a Map Portrayal Service displays a map and (5) a Route Service for a travel route.

OpenGIS Web Map Service (WMS) [108] architecture defines a simple interface for web based mapping applications. The WMS protocol is based on a simple query syntax for posting a request for the desired registered layers and region of a map to the map server, which returns a map as a standard image (GIF, PNG or other format). Accompanied with the returned maps, the detailed information of a feature at a location will be provided. Any client can access maps from any map server.

Location Based Services (LBS) are capable of providing real time data through different kinds of positioning methods by embedding the technology for mobile users. The standard specifications in providing an LBS service are OpenLS, WMS, and other standards from OpenGIS Consortium (OGC), e.g., GML, an XML-version specification of the OpenGIS for vector-based map content (geographic features) for GIS. By following the specification recommended by OGC, data sharing, data analysis and manipulation can be enhanced.

Figure 2.1: (a): Voronoi diagram of a set of points (b) order-2 Voronoi diagram of the set of points (c) order-3 Voronoi diagram of the set of points (d) furthest Voronoi diagram of the set of points

2.2 Related geometric concepts and algorithms

In this section, literature related to the theoretical research is reviewed and background algorithms are introduced.

2.2.1 Voronoi diagrams

Due to the large number of their applications, Voronoi diagrams have received considerable attention from researchers not only in the area of computational geometry,

but also from areas like crystallography, biology or economics. In this section, we first briefly introduce those Voronoi diagrams which are related to our research. Then, we present the details of their construction algorithms.

1. Nearest Voronoi diagram of a set of points

The Voronoi diagram of a set of n distinct sites (points) in a plane is a partition of the plane into convex polygons, such that each polygon contains exactly one generating site and every point in a given polygon is closer to its generating site than to any other. To be precisely, consider a point set $\mathbb{P} = \{p_1, \dots, p_n\}$. The Euclidean distance from a point q to a site p_i is defined as: $d(q, p_i) = \|q - p_i\|$. The Voronoi cell of p_i , V_{p_i} , is defined as:

$$V_{p_i} = \{q \in \mathbb{R}^2 \mid \forall j, d(p_i, q) < d(p_j, q)\}.$$

This Voronoi diagram is referred to as nearest neighbor Voronoi diagram, order-1 Voronoi diagram, or Dirichlet tessellation [12], where, the cells are called Dirichlet regions, or Thiessen polygons. Shamos and Hoey [110] first introduced the planar Voronoi diagram and presented how to construct it in $O(n \log n)$ time by divide-and-conquer. Fortune [38] gave an $O(n \log n)$ algorithm to construct a Voronoi diagram by sweeping line technique.

2. Furthest Voronoi diagram and Order- k Voronoi diagram of a set of points

Contrary to the nearest neighbor Voronoi diagrams, if every point in a convex polygon is further from its generating site than from any other site, i.e., the Voronoi cell of p_i , V_{p_i} , is defined as:

$$V_{p_i} = \{q \in \mathbb{R}^2 \mid \forall j, d(p_i, q) > d(p_j, q)\},$$

then this Voronoi diagram is called furthest Voronoi diagram (*FVD*).

If every point in a convex polygon is closer to the same k sites, and each convex polygon has k sites corresponding to it, then this Voronoi diagram is called Order- k Voronoi diagram [67, 102, 106]. Refer to Figure 2.1 for the nearest

Voronoi diagram, order-2 Voronoi diagram, order-3 Voronoi diagram and the furthest Voronoi diagram for the same set of points.

Furthest Voronoi diagrams and order- k Voronoi diagrams are useful for certain optimization problems, such as minimizing the maximum distance and facility location problem etc. [3, 7, 10, 21, 52, 77, 85, 86, 87, 88, 89, 91, 95, 96]. For example, Aronov et al. [10] applied the furthest Voronoi diagram on a terrain to find a location p which minimizes the maximum distance to the sites on the terrain. The terrain is composed of n triangles. The distance is measured by the Euclidean shortest path distance along the surface of the terrain. The terrain is extended to a polyhedron similar to a box, such that the terrain is restricted to the top and bottom surface of the box, and the shortest path between two points on the terrain is located on the two surfaces. They compute the furthest Voronoi diagram of the set of sites on the polyhedron, and point p is either in a Voronoi edge or is a Voronoi vertex of the furthest Voronoi diagram.

3. Variations of the Voronoi diagrams

Various other generalizations of the original concept of Voronoi diagram have been studied since the 1970s, including variations of the distance metric, e.g., Euclidean distance metric, L_1 and L_∞ metric, shortest path metric [93], Hausdorff metric [92] and power function [13] etc.; and Voronoi diagrams in higher dimensions [34]. The sites can be points (weighted or unweighted), discs (spheres), curves, segments and polygons, etc.. The prolific references can be found in [14, 16, 90] and a Ph.D thesis of Gavrilova [41], who studied Voronoi diagram for circles and spheres, and Voronoi diagram in Euclidean, power, and (Manhattan) metrics.

4. Abstract nearest and furthest Voronoi diagrams in $2-d$

These Voronoi diagrams were studied in [60, 61, 76], respectively. Klein [60] provided a universal tool (divide-and-conquer approach) that can be used to compute all types of concrete Voronoi diagrams whose bisector curves have the

following three properties [61]:

- (a) The bisector of every two sites is homeomorphic to an open interval $(0,1)$,
- (b) The bisector between every two sites divides the plane into three disjoint sets, i.e., the points on the bisector also belong to a set,
- (c) The intersection of two bisectors consists of finite connected components.

As stated in [16], the approach in [60] does not apply to the points with multiplicative weights, since the bisector curves can be closed, rather than homeomorphic to a line.

Mehlhorn et al. [76] presented a more general way to compute all abstract furthest Voronoi diagrams by a randomized incremental approach.

Next, we emphasize on the additively and/or multiplicatively weighted Voronoi diagrams which are applied in our research.

2.2.1.1 Additively weighted Voronoi diagrams

Consider a point set $\mathbb{O} = \{o_1, \dots, o_n\}$ in Euclidean space. For each point $o_i \in \mathbb{O}$, there is a weight r_i associated with it. The additively weighted distance from a point p to a point site o_i is defined as: $d(p, o_i) = \|p - o_i\| - r_i$. Additively weighted Voronoi cells are defined as Voronoi cells in a nearest Voronoi diagram.

The additively weighted Voronoi diagram for the set of points with additive weights in $\mathbb{O}$, is the union of the additively weighted Voronoi cells V_{o_i} , refer to Figure 2.3 (a). It is the same as the one for a disc set $\mathbb{C} = \{C_i(o_i, r_i), 1 \leq i \leq n\}$ in $\mathbb{R}^2$, where, o_i is the center and r_i is the radius of C_i . The reason is as follows. The distance function from a point $p \in \mathbb{R}^2$ to C_i is $d(p, C_i) = \|p - o_i\| - r_i$, which is the same as the distance function from p to point o_i with weight r_i . Furthermore, suppose that all the discs in $\mathbb{C}$ expand at the same speed v , i.e., the growing speed of each disc is v . For a point p which has the same distance to C_i and to C_j , the function is: $d(p, C_i) = \|p - o_i\| - r_i - vt = d(p, C_j) = \|p - o_j\| - r_j - vt$. This function is the

Figure 2.2: When an edge e in (a) is flipped, the topology is changed to e' in (b).

same as the function $\|p - o_i\| - r_i = \|p - o_j\| - r_j$. Therefore, we have the following property:

Property 2.1. *The additively weighted Voronoi diagram for the discs in a set $\mathbb{C}$, is the same as the Voronoi diagram constructed for the discs in $\mathbb{C}$ starting to grow at the same speed and at the same time.*

In an additively weighted Voronoi diagram, an edge is generally a hyperbolic arc or a line segment. Sharir [112] first dealt with the construction of the Voronoi diagram for a set of discs. He built the Voronoi diagram for a set of n discs in $O(n \log^2 n)$ time by a divide-and-conquer algorithm, and proposed as open problem to improve the time complexity to $O(n \log n)$ time. Currently, a number of algorithms construct additively weighted Voronoi diagrams in $O(n \log n)$ time [6, 57, 58, 94, 118], by using the popular methods: incremental, sweep line, divide-and-conquer and edge-flipping methods etc.. For example, Kim et al. [58] compute a Voronoi diagram for discs contained in a large circle, by computing the Voronoi diagram of the set of the inner disc centers first, then by edge-flip (see Figure 2.2 for an example) operations on this Voronoi diagram computed so far, the final Voronoi diagram is obtained. Weringh [118] constructs the Voronoi diagram for a set of discs by divide-and-conquer.

Figure 2.3: (a): Voronoi diagram of a set of discs (b) furthest Voronoi diagram of a set of discs

2.2.1.2 Multiplicatively weighted Voronoi diagrams

Consider a point set $\mathbb{P} = \{p_1, \dots, p_n\}$. For each point $p_i \in \mathbb{P}$, there is a weight w_i associated with it. The multiplicatively weighted distance from a point q to a site p_i is defined as: $d(q, p(i)) = 1/w(i) \|q - p(i)\|$. Multiplicatively weighted Voronoi cells are defined the same way as Voronoi cells in a nearest Voronoi diagram.

The Voronoi diagram for the points with multiplicatively weights is called multiplicatively weighted Voronoi diagram [15, 104]. As shown in [15, 51], the bisector between two weighted point sites in a multiplicatively weighted Voronoi diagram is a circle, or a line segment, and the Voronoi region(s) of a site can be non-connected and non-convex. Aurenhammer and Edelsbrunner gave an $\Theta(n^2)$ algorithm to construct a multiplicatively weighted Voronoi diagram for a set of weighted points [15]. They projected the bisectors (circles) between a point site, say p , and the other point sites

onto a 3-dimensional sphere. The projection forms a convex polyhedron, and the XY -plane where the bisectors are originally located was geometrically transformed to a sphere. Thus, for site p , its Voronoi region can be computed in $O(n)$ time, by computing the intersection of the convex polyhedron with the sphere corresponding to the XY – plane. The Voronoi diagram can consist of $\Theta(n^2)$ faces, edges and vertices. The total time complexity of the Voronoi diagram construction is $\Theta(n^2)$. The multiplicative weighted Voronoi diagram computed in [104] is quite different from Aurenhammer and Edelsbrunner's. The distance from a multiplicatively weighted site to a point in the Voronoi region of the site is measured along a shortest path lying entirely within the Voronoi region.

2.2.1.3 Additively, multiplicatively weighted Voronoi diagrams

Consider disc set $\mathbb{C}$ in $\mathbb{R}^2$. Each disc $C_i(o_i, r_i) \in \mathbb{C}$, has an associated weight w_i . A Voronoi diagram of the disc set is called additively multiplicatively weighted Voronoi diagram, or compoundly weighted Voronoi diagram [90]. An edge in a compound weighted Voronoi diagram is a Descartes oval [31], a line, a hyperbola, or a circle. The Descartes oval is explained next. The bisector between two discs $C_i \in \mathbb{C}$ and $C_j \in \mathbb{C}$ is denoted by $J(C_i, C_j)$. Let p be a point on $J(C_i, C_j)$. $J(C_i, C_j)$ can be obtained via the following equation:

$$\frac{\|p - o_i\| - r_i}{w_i} = \frac{\|p - o_j\| - r_j}{w_j}$$

$$\|p - o_i\| - \frac{w_i}{w_j} \|p - o_j\| = w_j r_i - w_i r_j$$

Let $h = \frac{w_i}{w_j}$, $h > 0$, and let $k = w_j r_i - w_i r_j$, $k < 0$, or $k = 0$, or $k > 0$. Then, the equation becomes

$$\|p - o_i\| - h \|p - o_j\| = k, \quad (2.1)$$

which is named as Descartes oval [31], studied in [22, 50]. Also refer to Figure 2.4. Applications of additively multiplicatively weighted Voronoi diagram, for example,

Figure 2.4: An example of the bisector $J(C_i, C_j)$ with function $\|p - o_i\| - \frac{w_i}{w_j}\|p - o_j\| = w_j r_i - w_i r_j$, $p \in J(C_i, C_j)$. $\frac{w_i}{w_j} = 1.5$, $\|o_i - o_j\| = 10$, the numbers on the curves represent the possible values of $w_i r_j - w_j r_i$.

can be found in [22, 50, 82]. However, to our best knowledge, we have not found any algorithm to compute this Voronoi diagram efficiently with time complexity analysis.

In the following sections, we will focus on furthest Voronoi diagrams for discs, refer to Figure 2.3 (b). As the computations are related to cones, we introduce the cone intersections first. Many authors have studied the additively weighted furthest Voronoi diagrams, we mainly refer to Rappaport [99] and Mehlhorn et al. [76] in the next section.

2.2.1.4 Furthest Voronoi diagram of a set of discs, cone intersection, and minimum spanning circle

1. Cone intersection and furthest Voronoi diagram of a set of discs

Consider a disc set $\mathbb{C}$ of size n in $\mathbb{R}^2$, $C_i(o_i, r_i) \in \mathbb{C}$, $1 \leq i \leq n$. Define a set of n cones in $\mathbb{R}^3$, $\mathcal{C} = \{cone_1, cone_2, \dots, cone_n\}$, where, t is the third dimension. For each $cone_i \in \mathcal{C}$, there is a disc $C_i \in \mathbb{C}$ corresponding to it.

$$cone_i = \{(p, t) \in \mathbb{R}^3 | p \in \mathbb{R}^2, t = \|p - o_i\| - r_i\} \quad (2.2)$$

Figure 2.5: Two cones intersect with a hyperbola in 3- d . Two discs are formed by cutting the cones with a horizontal plane. The projection of the hyperbola onto $\mathbb{R}^2$ is a Voronoi edge between the two discs.

I.e., if we cut the cones at a horizontal plane $t = 0$, then we obtain the circle set, $\mathbb{C} = \{C_1, C_2, \dots, C_n\}$. $Cone_i$ is an upwardly directed cone in $\mathbb{R}^3$, with a vertical axis t (time), apex o_i , and an interior angle of 90° . Cones may be viewed as bivariate functions on $\mathbb{R}^2$. In Chapter 4, cones are used to model the motion of participants with a known steady travel speed, but unknown travel directions. I.e., for a point (p, t) on a $cone_i$, p is a point that the participant can reach at time t . In Chapter 5, these cones model discs growing at the same speed. I.e., for a point (p, t) on a $cone_i$, p is a point on the disc $C_i(o_i, r_i)$ at time t .

We define the lower envelope of $\mathcal{C} = \{cone_i | C_i \in \mathbb{C}, 1 \leq i \leq n\}$, as the surface composed of that portion of each cone that lies below all other cones. From the definition of $cone_i$ (Equation 2.2), it is evident that this lower envelope projects vertically onto $V(\mathbb{C})$ (the Voronoi diagram for the disc set $\mathbb{C}$) onto $\mathbb{R}^2$. In Equation 2.2, intuitively, a smaller t means a smaller distance to the corresponding disc C_i in $\mathbb{R}^2$. Refer to Figure 2.5. Similarly, we can define the upper envelope of $\mathcal{C} = \{cone_i | C_i \in \mathbb{C}, 1 \leq i \leq n\}$. The projection of the upper envelope of the cones corresponds to the furthest Voronoi diagram for the discs in $\mathbb{C}$. This can also be explained by the following observation.

Let $E > 0$ be a constant and let C_i be the disc that results from intersecting cone $cone_i$ with the plane H that is parallel to, and of height $t = E$ above the XY -plane.

Figure 2.6: H is a plane parallel to XY plane with height E . disc $C_i = H \cap \text{cone}_i$.

Refer to Figure 2.6. The radius of C_i , is $r = E + r_i$, as the cone has a 90° opening angle. Let q be a point in the XY -plane, and let z be the point on H that lies vertically above q . Then the distance from z to disc C_i , that is, to its boundary, equals $\|o_i - q\| - r = \|o_i - q\| - E - r_i$. On the other hand, cone_i intersects the vertical line through q at time $\|o_i - q\| - r + E = \|o_i - q\| - r_i$. These two values differ only by E , which is the same for all the cones. Therefore, consider the distances from a point p to the cones, where, the distances are the lengths of the vertical lines from p to the surfaces of the cones. If cone_i is furthest to p , then the corresponding disc C_i is furthest to p among all the discs in $\mathbb{C}$.

Also observe that the value of constant E is not relevant. A change to E would cause all disks C_i to shrink or grow by the same amount—it would not alter the Voronoi diagram. Therefore, different disc sets are obtained when the cones are cut at different horizontal planes, and the furthest Voronoi diagram for each of those disc sets remains the same.

Consider the furthest Voronoi diagram for the discs in $\mathbb{C}$. Each $C_i \in \mathbb{C}$, produces

Figure 2.7: The dashed discs are the determining discs of the minimum spanning circle (MSC) g of the set of discs. (a): Two determining discs a and b . The center of a , b and g are on a line. (b) Three determining discs a , d , and e . (c): When the radii of the discs grow at the same speed, discs a , d and e intersect at a common point p , and all other discs include p .

one connected Voronoi region, containing all points p for which C_i is the furthest disc in $\{C_1, C_2, \dots, C_n\}$. The region of C_i can be empty, which happens if, and only if, C_i contains another disk C_j . This follows because C_j would be further away from each point p of the plane than C_i is. Two adjacent regions, of discs C_i and C_j , are separated by a hyperbolic arc, called a Voronoi edge, which consists of all points in the plane for which C_i and C_j are the furthest discs. Three or more such Voronoi edges meet at a Voronoi vertex.

2. Minimum spanning circle

The *minimum spanning circle* (MSC) $C(o, r)$, $r > 0$, for a set of discs $\mathbb{C}$, is defined as a circle with the smallest radius, that intersects each disc in $\mathbb{C}$, where, o is the center of C , and r its radius. Note that containment is also intersection. When the discs have a common intersection, we define that there is no MSC for them. Refer to Figure 2.8 (a). $C(o, r)$ is determined either by two or by three discs. If two discs $C_1(o_1, r_1)$ and $C_2(o_2, r_2)$ determine $C(o, r)$, then o , o_1 and o_2 are on a straight line, $r_1 + r_2 + 2r = |o_1 o_2|$. I.e., the two determining discs C_1 and C_2 are outside MSC and

Figure 2.8: (a) No minimal spanning circle of C_1 , C_2 and C_3 . (b) The minimum spanning circle C of three circles C_1 , C_2 and C_3 .

tangent to MSC. Similarly, if three discs $C_1(o_1, r_1)$, $C_2(o_2, r_2)$ and $C_3(o_3, r_3)$ determine $C(o, r)$, then discs C_1 , C_2 and C_3 are outside MSC and tangent to MSC. Each disc in a disc set that determines the center of the minimum spanning circle is called a *determining disc*. Refer to Figure 2.7 for an illustration. When the discs in $\mathbb{C}$ start to expand at the same speed, then the center of the MSC is the only intersection point of all the discs at a moment.

The center of the MSC of a set of discs can be computed in $O(n)$ expected time, where n is the number of discs, by using Megiddo's linear programming algorithm [75]. However, Megiddo's algorithm is suitable only for discs in general position. In particular, it does not solve the case when the minimum spanning circle (MSC) is fully included in an input disc, refer to Figure 2.8 (b). The minimum spanning circle $C(o, r)$ for the disc set $\mathbb{C}$ satisfies the following equation:

$$\|o - o_i\| + r_i \leq r.$$

In the case illustrated in Figure 2.8, the above formula cannot be applied. If we eliminate this case, then we can use [75] to compute the center of MSC for the disc set. Megiddo's algorithm works in the following way. First, generate the equation constraints for the problem in order to eliminate variables. The underlying optimization

problem is the following:

$$\begin{cases} \text{minimize } r \\ \text{subject to } \|o - o_i\| + r_i \leq r (i = 1, 2, \dots, n) \end{cases}$$

The inequalities can be written as: $\|o - o_i\|^2 \leq (r - r_i)^2$, and then let

$$f_i(o, r) = \|o\|^2 - 2(o_i)^T o + \|o_i\|^2 - r^2 + 2r_i r - r_i^2, \quad f_i(o, r) \leq 0.$$

Although the functions $f_i(o, r)$, $1 \leq i \leq n$ are not linear, they all have the same nonlinear terms, $\|o\|^2$ and $-r^2$. Thus, the equation $f_i(o, r) = f_j(o, r)$ defines a hyperplane

$$H_{ij} = \{(o, r) : 2(o_i - o_j)^T o - 2(r_i - r_j)r = \|o_i\|^2 - \|o_j\|^2 - r_i^2 + r_j^2\}.$$

Then, compute the extreme value (o', r') located on the hyperplanes, and go on with finding one side of the hyperplane such that a possible smaller value of r may exist in this side. Such a side may be found by solving a linear programming problem $(o - o_k)^T h < 0$ for a direction h of movement from o' , where, $\|o' - o_k\| + r_k = r'$. With the movement, $\|o - o_i\| + r_i$ is decreasing. Then, set $r = \max\{\|o(\epsilon) - o_i\| + r_i\}$, where, $o(\epsilon) = o' + \epsilon h$ and $\epsilon \geq 0$. At least the first linear piece of r indicates a direction of improvement and the correct side to go. It is clear that through the linear constraints, the dimension is reduced. In a fixed dimension, the search technique discards, in linear time, a fixed fraction of the constraints and hence the entire procedure runs in linear time.

In fact, Megiddo's algorithm is quite complicated in our case. It is related to linear programming in 3-dimensions, and a recursion on this computation. Comparatively, Megiddo's solution does not work well practically.

3. *Relationships among furthest Voronoi diagram of a set of discs, cone arrangement, and minimum spanning circle*

There are strong relationships between furthest Voronoi diagram of a set of discs, cones arrangement, and minimum spanning circle of the set of discs, summarized in

Lemma 2.1. Rappaport [99] showed how they are related to each other, and how they are computed by a method based on computing the convex hull of the set of discs. His algorithms will be applied in our algorithms in Chapter 4.

Lemma 2.1. (1) *The lowest intersection point of the set of cones, is the center of the minimum spanning circle(MSC) of the discs obtained by cutting the cones at a horizontal plane.*

(2) *The center of the MSC is determined by one of three cases. If a disc is enclosed in all other discs, then the center of the MSC is the center of the disc. Otherwise, the center is determined either by two or by three discs. Only the two or three discs are outside the MSC and tangent to MSC, if the sites are discs. If the determining discs are point discs, then they are located on the boundary of the MSC.*

(3) *The center of the MSC is either located on a Voronoi edge of the furthest Voronoi diagram of the set of discs, or is a Voronoi vertex in the furthest Voronoi diagram.*

2.2.1.5 Algorithm to compute the furthest Voronoi diagram of a set of discs

Rappaport [98] showed that the furthest Voronoi diagram (FVD) for a set of discs can be computed from a set of axis parallel cones. Let time t be the third dimension. The intersection of the set of cones with the plane $z = t$ is a set of discs at t . The volume produced by the intersection of the cones themselves is convex and unbounded. Rappaport showed that each vertex of the FVD is an intersection point of the convex body projected on the horizontal plane. We briefly summarize his solution for the intersection of the set of cones $\mathcal{C}$, the minimum spanning circle of the set of discs in $\mathbb{C}$, and the diameter of the set of discs in $\mathbb{C}$.

Let the apex of $cone_i \in \mathcal{C}$ ($1 \leq i \leq n$) be o_i , and o_i is located at plane $z = t_i$. Let convex body P be the intersection of the set of cones. Suppose that a point g , which is the highest among all intersection points of three cones (i.e., g is located on each of the three cones), of the convex body P is known in advance. We will come

(a)

(b)

Figure 2.9: (a) : The intersection of three cones $cone_1$, $cone_2$ and $cone_3$. (b) : The discs obtained by cutting the cones at the time plane t . The intersection of the discs are three arcs a_1 , a_2 and a_3 .

back to the computation of g later. Let the horizontal plane passing through point g be at $t = t_g$. This plane cuts the cones yielding a set of discs as in $\mathbb{C}$, possibly with different radii, denoted by $\mathbb{C}(t_g) = \{C_1(t_g), C_2(t_g), \dots, C_n(t_g)\}$. The center of $C_i(t_g) \in \mathbb{C}(t_g)$ is o_i . Its radius is $r_i = t_g - t_i$, ($1 \leq i \leq n$). The intersection of the discs is represented by a set of arcs $\Phi = \{a_1, a_2, \dots, a_m\}$ ($m \leq n$) listed in clockwise order which belong to the discs in $\mathbb{C}(t_g)$. Refer to Figure 2.9 for an example. The set of arcs is also the intersection of the set of cones with the plane $t = t_g$. For convenience, we use $cone_{a_j}$ ($1 \leq j \leq m$) to represent the cone corresponding to the arc $a_j \in \Phi$. The relationship between $cone_{a_j}$ and arc a_j is as the follows: Plane t_g cuts $cone_{a_j}$ to a disc, and arc a_j is part of the disc located in the intersection area of the set of discs $\mathbb{C}(t_g)$. Iteratively, we find an arc $a \in \Phi$ ($|\Phi| \geq 3$), whose predecessor and successor are $pre(a) \in \Phi$ and $suc(a) \in \Phi$, respectively, with the highest intersection point $q(a) = cone_a \cap cone_{pre(a)} \cap cone_{suc(a)}$. Then, delete arc a from Φ , that is: $\Phi = \Phi - \{a\}$. The last computed intersection point $p = q(a)$ located at the horizontal plane $z = t_p$ maybe the center of the minimum spanning circle for the set of discs in $\mathbb{C}(t_p)$, if the minimum spanning circle is determined by three discs. If the minimum spanning circle is determined by two discs, then we can compute the diameter of the set of discs, which will be introduced briefly later. Then the center of the minimum spanning circle is the center of the diameter.

The remaining problem is to define the highest intersection point g of the convex body of the set of cones. Rappaport [99] used the concept *stabbing line* in the disc set $\mathbb{C}(t_a)$ to solve this problem. The discs in $\mathbb{C}(t_a)$ are the intersection of the set of cones with the plane $t = t_a$. A stabbing line is a directed edge, tangent to two discs in $\mathbb{C}(t_a)$, the left side of the tangent line contains only the two tangent discs, and the right side of the tangent line contains or intersects all other discs in $\mathbb{C}(t_a)$. Refer to Figure 2.10. The regions which are composed of the discs on the left side of each stabbing line are called *stabbing regions*. Thus, there is a set of discs in the stabbing regions. The highest intersection point g is computed by detecting the circumcircle of every three neighboring discs in the stabbing regions of the disc set $\mathbb{C}(t_a)$, and

Figure 2.10: The stabbing lines for a set of discs. The stabbing region is composed of bold discs. The bold discs determine the stabbing lines.

the center of the circumcircle with the maximal radius is g . Similar to the minimum spanning circle of a disc set, the circumcircle has the minimum radius, and is tangent to the three neighboring discs and intersect all other discs in $\mathbb{C}(t_a)$.

How to compute the stabbing region for the set of discs in $\mathbb{C}(t_a)$? Let C_j be a disc in the set $\mathbb{C}(t_a)$ with the largest radius. We then reduce the radius of each disc C_k in $\mathbb{C}(t_a)$ by r_j , i.e., $\|r_k - r_j\|$. The discs on the convex hull of the set of discs with new radius, are the discs in the stabbing regions. He employed a divide and conquer paradigm to compute the convex hull of a set of discs. (An incremental algorithm to find the convex hull of discs in $O(n \log n)$ time can be found in [32]).

Now we show how to compute the diameter of a set of discs based on the convex hull of the discs. In [110], Shamos and Hoey used a rotating line to determine the diameter of a set of points, started with computing the convex hull of the set of points. Similarly, Rappaport used a rotating line parallel to each stabbing line to determine the diameter of the discs in $\mathbb{C}(t_p)$, recall that p is the lowest point on the convex body P .

We summarize the result from Rappaport [99] (also see Mehlhorn et al. [76] for

the furthest Voronoi diagram) as follows.

Theorem 2.1. *The furthest site Voronoi diagram of n discs contains $O(n)$ many edges and vertices. It has the graph structure of a tree, since all of its faces are unbounded. The furthest site Voronoi diagram of n discs, and the minimum spanning circle of the set of discs can be computed in $O(n \log n)$ time.*

Next, we present one application of furthest Voronoi diagrams which is closely related to our research.

2.2.1.6 Weighted farthest color Voronoi diagram on trees and graphs

Another variant of the Voronoi diagram is of particular interest to applications in location planning. It is called weighted furthest color Voronoi diagram ($WFCVD_G$) and was studied by Hurtado et al. [49]. We apply their results in Chapter 4.

There are two key words in the weighted furthest color Voronoi diagram. One is the weighted, the other is the colored variant. For the weighted variant, each site is assigned a weight factor, so as to model each site's importance (this variant is known as multiplicatively weighted Voronoi diagram). The colored variant applies to a situation where each of the n sites represents one of k different types of facilities, like schools, shopping malls, hospitals, etc.. A person choosing a new residence might want to have one representative of each type as close by as possible. The problem can be solved by means of the furthest color Voronoi diagram [1]. For each type i of utility, $1 \leq i \leq k$, there is a set S_i containing the locations of all utilities of that type. Each site is colored by type. Then, the cost function to be minimized can be formulated as:

$$c(x) = \max_{1 \leq i \leq k} \min_{s \in S_i} d(s, x),$$

where, $d(s, x) = ||s - x||$ is the distance between s and x . For each point x of the plane, let $c(i, x)$ denote the closest site of color i . Then, x belongs to the region of the site among $c(1, x), \dots, c(k, x)$ that is furthest away from x . The new residence is at the place which minimizes the cost $c(x)$. Refer to the example in Figure 2.11.

Figure 2.11: Cited figure in [1]. The furthest color Voronoi diagram for three colors and seven sites. The doubly connected unbounded area belongs to the site in the middle while the dark (resp. light) shaded areas belong to the closest dark (resp. light) sites (following the arrows).

By combining both the weighted and colored variant, Hurtado et al. [49] computed the weighted furthest color Voronoi diagram on a geometric graph (or tree): let n point sites be situated on the vertices or edges of a graph G over e edges. Each site is assigned a multiplicative weight and a color. The weighted furthest color Voronoi diagram of the n point sites is the division of the graph G , such that each point x of G belongs to a Voronoi cell of site s , where, s is the closest among all the sites which have the furthest color to x . Refer to Figure 2.12 for an impression of the diagram.

Theorem 2.2. [49] *The weighted farthest color Voronoi diagram of n points of k colors on a graph $G = (V, E)$ with $|V| = m$ and $|E| = e$ can be computed in $O(en\alpha(k) \log k + n \log n + nm \log m)$ time, where $\alpha(k)$ is the inverse of Ackermann's function.*

The computation starts with an edge in G . Consider first the case that all sites having the same color. The weighted Voronoi diagram on an edge of G (WVD_G) is

Figure 2.12: (a): Weighted distance functions of sites p , q and s . The *WVD* on a line is equivalent to the projection of the lower envelope of the weighted distance functions. (b): The weighted Voronoi regions on the edges (edges are bold). They are determined by different sites. The dashed lines are the lower envelope of the weighted distance functions. (c): When s is processed, r , p and q have been processed as they have less weights. The distance function of s is blocked by p and q 's.

equivalent to the lower envelope of the weighted distance functions $f: f = \pm w \cdot d(x, s)$, where x is a point on the edge of G , and w is the weight of site s . The distance function f starts at site s with slopes w and $-w$, refer to Figure 2.12 (a). The weighted Voronoi regions are $WVR(p_i) = \{x \in G | w_i \cdot d(x, p_i) < w_j \cdot d(x, p_j) \forall j \neq i\}$, where, p_i and p_j are different sites. The Voronoi region of a site may not be connected, see Figure 2.12 (a). In Figure 2.12 (b), the regions on the five edges of G are determined by different sites. The single color weighted Voronoi diagram WVD_G on graph G can be computed in $O(e + m \log m + \log n)$ time by using an incremental approach. The sites are sorted by their weights in increasing order. For the i th site, s_i , a single source Dijkstra on the graph is executed, until the distance function of s_i “hits” the distance function of another inserted site with lower weight. Refer to Figure 2.12 (c), sites r , p and q have been inserted before s is processed. This step can be done in $O(e + m \log m)$ time. Also, $O(\log n)$ is needed to locate s_i in G , and $O(e)$ time for the deletion of old bisectors while inserting s_i .

Therefore, for all colors, the weighted color Voronoi diagram $WCVD_G$ can be computed in overall time:

$$\sum_i (n_i(e + m \log m + \log n_i)) = O(en + n \log n + nm \log m).$$

On a single edge, the lower envelope of the weighted distance functions of color i is a piece-wise linear function, with complexity $O(n_i)$ as used above. There are totally $O(en)$ piece-wise linear functions in G . The lower envelope of k color piece-wise linear functions can be computed in $O(en\alpha(k) \log k)$ time, by Sharir and Agarwal [2]. The weighted furthest color Voronoi diagram $WFCVD_G$ is equivalent to the upper envelope of the k single color lower envelopes of weighted distance functions in WVD_G . Thus, the time complexity as stated in Theorem 2.2 is obtained.

2.2.2 Shortest paths among growing discs

This section introduces previous work on the problem of finding a shortest path among growing discs, presented in van den Berg and Overmars [115]. Refer to Figure 2.13 for

Figure 2.13: A shortest path from s to g among the set of growing discs. It is composed of two tangent segments and a spiral segment. At the time when the spiral reaches tangent point q , disc C becomes the dashed disc.

an example. We build and extend upon [115] in the development of our algorithms discussed in Chapter 5.

Van den Berg and Overmars presented an $O(n^3 \log n)$ algorithm for computing a shortest path from a point s to a point g in the plane amidst growing discs. The discs represent obstacles and grow at the same speed. A point robot moves from s to g along the collision-free shortest path at a steady speed. The set of growing discs is defined as $\mathcal{D} = \{C_1(o_1, r_1(t)), C_2(o_2, r_2(t)), \dots, C_n(o_n, r_n(t))\}$.

Since the discs grow at the same speed, they can be modelled by cones, whose apices are located at the centers of the discs, and open angles are 90° . If the cones are cut at a horizontal plane with time $t = T_0$, when the point robot starts from s ,

a set of discs $C_i(T_0) \in \mathcal{D}$ ($1 \leq i \leq n$) is obtained. It is shown that the shortest path from s to g for the robot is composed of straight line segments and logarithmic spiral segments. Each straight line segment, called tangent, is tangent to two cones. Each logarithmic spiral segment is on the surface of a cone. A graph G is constructed and then a shortest path from s to g is found by applying Dijkstra's shortest path algorithm on G .

The graph G is called a directed state transition graph by LaValle [66], which is a natural tool to solve path planning problems. Generally speaking, the initial state and goal set are designated as special vertices (nodes) in the graph. A directed edge from a node a to another node b exists in the graph G if and only if there exists an action from a to b . The directed state transition graph is revealed incrementally through the application of actions, instead of being fully specified in advance. Thus, searching for a shortest path from s to g among the growing discs in the plane can be interpreted as a search in G .

Let the growing speed of each obstacle disc be 1, and let the speed of the robot be V_R , $V_R > 1$. The *nodes* and *edges* of graph G are defined below. For two cones, theoretically, there are an infinite number of tangents between them with slope $\frac{1}{V_R}$. The endpoints of the tangents on the starting cone form a continuous curve called *departure curve*. Each departure curve is cut by $n - 1$ cones thus defining $O(n)$ intervals. Each interval is a *node* in the graph G , thus there are totally $O(n^2)$ intervals on a cone. For each interval, only the collision free path that arrives at the interval earliest is kept. An *edge* of the graph G is either a collision free tangent line segment between two cones, or a piece of spiral segment between two neighboring departure curves on a cone. Imagine that a piece of a spiral curve on cone C_1 starting from a point $p \in C_1$. With time advancing, the spiral will intersect the departure curves on C_1 one by one. A valid interval on a departure curve has two out edges, one goes to the next departure curve on C_1 , the other is a tangent line segment with an endpoint located on some cone C_i , possibly followed by a spiral curve on C_i , and reaches a departure curve interval on C_i . To compute the edges in G , a brute force algorithm is

used. The arrangement of the cones and the departure curves are computed, and then a trapezoidal map for the arrangement is built. The sides of a trapezoid are departure curves, and the top and bottom edges of the trapezoids are spiral segments. Also, by computing the arrangement of the cones, it can be tested whether there exists a collision free tangent path between two cones or not. Detecting the collision free spiral paths from each node w to its neighboring nodes in G which are located on the same cone, can be done by querying the trapezoidal map of the arrangement of the departure curves in $O(\log n)$ time for each node. This is done by performing a point location query in the trapezoidal map to locate the arriving point p , and then inside the corresponding trapezoid to compute which departure curve will be met by a spiral passing through p . There are $O(n^3)$ nodes and $O(n^3)$ edges in the graph G . Thus, by applying Dijkstra's shortest path algorithm, an $O(n^3 \log n)$ time complexity is obtained.

2.2.3 3SUM-hard problems

This section introduces 3SUM-hard problems presented by Gajentaan and Overmars [40]. There are many problems in computational geometry for which the best known algorithms take time $\Theta(n^2)$ in the worst case while the known lower bounds on the problems are substantially small. Among them, there are a large class of problems that are proved at least as difficult as the following 3SUM problem:

Definition 2.1. *Base problem — 3SUM problem: Given a set of n integers, are there three elements of S that sum up to 0?*

The base problem is called 3SUM-hard, as the best known algorithm for it takes $\Theta(n^2)$ time. Gajentaan and Overmars proved a class of problems to be 3SUM-hard problems in [40] by reducing the base problem to the set of problems.

Definition 2.2. *Given two problems PR1 and PR2, we say that PR1 is $f(n)$ -solvable using PR2 iff every instance of PR1 of size n can be solved using a constant number*

of instances of $PR2$ of at most linear size and $O(f(n))$ additional time. Denote this as $PR1 \lll_{f(n)} PR2$.

If $PR1 \lll_{f(n)} PR2$ and $PR2 \lll_{f(n)} PR1$, we say that $PR1$ and $PR2$ are $f(n)$ -equivalent, denoted as $PR1 =_{f(n)} PR2$.

Lemma 2.2. Let $PR1 \lll_{f(n)} PR2$. Let $f(n)$ and $g(n)$ be polynomials. If $PR2$ can be solved in $O(g(n))$ time and $f(n) = O(g(n))$, then $PR1$ can be solved in $O(g(n))$ time. Reverseely, if $\Omega(g(n))$ is a lower bound for $PR1$ and $f(n) = o(g(n))$, then, $\Omega(g(n))$ is also a lower bound for $PR2$.

Definition 2.3. A problem PR is called $3SUM$ -hard if and only if $3SUM$ is $f(n)$ -solvable using PR , where, $f(n) = o(n^2)$.

$3SUM$ problem can be reduced directly to the following $3SUM'$ problem:

$3SUM'$ problem: Given three sets of integers A , B and C of total size n , are there $a \in A$, $b \in B$ and $c \in C$ with $a + b = c$? $3SUM' =_n 3SUM$

$GEOMBASE$ problem: Given a set of n points with integer coordinates on three horizontal lines $y = 0$, $y = 1$, and $y = 2$, determine whether there exists a non-horizontal line containing three of the points. $GEOMBASE =_n 3SUM'$

Two steps are needed to reduce problem $PR1$ to $PR2$:

1. Construct instances from $PR1$ to $PR2$.
2. $PR1$ is satisfied if and only if $PR2$ is satisfied.

For example, to prove that $3SUM' \lll_n GEOMBASE$, for each element $a \in A$ create point $(a, 0)$ on line $y = 0$; for each element $b \in B$, create point $(b, 2)$ on $y = 2$, and create point $(c/2, 1)$ on $y = 1$ for each element $c \in C$. Then, $a + b = c$ if and only if $(a, 0)$, $(b, 2)$ and $(c/2, 1)$ are collinear. $GEOMBASE \lll_n 3SUM'$ is proved exactly in the reverse way. Refer to Figure 2.14.

The set of problems proved to be $3SUM$ -hard in [40] are: (1) incidence problems, e.g., given a set of lines in the plane, are there three that pass through the same point? (2) separator problems, e.g., given a set of (non-intersecting, axis-parallel)

Figure 2.14: An example of GEOMBASE.

line segments, is there a line that separates them into two non-empty subsets? (3) covering problems, e.g., given a set of (infinite) strips in the plane, do they fully cover a given rectangle? (4) visibility problems, e.g., given a set of axis-parallel line segments in the plane and one particular horizontal segment s , determine whether there is a point on s can be seen from ∞ ? and (5) motion planning problems, e.g., given a set of (non-intersecting, axis-parallel) line segment obstacles in the plane, and a rod, can the rod be moved, allowing translation and rotation, from a given source to a given destination without colliding with the obstacles?

Because the base problem can be reduced to all of these problems, none of the problems can be solved in $o(n^2)$ time unless a subquadratic solution exists for the base problem. However, this does not mean that all problems are equivalent, like NP -complete problem. It only says that the problems are at least as hard as the base problem. Also, being 3SUM-hard does not mean that a subquadratic solution is impossible. One better first try to solve the base problem faster.

Chapter 3

IMA System

This chapter introduces the IMA system. The architecture is introduced first, and then the details of the implementation are given. The architecture of IMA was designed by the IMA research group, and the prototype implementation of the framework was started by Hongsheng Liu. A more detailed description of the IMA system can be found in [43].

3.1 Intelligent map agents architecture

The objective of the IMA is to develop a framework which permits the proliferation of geographic data within our society through seamless connection, collaboration between users, and interaction with distributed geographic services and data centres. In Section 3.1.1, we give a high level overview of the IMA architecture. Then, in Section 3.1.2, we describe the IMA components. We address interaction and collaboration in Sections 3.1.3.

3.1.1 Architecture overview

The IMA architecture is user-centric and is designed to support the user interaction patterns. It is structured to serve the user community regardless of each user's location. Depending on the needs, it supports a variety of user activities (see Section

Figure 3.1: The IMA architecture is user-centric. As such the IMA is aimed at providing each user with personal set of functionality capabilities, interaction patterns and visualization.

3.2.2). Each user is represented by one or more intelligent map agents that can co-operate so that they meet user demands and/or take action upon requests. Figure 3.1 illustrates the IMA's user-centric architecture. These agents consult the user's personal contextual information in order to complete user tasks. The map agents draw on three types of resources and/or services:

- **User Domain**

The user is mobile and can employ a variety of mobile devices to interact with the surrounding environment. Because the user is always connected, the interaction is continuous. The map agents interact with the users mainly during collaboration where the agents attempt to coordinate work or complete a task that requires input from more than one user. An important aspect of the map agents is its ability to obtain information about the user with minimal or no interaction with the user. The user's contexts provide private physical and/or behavioural information about the user, e.g., driving habits, color blindness. This private data can be later on used in providing proper services to the user or when interacting with other map agents.

- **Functional Services**

The functionality services are a set of services which are ready to provide solutions to requests by an agent or conduct work on behalf of an agent. The functional services are not constrained by a number, or to a particular family, of functions. The aim is to augment the number and types of available services. The services are located in a variety of locations and are operating in a heterogeneous environment.

- **Data Services**

The data services are a distributed set of resources comprised mainly of data centres, which provide the intelligent map agents and thus users with access to required data. The data centres serve as a source for different data (spatial and

Figure 3.2: The IMA consists of three main tiers: user tier, services tier and data resources tier. Although the IMA permits each tier to interact with any other tier, it is envisioned that users will access the resource tier via the services tier. Note, that services can act as users and thus access other services to accomplish their task(s).

non spatial) as well as a repository for acquired or manipulated data. (Again, data centres may provide different data, e.g., of varying quality, accuracy, resolution, or access speed. The map agent will make the appropriate selection based on the need specified by the service and the interaction pattern/history of its user.)

A layered view of the IMA architecture is depicted in Figure 3.2. The data centres are the basis for all required activities and interactions. Therefore, they form the foundation of the hierarchy. The functionality services layers include the intelligent

map agents. Similarly to the data centres, the different services reside in a variety of physical locations. The top layer is the mobile user which interacts with any of the layers.

3.1.2 System components

The IMA project is an agent-based framework designed to support a large number of concurrent users and services. Each of the components plays a particular role within the framework. The main components as shown in Figure 3.3-3.4 are described next:

- **User**

The user is any entity requiring spatial services from the IMA. Thus, a user can either be a human or a software system. It is envisioned that, in most cases, users will be humans. A human user is characterized to be mobile and to carry a small device capable of (wireless) communication with its environment.

- **IMA Personal Agent (PA)**

The IMA personal agent operates on behalf of the user in order to accomplish a task as requested by the user. The personal agent acts as a liaison between the user and the services. This frees the user to work on other tasks. As a result each user may have instantiated several concurrent personal agents.

- **Context Manager (CM)**

The context manager represents private and intimate knowledge about the user. The context manager allows the IMA personal agent to obtain information about the user (e.g., the user's driving habits) and to direct services (e.g., a shortest path service) on how to compute a shortest path using the user's preferences. Context awareness is not dealt with in the current IMA system.

- **Directory Facility Broker (DFB)**

The Directory Facility Broker (DFB) is at the heart of the IMA services. It contains all required information about available services and how to reach

Figure 3.3: The figure illustrates the main components of the IMA and gives a possible snapshot of the IMA during operation. The double ended arrows represent communication between components (a Service Access Point, a Service Worker, and a Resource Provider are represented by SAP, SW and RP respectively).

Figure 3.4: The figure shows an example of an invocation sequence of service and interaction between the components. The solid double ended arrows represent ongoing communication between components while dashed double ended arrows represent temporary invocation which occurs during invocation of the different services and applications (a Service Access Point, a Service Worker, and a Resource Provider are represented by SAP, SW and RP respectively).

them. The DFB can be viewed as a yellow page book listing all services available to IMA users and describing how to evoke those. Each new service registers itself with the DFB and by doing so advertises its availability to potential and existing users.

- **Service Access Point (SAP)**

A Service Access Point (SAP) represents a service that is available in the IMA. The SAP of a service acts as an initial mediator between a personal user agent and a requested service. As such, once a request for a particular service is sent to the DFB, the DFB invokes a SAP. The SAP negotiates with the user's personal agent and, upon receiving the required information, it invokes a service worker to complete the task.

- **Service Worker (SW)**

A Service Worker (SW) provides the requested functionality for which the service has been invoked. A SW is invoked by the corresponding SAP to provide a service to a particular user. As a result, there can be several SWs for each registered service.

- **Resource Provider (RP)**

A Resource Provider (RP) provides access to resources within the environment. The resources can vary between one another. In most cases we envision that the resources will be data centres which contain geographic data - vector data, and raster data.

Not all of the main components of the IMA, which are depicted in Figure 3.3, exist in the system at all times. Some of the components are invoked as and when required. We demonstrate this concept by presenting an invocation sequence as well as a communication interaction between components. In this scenario, *User₁* would like to meet *User₂* in the office of *User₂*. We present the interaction and the invocation in a sequence of messages in Figure 3.4. Each of the steps is numbered on the

arrows. Dashed double arrows represent temporary communication which is used during invocation, while solid double arrows represent communication lines between the components.

- 1 $User_1$ instructs its Personal Agent (PA_1) to find the location for the office of $User_2$ and how to reach it.
- 2 PA_1 contacts the Directory Facility Broker (DFB) to find a service which can provide a route between two addresses.
- 3 The DFB finds a service (shortest path service) and returns a “link” to the Service Access Point (SAP_1) of the shortest path service.
- 4 PA_1 contacts this SAP_1 requesting services from it.
- 5 SAP_1 creates a Service Worker (SW_1) which will provide a shortest path service to PA_1 .
- 6 SW_1 contacts PA_1 and indicates that it is ready to provide its services.
- 7 PA_1 contacts PA_2 (the personal agent of $User_2$) to determine the office location of $User_2$.
- 8 PA_2 sends the office location of $User_2$ to PA_1 .
- 9 PA_1 contacts the Context Manager of $User_1$ (CM_1) to obtain the driving habits of $User_1$.
- 10 CM_1 provides the driving habits of $User_1$ to PA_1 .
- 11 PA_1 forwards the locations of $User_1$ and $User_2$ as well as the driving habits of $User_1$ to SW_1 .
- 12 SW_1 contacts the Resource Provider (RP_1) in order to retrieve the required data.
- 13 RP_1 returns the required data.

14 SW_1 computes the required route between $User_1$ and the office of $User_2$ taking into consideration the driving habits of $User_1$ and sends the route to PA_1 .

15 PA_1 sends the route to $User_1$ who can now travel to meet $User_2$.

The IMA architecture is aimed at supporting a large number of concurrent users and services which are mobile or are distributed across a large geographical region. Therefore, it is designed to operate in a distributed environment which addresses local as well as global needs. In order to accomplish distributed operations the IMA is designed as a federated environment. Each cluster of the IMA, as depicted in Figure 3.3, consists of a large number of components (concurrent users, concurrent services, etc.) all of which are registered with a particular DFB. The federated clusters of the IMA are interconnected via the DFBs. Namely, the DFBs are connected to each other as seen in Figure 3.5. Thus, when a service is requested by a user, all interconnected DFBs are being searched until the desired service is obtained.

3.1.3 Interaction and collaboration

Interactions and/or collaborations between users are important and integral parts of the IMA architecture. Interaction occurs when one user requests information from another user to complete a particular task. For example, in delivery and courier services, courier “John”, may contact courier “Jack” to find a phone number of a customer. Jack is not impacted by the request made by John. Each of them works independently in order to complete the overall task of the courier company (which is to deliver all parcels). The IMA is aimed at supporting a higher level of interaction - *interdependence collaboration*. In interdependence collaboration, users must work together and coordinate their activities in order to accomplish the task. In this form of collaboration, all users synchronize their efforts and activities, otherwise they will fail. In such collaboration the usage of user context is at times essential for the success of the collaboration. The context may vary from private user context to location based context. For example, in Search and Rescue (S&R), the S&R parties must coordinate

Figure 3.5: The IMA can operate in a distributed environment. Although, users are handled by a single IMA cluster (depicted in Figure 3.3), they have access to all services via a federated network of IMA clusters. The figure shows an example of a three federated IMA clusters which are located across Canada (Halifax, Ottawa, and Vancouver). The three clusters are connected to each other via the IMA DFB. Note that the services registered at the various locations would typically be different.

their efforts. Otherwise, they may not know what other parties are doing and thus search regions that were already covered, or worse, miss some regions entirely. Other examples are forest fire fighting, epidemic fighting, and traffic management.

3.2 IMA implementation and applications

This section presents the current state of the IMA. It describes the implementation of the system and the currently available services.

3.2.1 IMA implementation

The IMA project is based on the agent framework JADE/LEAP. JADE simplifies the development of multi-agent applications that comply with the latest FIPA 2000 specifications. JADE is implemented in Java and has no further dependency on third-party software. A JADE/LEAP platform consists of a single main container and zero or more peripheral containers. A container is the run-time environment in which agents operate. While appearing as a single entity to the outside world, a JADE agent platform can be distributed over several hosts. The main container provides life cycle management for agents in the platform, as well as yellow pages services that allow agents to search for other agents and to inspect the services that they offer. All containers share a distributed, TCP/IP-based, Agent Communication Channel (ACC). Agents communicate by exchanging messages in the Agent Communication Language (ACL). Agents can also migrate or clone themselves to other hosts of the platform, regardless of the OS.

The implementation of the IMA assumes that its components are deployed on heterogeneous physical devices in terms of capabilities and that all devices are able to communicate via wired or wireless networks. The physical devices are tailored to the needs of each of the different tiers (see Figure 3.2). The devices of the user tier are assumed to be PDAs or cellular phone with weak computation power and low communication bandwidth (though the system is designed to also handle other devices including e.g., PCs). A Java/LEAP runtime environment is used to provide

an interface protocol with the functional services tier. Since the main purpose of the components in this tier is to interact with the user, the components and the graphical user interface were implemented in Java to enable easy migration from one platform to another.

The functional services tier is the tier where services can be searched and/or invoked by users, as well as executed in order to perform a requested task. In this tier, the IMA employs a federated clusters of the JADE's containers. The IMA uses JADE's directory facilitator as the yellow pages tool (DFB) for users searching for services. The services, in this tier, do not have to be deployed on an IMA cluster (or a host) or in the JADE's main container. When a service is ready to provide services to users it registers itself with the DFB. When a service is located on a remote platform, it is configured for that platform. Upon termination, a service agent de-registers from the DFB. In the IMA implementation we also use the federated DFB as a white pages tool to find other users in the IMA. Thus, when an IMA PA agent starts, it registers itself with the DFB.

A Service Access Point (SAP) is used for service registration and service access while a Service Worker (SW) provides the actual service to the user. Each SAP corresponds to one service/application, and it registers with the JADE DFB. Different linking schemes between a user and a service worker were used in the IMA implementation. The SAP either creates a dynamic service worker, which terminates once the task is completed, or points to a static service worker. A service worker can either work for one user, e.g., the dynamic map agent, or for a group of users, e.g., the meeting scheduler agent. A service worker may need, in order to complete its task, computational back-end resources from several resource providers. Depending on the program languages used on the back-end servers, a variety of communication technologies such as RMI, CORBA, JNI, or IIOP can be used.

Scalability is an essential element of the IMA and thus was addressed in the implementation in two ways: (i) Workload monitoring - here the workload is being continuously monitored and when a SAP is about to invoked a new service worker or

to assign work to an existing service, it examines the work-load and then determines where to new server worker should be invoked or to which service the work should be redirected. For example, it can launch a new dynamic service worker on a machine with a small load or on one machine with more resources such as memory. (ii) Concurrent query execution - here the IMA static service provide services to multiple queries concurrently where each query is serviced by a powerful machine.

3.2.2 Services

The architecture of the IMA has been designed as a service-oriented architecture. It is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

The current IMA system provides several services:

1. **Shortest path in a road network** The shortest path service is a prototype for a variety of network related algorithms. Currently, in the IMA it provides basic measures of shortest cost paths where the cost can vary from distance to travel time.

2. **Meeting Scheduler**

The meeting scheduler is a service which explores different ways for coordinating a meeting between users see e.g., [62]. The service determines time, duration and location for a new meeting consulting the currently scheduled meetings (including time and location) of each user participating in the meeting.

3. **Personalized navigation**

This service addresses the preferences of user navigation. It provides the user with several navigational options. The navigational options provided by the service allow the user to choose between different map presentations schemes which best suites the user at that time [44].

Figure 3.6: The figure shows a set up of a meeting scheduling application. The set up mimics a real-life distributed users and services which are registered with the IMA clusters (locations are across Canada). The arrows between the components represent the registration and knowledge relationships between the components.

Next we present a sample application that we developed to test and validate our approach. The application chosen is a meeting scheduling application. This application schedules a new meeting among several users and finds the users' paths to the meeting location. The meeting application invokes an IMA service: meeting scheduler. The meeting scheduler, in turn acting as an IMA agent, invokes a shortest path service. In this example, we simulated a setting in which the services are located across Canada (see Figure 3.6). Two IMA clusters are used. Each cluster is deployed on a SUN server/Unix (one representing an IMA cluster in Ottawa and one in Halifax). Next we used a meeting scheduler service which resides in Toronto. The meeting scheduler is deployed on an Intel/Windows platform and it is registered with the IMA cluster in Ottawa. A shortest path service is deployed on a parallel computer (Beowulf computer/Linux platform) and it simulates a service in St. John's Newfoundland. The shortest path service is registered with the IMA cluster, in Halifax. The purpose of using mixed platforms is to test and demonstrate that the IMA can operate and support a heterogeneous environment. Lastly, we employed three users for this application. The users are either mobile (using wireless communication) or stationary users (using wired communication). Users John and Jack are connected to the IMA cluster in Ottawa, while user Linda is connected to the IMA cluster in Halifax.

The snapshot in Figure 3.7 shows the state of the IMA after the users, and the services have registered with the IMA system. Note that corresponding agents have registered with the JADE platforms (local and remote). The service provider of the meeting scheduler is denoted as schedulerAP, which is the service access point.

Linda, who is stationed in Halifax decided that she would like to meet with John and Jack during her visit to Ottawa. Therefore, she uses the IMA to coordinate a meeting with her two colleagues. Linda is required to complete the information about the meeting (see Figure 3.8) including meeting duration, date and invited persons. Once Linda had entered the information about the new meeting, Linda's agent in the IMA agents is taking over and schedules the meeting. The interaction between the

Figure 3.7: This snapshot shows the status of the users and services after the users and services have registered with the IMA system.

Set New Meeting

subject
Preparing for the workshop in Ottawa

duration
1

region
Ottawa

Starting On

July 2006						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Ending With

July 2006						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Known Persons

- (agent-identifier :name Jack...)
- (agent-identifier :name John...)
- (agent-identifier :name Linda...)

Invited Persons

- (agent-identifier :name Jack...)
- (agent-identifier :name John...)
- (agent-identifier :name Linda...)

Ok Exit

Figure 3.8: Linda, who is the meeting initiator, sets up the meeting subject, duration, location, date, and invites persons for the new meeting.

Figure 3.9: The figure presents an interaction diagram between the different components of the IMA. For brevity we omitted the interaction with the SAPs.

Figure 3.10: The figure presents the dynamic agents representing users and services that are produced during the processing of a service in the IMA system. After the service requests have been fulfilled, the dynamic agents will be deleted by the IMA.

Figure 3.11: Each user receives a map with the new meeting location and two shortest paths. The first path, shown in red, is the path from the current meeting to the new meeting location. The second path, shown in blue, is the path from the new meeting location to the next meeting location. (Note, the blue path travels along the river until it can cross the Rideau river at a bridge.)

different IMA components and users is depicted in Figure 3.9. For brevity we omitted the interaction step with the service access points (SAPs).

The snapshot in Figure 3.10 shows the state of the IMA system during the operation of scheduling the meeting between Linda and her colleagues. The dynamic agents representing Linda, John, Jack and meeting scheduler service during the scheduling are shown (e.g., Linda's agent is: `Linda.Meeting.scheduler@134.117.27.195:1099/JADE`, and the scheduler is: `scheduler0@134.117.27.193:2099/JADE`). Note that the life cycle of these dynamic agents is short. Once, they completed the task of scheduling the meeting they will terminate and only the agents representing Linda, John, Jack and the service will remain active.

Once the meeting scheduler completed its work, each of the users is presented with the relevant information about the schedule. A new meeting with subject, time and date is entered in each participating user schedule. In addition, each user receives a map depicting the meeting location and two paths (Figure 3.11): (i) a path to the meeting location, and, (ii) a path from the meeting location.

3.3 Conclusions and future directions

After a two/three decade-long flurry of activities in designing and developing diverse approaches to storing, handling and manipulating of geographic and spatial data, in recent years some convergence, consolidation of GIS development and systems can be witnessed. The emergence of mobile devices, technologies and communication infrastructure as well as dramatically altered behaviour, expectations and demands by users are again posing new and exciting challenges for GIS developers. The work described in this chapter is intended as our contribution to identifying and addressing some of these challenges. Here, we have identified, stated and analyzed factors driving system design and development; these are categorized by users, services and user community/interaction. We captured/addressed these factors in our system design. We proposed, designed, developed and discussed our prototype system, its components and features including context handling, interaction and collaboration. To test and

validate our approach, we have designed and demonstrated a shortest path service and a collaborative meeting scheduler. Representing users via agents appears to be vital to free users from having to carry out unnecessary tasks, allowing users to function effectively and efficiently and to increase system reliability. Furthermore, it removes some of the difficulties encountered by devices having varying communication and computation capabilities. JADE/LEAP is a reasonable choice for an agent platform. The yellow-page like approach (Directory Facility Broker) for services is somewhat restrictive in JADE, but provides a workable solution. Providing tailored responses to individual users through usage and managing of user context could well prove to be a key factor in determining user acceptance of mobile GIS.

We believe that our approach provides a sound basis for future work in this area. Further work requires expanding system functionality, context storing and handling, security and privacy, and service offering. Subsequently, exhaustive testing of the system, its new services, quality of services and scalability issues is required.

Chapter 4

Algorithms on Meeting Scheduling

This chapter focuses on geometric algorithms for meeting scheduling problems. We aim at finding a new meeting location with the longest meeting duration among all the participants based on the time and spatial constraints. If no new meeting can be scheduled, then we compute a new meeting location with maximal participants instead.

4.1 Introduction

Consider the following problem: A set of persons, called participants, would like to schedule a meeting. The participants are located at different sites and would like to find a common point for the meeting to be held. These sites are rather far from each other, so that the travel times to the meeting point are important. In a geometric model, travel distances between locations can be measured using the Euclidean distance in the plane. In a graph-based model one could assume that travellers use a transportation network that combines road, railway, and air line connections.

Participants have individual schedules which specify: the earliest possible time they can leave their current location and the latest possible time they can arrive at their next location. We wish to solve the problem of finding all meeting points so that the time available for all participants to meet is maximized.

Our general objective for this research is to derive efficient algorithms for solving general meeting scheduling problems, to find approximate solutions, where appropriate, to implement our solutions, and to integrate them into applications that allow users who are connected over a network to schedule meetings. Most current meeting scheduling systems take into consideration only time and not location/geometry, see e.g., [17, 48, 107, 109]. There are projects or algorithms other than meeting scheduling dealing with both spatial and temporal information [4, 24, 45, 47]. The systems in [28, 54] apply distributed meeting scheduling algorithms. Rasmussen and Trick [100] define the timetable constrained distance minimization problem, which is a sports scheduling problem applicable for tournaments where the total travel distance must be minimized. They use integer programming and constraint programming formulation for the problem. Integer programming is also applied by Wang et al. [116] for scheduling meeting among students and teachers in universities. For another scheduling approach to school meetings (between teachers and parents), Rinaldi et al. [101] set weights and build directed graphs based on the time slots, then they find shortest paths in the resulting graphs.

Some meeting scheduling systems allow as input a list of possible meeting locations and they then select, from that list, one location based on different optimization criteria (such as time to meeting or cost of meeting). Examples of such work include: [5, 103], and our intelligent map agent system (IMA) [43] introduced in Chapter 3 which contains a service for this meeting scheduling problem. However, The algorithms used in this work ([5, 43, 103]) are relatively simple; they repeatedly execute a minimum cost algorithm for each of the preselected locations and then selects the best option. Here, we are dealing with the more complex problem instance, where we are not given preselected locations, but rather have to compute an optimal location which could be anywhere on the plane.

4.2 Organization

This chapter is organized as follows. In Section 4.3, we give a formal definition of the general meeting scheduling problem. In Section 4.4, we assume that a new meeting can always be scheduled for the set of participants. The participants can travel to their meeting location along straight paths in the Euclidean plane at the same speed. Two algorithms for finding a new meeting location with longest duration are presented. One algorithm is purely geometric, and the other one is applying linear programming. The geometric approach is studied in Section 4.4.1. We analyze the structure of the problem, and show how to use tools from computational geometry in order to efficiently compute an optimal solution. In Section 4.4.2, the solution via LP-type programming is sketched ¹. It uses the concept of LP-type problems and leads to a randomized algorithm with expected running time $O(n)$. The solution to the case where the participants travel at different speeds is illustrated at the end of this section. Section 4.5 assumes that no new meeting can be scheduled for the set of participants at the same speed. It presents a solution to the following problem: how to compute a new meeting location l , such that the maximum number of participants can join in the meeting at l ? In Section 4.6, a graph-based model is considered, where connections are provided by some geometric network in the plane. Participants have individual speeds, and participants come in groups subject to the condition that only one member of each group must attend the meeting. The techniques applied in this part build on work by Klein et al. [49]. As an application, this meeting scheduler problem is solved in the graph-based model. This section also assumes that a new meeting can be scheduled for all the participants. Finally, we mention some directions for further research in Section 4.7.

¹Work on the meeting scheduler problem was carried in external collaboration with R. Klein and F. Berger, University of Bonn. Their main contribution was on the graph model and LP-type programming (the latter one was published in [18]).

4.3 General problem definition

Assume that each participant's meeting schedule MS is represented as a sequence of meetings $(M_1, M_2, \dots, M_k)$. Each meeting M_j , $1 \leq j \leq k$, is represented as a triple $M_j(l, \bar{t}, \bar{\bar{t}})$, where $l \in \mathbb{R}^2$ is the meeting location, $\bar{t}$ and $\bar{\bar{t}}$ denote the starting and ending times, respectively, for the meeting. The duration of the meeting M_j is $\bar{\bar{t}} - \bar{t} > 0$. Define $M_j(f) = f$, for each $f \in \{l, \bar{t}, \bar{\bar{t}}\}$. If no ambiguity arises, M_j is used instead of $M_j(l, \bar{t}, \bar{\bar{t}})$. Travel distances between meeting locations l and l' are measured by some distance measure $d(l, l')$, and s is the individual travelling speed of the given participant.

Observe that individual speeds can be used to model the importance of the participants. In fact, a low speed may be assigned to a participant of high status, in order to ensure that his/her way to the meeting point will be short.

A participant's meeting schedule $(M_1, M_2, \dots, M_k)$ is *valid* if it satisfies the following condition for all j , where $1 \leq j < k$:

$$M_j(\bar{\bar{t}}) + \frac{d(M_j(l), M_{j+1}(l))}{s} \leq M_{j+1}(\bar{t}).$$

I.e., the sum of the previous meeting ending time and the travel time spent from the previous meeting location to the subsequent meeting location should not exceed the subsequent meeting starting time. Now consider a set of participants $\{P_1, P_2, \dots, P_n\}$; each P_i , $1 \leq i \leq n$, has an associated meeting schedule M_i . The participant set has a *valid meeting schedule* if each M_i , $1 \leq i \leq n$, is valid.

To *schedule a meeting* for participants $\{P_1, P_2, \dots, P_n\}$ means to find a triple $M^*(l, \bar{t}, \bar{\bar{t}})$ so that after insertion of M^* into each M_i , a valid meeting schedule for the entire set of participants is obtained. To *schedule a meeting $M^*(l, \bar{t}, \bar{\bar{t}})$ of longest duration* means that for any other scheduled meeting $\bar{M}(l, \bar{t}, \bar{\bar{t}})$, $M^*(\bar{\bar{t}}) - M^*(\bar{t}) \geq \bar{M}(\bar{\bar{t}}) - \bar{M}(\bar{t})$.

4.4 Euclidean meeting scheduling

In this section we consider the problem of fitting another meeting into the schedules of participants whose calendars are already rather full. Not only does this problem frequently occur, its practical solution can be rather cumbersome if more than very few people are involved. We study a situation where all participants can travel at the same, constant speed s in the plane (various speed case is briefly sketched in the LP solution in Section 4.4.2.2). Since the precise value of s is not critical, we may as well assume that $s = 1$ holds. Then, the time it takes to travel from location l to location l' is given by the Euclidean distance $||l - l'||$. (In Section 4.6 graph-based models are studied that allow for individual speeds and for more realistic distance measures.)

We can simplify the general notations given in the previous section in the following way. From each participant's schedule, we compute the intervals when the participant is free. Then, from those intervals, we find common intervals that are free for all the participants. In this chapter, we focus on one common free interval. In each participant's schedule, the meeting before this common free interval is called the previous meeting, and the meeting after the interval is called the subsequent meeting. Let the i th participant P_i has the previous meeting at location $l_i^{\text{pre}} = M_1^i(l)$ that finishes at time $t_i^{\text{pre}} = M_1^i(\bar{t})$. His/her subsequent meeting starts at time $t_i^{\text{sub}} = M_2^i(\bar{t})$ at location $l_i^{\text{sub}} = M_2^i(l)$.

We want to schedule a meeting of longest possible duration in between the previous and subsequent meetings of all participants (or report that no such meeting is possible, under the time constraints given). To solve our problem, we have to find a location l such that the time interval between the arrival of the last participants from their previous meetings at time

$$t_1 = \max_{1 \leq i \leq n} (t_i^{\text{pre}} + ||l_i^{\text{pre}} - l||) \quad (4.1)$$

and the departure of the first participants to their subsequent meetings at time

$$t_2 = \min_{1 \leq i \leq n} (t_i^{\text{sub}} - ||l_i^{\text{sub}} - l||) \quad (4.2)$$

is maximized.

The possible locations participant P_i can reach, once his/her previous meeting has ended, are given by a system of discs, where each disc is centered at l_i^{pre} and starts expanding at time t_i^{pre} . It is quite natural to introduce time T as a third coordinate, in addition to the spatial coordinates X, Y that describe the location $l = (l_X, l_Y)$.

The expanding disc then forms a vertical cone PRE_i of opening angle² 90° whose bottommost point is its apex $(l_{i,X}^{\text{pre}}, l_{i,Y}^{\text{pre}}, t_i^{\text{pre}})$.

Similarly, the points in XYT -space from which participant P_i can reach in time his/her next meeting are given by the vertical cone SUB_i whose uppermost point is its apex at $(l_{i,X}^{\text{sub}}, l_{i,Y}^{\text{sub}}, t_i^{\text{sub}})$; this can be verified easily by imagining time running backwards. Refer to Figure 4.1.

Let

$$\text{PRE} := \bigcap_{i=1}^n \text{PRE}_i \quad \text{and} \quad \text{SUB} := \bigcap_{i=1}^n \text{SUB}_i.$$

The intersection

$$F := \text{PRE} \cap \text{SUB}$$

consists of all points (l_X, l_Y, t) such that all participants can meet at location l at time t . Hence, we call F the set of *feasible points*. Observe that F might be empty, if time constraints do not permit to squeeze in another meeting (the case of empty F is processed in Section 4.5). Otherwise, being an intersection of cones, F is a convex set whose boundary consists of conic surface patches.

A meeting at l is possible from time t_1 to t_2 if, and only if, the vertical line segment with endpoints (l_X, l_Y, t_1) and (l_X, l_Y, t_2) is fully contained in the set F (refer to the vertical line in Figure 4.1 (a)). Consequently, the solution to our problem is given by the *vertical line segment of maximum length that is contained in F* .

We would like to mention that the usage of cones, that has led to such a nice geometric formulation, is a well-known technique in computational geometry; see, e. g., Aurenhammer and Klein [16]. Cones have independently been used by Homsby and

²For an arbitrary travelling speed s , the cone's angle equals $2 \arctan s$.

Figure 4.1: (a): Two vertical cones $PRE_{1,2}$ of opening angle 90° with the projection of apex on the XY plane $l_{1,2}^{pre}$. $PRE = PRE_1 \cap PRE_2$. A vertical line through l intersects PRE_1 and PRE_2 . (b): Two vertical cones $SUB_{1,2}$ of opening angle 90° with the projection of apex on the XY plane $l_{1,2}^{sub}$. $SUB = SUB_1 \cap SUB_2$. (c) An example of three cones intersect.

Egenhofer [47] to model movement of objects or individuals over multiple granularities.

In the following, we present the geometric solution first, and then, is the LP solution.

4.4.1 Geometric model

Our main result in this section is an algorithm that does this in time $O(n \log n)$ and uses only basic tools from computational geometry, so that it could be implemented in a practical setting.

Let us imagine the discs that start expanding from the previous meeting locations l_i^{pre} at time t_i^{pre} . After a while, each point of the plane is covered by all expanding discs. The first point, l' , covered by all expanding discs at some time t' is the location of the *earliest meeting possible*. Clearly, $m' := (l'_X, l'_Y, t')$ is the bottommost point of the convex set PRE.

Similarly, let l'' denote the last point covered by all discs shrinking towards their centers l_i^{sub} , and let t'' be the latest time where l'' is still covered. Then l'' is the location of the *latest meeting possible*, that must be over at time t'' , and $m'' := (l''_X, l''_Y, t'')$ is the topmost point of the set SUB.

If the intersection F of PRE with SUB is not empty then it contains both points, m' and m'' , because all cones have identical slopes. Thus, our problem is nicely solved if $l' = l''$ holds: the meeting at l' from t' to t'' is the longest one the participants can have.

But in general, neither m' nor m'' needs to be an endpoint of the longest vertical line segment contained in F we are interested in. That is, neither l' nor l'' needs to be an optimum meeting place.

4.4.1.1 Applying the furthest Voronoi diagrams of discs

In order to compute the longest vertical line segment contained in F we shall make use of the Voronoi diagram introduced in Section 2.2.1.

For a potential meeting location, l , it is most interesting to know which participant P_j would be the last to arrive there (because the meeting cannot start before). If we draw a vertical line through l from $T = +\infty$ downwards, the first cone intersected by this line belongs to P_j . For example in Figure 4.1 (a), the boundary of the intersection region PRE is composed of two patches from PRE_1 and PRE_2 , respectively, separated by a hyperbolic curve. PRE_2 is further from l than PRE_1 . More generally, imagine that cone PRE_i is opaquely colored by color i . If we look from $T = +\infty$ downwards on the arrangement of cones PRE_i , we see the decomposition of the boundary of the set PRE into multiply colored surface patches. For all locations l vertically below a patch of color j , participant P_j would be the last to arrive at l .

The projection onto the XY -plane of this patch decomposition is a furthest site Voronoi diagram of discs, see the details in Section 2.2.1.4. The set of discs C_j are obtained by intersecting each cone PRE_j with plane H that is parallel to the XY -plane. Refer to Figure 2.6, participant P_j is the last one to arrive at l if, and only if, C_j is the disc furthest away from z , where z is the projection of l onto H . After projecting all discs C_j onto the XY -plane, the disc furthest from l corresponds to the participant arriving last.

So, let us denote by V^{pre} the furthest site Voronoi diagram we have obtained by projecting the patched boundary of PRE downwards onto the XY -plane. Refer to Figure 2.3 (b). According to Theorem 2.1 in Section 2.2.1, the location l' of the earliest possible meeting is either a Voronoi vertex or an interior Voronoi edge point of V^{pre} , since at least two participants will be the last to arrive there. That all regions in the furthest Voronoi diagram of discs are unbounded is a consequence of the following stronger observation.

Lemma 4.1. *Let l be a point in the region of disc C_i with center o_i in the furthest site Voronoi diagram V^{pre} . Then a ray ρ starting in l with direction $\overrightarrow{o_i l}$ is fully contained in the region of C_i .*

Proof. Let w denote the point of C_i closest to l . Since l belongs to the region of C_i , the disc centered at l that passes through w contains, or intersects, all other discs C_j .

Figure 4.2: Illustrating the proof of Lemma 4.1.

Then the same must hold for each point p on ray ρ , as the radius of the disc centered at p is bigger than the one of the disc centered at l , and w is located on both discs, the disc centered at l is enclosed by the disc centered at p . Thus, ρ is contained in the region of C_i . (Refer to Figure 4.2.) $\square$

After computing V^{pre} in time $O(n \log n)$, by Theorem 2.1, the point m' of lowest T -coordinate in PRE can be easily determined. To this end, we simply visit each Voronoi region (resp. edge, and vertex) of V^{pre} and compute the lowest point of the surface patch (resp. edge, and vertex) in PRE that lies above it. Since each of these features is of constant algebraic degree, this can be achieved in time proportional to the number of regions, edges, and vertices, i. e., in time $O(n)$. Observe that we could even construct the boundary of the set PRE in XYT -space in linear time, by simply gluing together all features visited; but there is no need to do so. We can also apply Rappaport's algorithm introduced in Section 2.2.1.4 to compute m' .

Obviously, the same arguments apply to the furthest Voronoi diagram V^{sub} associated with the cones $\text{SUB}_i, 1 \leq i \leq n$. Now we are ready to compute a meeting of longest possible duration.

4.4.1.2 Computing the longest vertical segment

The simplest approach to computing the longest vertical line segment in F would be to superimpose the projections V^{pre} and V^{sub} of PRE and SUB, respectively, and to inspect each of the sub-faces of the resulting decomposition for the longest vertical segment. Unfortunately, there could be a quadratic number of them. To keep our algorithm efficient, we shall restrict our search by applying the sweep technique.

Imagine an XY -parallel plane H' sweeping upwards through the set PRE, starting from its bottommost point m' . The intersection

$$W' := H' \cap \text{PRE}$$

is a convex shape bounded by circular arcs. Let S' denote the projection of W' onto the XY -plane. Within the Voronoi region of disc C_i in the furthest site Voronoi diagram V^{pre} , shape S' consists of a circular arc concentric with C_i ; this is the disc expanding from the participant that would arrive last in this region.

As plane H' moves upwards, S' expands, from l' , over V^{pre} , encircling a subtree at each time. Refer to Figure 4.3. Similarly, we have a convex shape

$$W'' := H'' \cap \text{SUB}$$

resulting from a top-down sweep of SUB, whose projection S'' expands from l'' over the diagram V^{sub} .

Now let us assume that the longest possible meeting corresponds to the vertical line segment with endpoints $m_1 = (l_X, l_Y, t_1)$ on PRE and $m_2 = (l_X, l_Y, t_2)$ on SUB. At time t_1 , shape W' passes through m_1 , while W'' passes m_2 at time t_2 . With W' and W'' in this configuration the following lemma holds.

Lemma 4.2. *The sets S' and S'' have a common touching point at the optimum meeting location l .*

Proof. Suppose that the boundaries of S' and S'' crossed each other at l (instead of merely touching). S' is obtained at time $\hat{t}$, and S'' is obtained at time $\hat{t}'$. Then before

Figure 4.3: Shape S' expanding over the furthest site Voronoi diagram. In the Voronoi cell V_{C_5} of the disc C_5 , the arc on the boundary of S' belongs to the dashed disc expanded from disc C_5 .

S' and S'' cross, they must have a common touching point. This can only happen when S' and S'' have smaller shapes. I.e., when S' is obtained at time $t_l < \hat{t}$, and S'' is obtained at time $t'_l > \hat{t}$, S' and S'' have a common touching point p . Thus, a longer vertical segment with length $t'_l - t_l > \hat{t} - \hat{t}$ at point p can be found in F , which leads a contradiction. $\square$

Since the optimum meeting location l is a point where S' and S'' touch, we can conceptually proceed as follows. We move both planes H' and H'' upwards, so that W' expands and W'' shrinks, in such a way that S' and S'' always touch in a point l . For each l , determine the length of the vertical line segment L over l in F , and report the maximum segment thus obtained.

Fortunately, we need not test every touch point, l . As long as l lies within the same circular arcs of S' and S'' , respectively, the endpoints of L are determined by the same cones PRE_i and SUB_k , whose center points l_i^{pre} and l_k^{sub} are co-linear with l , so that the length of L does not change.

Similarly, as long as l is a common endpoint of two neighboring arcs of S' and an interior arc point of S'' , the lower endpoint of L moves along the intersection of two cones PRE_i and PRE_j , that lies over a Voronoi edge of V^{pre} , while the upper endpoint of L is determined by some cone SUB_k . Since the intersection curve is of constant algebraic degree, we can compute in constant time the maximum length attained by segment L over this period.

The simultaneous motion of the planes H', H'' can be split into *phases* during which the status (relative location) of touch point l does not change on either set S', S'' . I.e., l is either (a) an interior arc point, or (b) a common end point of two arcs, or (c) a common intersection points of four arcs, two of which belong to S' , and the other two belong to S'' . Case (c) happens when l passes through a point where two Voronoi edges of V^{pre} and V^{sub} cross. For each phase we can, in constant computing time, determine the maximum length of vertical segment L . The following statement is crucial.

Lemma 4.3. *There are only $O(n)$ many phases.*

Proof. There are three types of events that can cause the status of touch point l to change. A *type 1 event* happens when a circular arc appears on, or disappears from the boundary of one of the sets S', S'' . This occurs whenever the boundary of a set passes through a Voronoi vertex of V^{pre} or V^{sub} . Since both sets move in a monotonic way, each vertex gets passed only once. By Lemma 2.1, there are only $O(n)$ vertices.

A *type 2 event* occurs when l moves to the endpoint of its circular arc, or vice versa. Refer to Figure 4.4 (a). More specifically, suppose that l was a common endpoint of the arc of cone PRE_i with the neighboring arc of cone PRE_j in S' , and has just moved to the interior part of the arc of cone PRE_i . Assume that l is an interior point of the arc of cone SUB_k in S'' .

Since the arcs of PRE_i and SUB_k have a common tangent at l , the points l_i^{pre}, l, l_k^{sub} are collinear. The point l lies in the Voronoi region of SUB_k in V^{sub} . By Lemma 4.1, the center point l_i^{pre} of PRE_i is also contained in this Voronoi region.

Thus, the event in l is caused by a triplet of cones (PRE_i, PRE_j, SUB_k) , where

1. PRE_i, PRE_j have a common Voronoi edge in V^{pre} that intersects the line segment $l_i^{pre}l_k^{sub}$ in l , and
2. l_i^{pre} lies in the furthest site Voronoi region of SUB_k in V^{sub} .

The Voronoi edge of PRE_i, PRE_j in V^{pre} can intersect the line segment $l_i^{pre}l_k^{sub}$ in at most two points. Therefore, each such triplet can cause $O(1)$ events. But there is only a linear number of triplets that have properties 1 and 2, because V^{pre} has only a linear number of Voronoi edges, by Theorem 2.1, and SUB_k is determined by l_i^{pre} in V^{sub} .

A *type 3 event* is a complement of *type 2 event*. It occurs at the intersection point of two Voronoi edges, one from V^{pre} , and the other one from V^{sub} . When the touch point l , which is an interior point of the arc of SUB_k in S'' , moves along the Voronoi edge e between two cones PRE_i and PRE_j in V^{pre} , towards the tangent point q of PRE_i and SUB_k . But l can not reach q , as q is not located inside the Voronoi cell of SUB_k in V^{sub} . The touch point l along e stops at the common endpoint of the arc of SUB_k with its neighboring arc of cone SUB_h in V^{sub} , i.e., SUB_k and SUB_h share a Voronoi edge in V^{sub} . Obviously, the four cones cause one event. $\square$

Now we can state our first main result.

Theorem 4.1. *We can compute a meeting of longest duration (or determine that no meeting is possible) for a set of n participants in time $O(n \log n)$.*

Proof. By Theorem 2.1, the Voronoi diagrams V^{pre} and V^{sub} can be constructed in time $O(n \log n)$. Next, we sort the Voronoi vertices by the order in which they are hit by the sets S' and S'' , respectively. Then we determine the T -coordinate for plane H'' such that S'' touches the location l' of the earliest possible meeting, and determine the circular arc of S'' that touches l' . Also, we compute which arc of S' will make the contact, as S' now starts expanding from l' . This preprocessing step can be found in Algorithm 2. Using the sorted vertex lists we can correctly process all events of

Figure 4.4: (a): Touching point l moves from the Voronoi edge between PRE_i and PRE_j to the interior of the arc of PRE_i . The arc of PRE_i is tangent to the arc of SUB_k at l . The dotted curves are the Voronoi edges between the left (resp., right) two discs, respectively. The Voronoi edge between PRE_i and PRE_j intersects $l_i^{pre} l_k^{sub}$ in l . (b): A type 3 event.

type 1 and, thus, maintain the boundary structure of S' and S'' . This step is in Algorithm 3. From the description of touching circular arcs and their neighbors we can predict when a type 2 event or type 3 event will occur that ends the current phase. This is presented in Algorithm 4. For each phase, we determine the maximum vertical segment, and output the maximum. This can be done in time $O(n)$ (Lemma 4.3). $\square$

The pseudo code of our algorithm for computing a new meeting location with longest duration for a set of participants is given in Algorithm 1. The preprocessing step is described in Algorithm 2, while the handling of different events of the sweep is shown in Algorithms 3 and 4.

A much more detailed algorithm is shown in Algorithm 5.

Figure 4.5 is an execution trace example of the algorithms. Two Voronoi diagrams V^{pre} and V^{sub} , and point m' is the bottommost point, and m'' is the topmost point. The optimal meeting location with the longest duration is on the Voronoi edge of C_6 and C_7 in (a). The circular arc of C_7 in S'' passes through m' in (b). S' is empty at

Algorithm 1 Compute the new meeting point with the longest duration

Input: Cone set $\{PRE_1, \dots, PRE_n\}$ and $\{SUB_1, \dots, SUB_n\}$. $PRE := \cap_{i=1}^n PRE_i$, $SUB := \cap_{i=1}^n SUB_i$ and $F := PRE \cap SUB$. The apex angle of PRE_i (resp. SUB_i) is $\frac{\pi}{2}$, and the bottommost (resp. topmost) point is its apex $((l_i^{pre})_X, (l_i^{pre})_Y, t_i^{pre})$ (resp. $((l_i^{sub})_X, (l_i^{sub})_Y, t_i^{sub})$).

Output: point l , maximize a vertical line segment $L \in F$, where l is the projection point of L on the X - Y plane.

- 1: execute Algorithm 2 to obtain the two sets of arcs L_1 and L_2 , one from PRE and the other one from SUB .
 - 2: **while** $L_2 \neq \emptyset$ /* L_2 holds the arcs from SUB */ **do**
 - 3: **if** at the current moment, a type 2 or type 3 event happens **then**
 - 4: execute Algorithm 4.
 - 5: **if** a type 1 event is included inside a type 2 or type 3 event /*a type 1 event happens when or before a type 2 or type 3 event ends*/ **then**
 - 6: execute Algorithm 3.
 - 7: **end if**
 - 8: **end if**
 - 9: **end while**
 - 10: from the computed meeting locations, find the meeting location with the longest duration.
-

Algorithm 2 Preprocessing Steps

- 1: construct the Voronoi diagrams V^{pre} and V^{sub} , respectively
 - 2: compute the bottommost point $m'(l'_X, l'_Y, t')$ of PRE and the topmost point $m''(l''_X, l''_Y, t'')$ of SUB , respectively
 - 3: sort the Voronoi vertices in increasing order of height (projected back in 3-d) in the Voronoi diagram of V^{pre} and V^{sub} , respectively
 - 4: let the circular arcs of SUB cut by the plane at time t' be S'' . Put the arcs of S'' into a linked list L_2 . Let the arc of SUB_k pass through l' . Let $SUB_h = \emptyset$.
 - 5: determine the discs PRE_i and/or PRE_j , whose arcs will touch the arc of SUB_k at time $t' + \epsilon$, where ϵ is a sufficient small value. Let the circular arcs of PRE cut by the plane at time $t' + \epsilon$ be S' . Put the arcs of S' into a linked list L_1 .
-

Algorithm 3 Processing type 1 events

- 1: **if** point p is a vertex in PRE /* p is currently being processed*/ **then**
 - 2: insert arc of PRE_r into L_1 , where, PRE_i , PRE_r and PRE_j determine p , and the arcs of PRE_i , PRE_r and PRE_j are consecutive neighbors in S' .
 - 3: **else if** point p is a vertex in SUB /* p is currently being processed*/ **then**
 - 4: delete the corresponding arc SUB_w from L_2 , where, SUB_k , SUB_w and SUB_h determine p , and the arcs of SUB_k , SUB_w and SUB_h are consecutive neighbors in S'' .
 - 5: **end if**
-

Algorithm 4 Processing type 2 and type 3 events

- 1: **if** $S' \cap S''$ is a point, determined by two neighboring arcs in S' , and one arc in S'' at the current moment, or determined by two neighboring arcs in S'' , and one arc in S' **then**
 - 2: compute the meeting location with the minimum meeting duration from the current moment to the moment when the determining arcs in either S' or S'' or both change. The meeting location is determined by two corresponding circles from S' , and one circle from S'' , or two circles from S'' , and one from S' , or by the intersection point of two circles in S' , and two circles in S'' .
 - 3: **else if** $S' \cap S''$ is a point, determined by one arc in S' , and one arc in S'' at the current moment /*the touching point is a tangent point of two arcs*/ **then**
 - 4: compute the meeting location with the minimum meeting duration from the current moment to the moment when the determining arc in either S' or S'' or both changes. The meeting location is determined by one corresponding circle from S' , and one circle from S'' .
 - 5: **end if**
-

Figure 4.5: (a) The trace of the touching points of S' and S'' . The new meeting point is located on a Voronoi edge. The left Voronoi diagram is V^{pre} , and the right Voronoi diagram is V^{sub} . (b) The trace starts from the earliest meeting point m' . (c) The trace along a Voronoi edge from V^{pre} . (d) The trace is determined by two circles, one from V^{pre} , and one from V^{sub} . (e) The trace along a Voronoi edge from V^{sub} , and ends at the latest meeting point m'' .

Algorithm 5 Compute the new meeting point with the longest duration

-
- 1: construct the Voronoi diagrams V^{pre} and V^{sub} , respectively
 - 2: compute the bottommost point $m'(l'_X, l'_Y, t')$ of PRE and the highest point $m''(l''_X, l''_Y, t'')$ of SUB , respectively
 - 3: sort the Voronoi vertices in increasing order of height (projected back in 3-d) in the Voronoi diagram of V^{pre} and V^{sub} , respectively
 - 4: let the circular arcs of SUB cut by the plane at time t' be S'' . Put the arcs of S'' into a linked list L_2 . Let the arc of SUB_k passes through l' . Let $SUB_h = \emptyset$.
 - 5: determine the discs PRE_i and/or PRE_j , whose arcs will touch the arc of SUB_k at time $t' + \epsilon$, where ϵ is a very small value. Let the circular arcs of PRE cut by the plane at time $t' + \epsilon$ be S' . Put the arcs of S' into a linked list L_1 . Let the disc be PRE_i such that $\overline{l'l''}$ pass through the Voronoi cell of PRE_i .
 - 6: **while** $L_2 \neq \emptyset$ **do**
 - 7: **if** both PRE_i and PRE_j exist and $SUB_h = \emptyset$, or both SUB_k and SUB_h exist and $PRE_j = \emptyset$ /*candidate l is on the Voronoi edge e of PRE_i and PRE_j , or on the Voronoi edge of SUB_k and SUB_h */ /*we only focus on the first case, the second case is in symmetry*/ **then**
 - 8: /*candidate l is also an interior point on the arc of SUB_k */ let the endpoint of Voronoi edge e further away from l' be p . Let the intersection point of e with $\overline{l_i^{pre} l_k^{sub}}$ be q , refer to Figure 4.4 (b)
 - 9: **if** $q \in e$ /*the endpoint p of e is after q along e */ **then**
 - 10: **if** q is in the Voronoi cell of SUB_k /*as q is also located in the Voronoi cell of PRE_i , the candidate l is a tangent point of SUB_k and PRE_i */ **then**
 - 11: compute the meeting location l and duration ll determined by PRE_i , PRE_j and SUB_k , put $\langle l, ll \rangle$ in a set A
 - 12: **else**
 - 13: compute the intersection point l of e and the Voronoi edge between SUB_k and SUB_h , where, the arc of SUB_h is the neighboring arc of SUB_k in L_2 , refer to Figure 4.4 (b). Compute the meeting duration ll at l and put $\langle l, ll \rangle$ in A
 - 14: **end if**
 - 15: **else**
 - 16: /*the endpoint p of e is before q , a type 1 event*/ compute the meeting location l and duration ll determined by PRE_i , PRE_j and SUB_k , put $\langle l, ll \rangle$ in A
 - 17: insert arc of PRE_r into L_1 , where, PRE_i , PRE_r and PRE_j determine p , and the arcs of PRE_i , PRE_r and PRE_j are consecutive neighbors
 - 18: **end if**
 - 19: **else if** $PRE_j = \emptyset$ and $SUB_h = \emptyset$ **then**
 - 20: /*candidate l is a tangent point between PRE_i and SUB_k */ compute the meeting location l and duration ll determined by PRE_i and SUB_k , put $\langle l, ll \rangle$ in A
 - 21: **end if**
 - 22: **if** /*dealing with type 1 events*/ the height of a Voronoi vertex is smaller than the time t of the current candidate l and the corresponding arc is not processed **then**
 - 23: insert the corresponding arc into L_1 , or delete the corresponding arc from L_2 , depending on the location of the Voronoi vertex.
 - 24: **end if**
 - 25: determine the arc PRE_i and/or its neighboring arc PRE_j in L_1 , that will touch the arc of SUB_k and/or its neighboring arc SUB_h at time $t + \epsilon$ in L_2 .
 - 26: **end while**
 - 27: from set A , find the meeting location l with the longest duration ll .
-

the beginning. The touching points are along the Voronoi edge between C_1 and C_2 in (c). Then, as S' expands, and S'' shrinks, the touching points are determined by C_1 and C_7 in (d). After that, the touching points are along the Voronoi edge between C_6 and C_7 in (e). The circular arc of C_1 in S' passes through the touching points, and finally the topmost point m'' .

4.4.2 Linear programming model

This section summarizes how to use the framework of the solution to linear programming type (LP-type) problems [113] to solve our problem. This work is mainly contributed by our co-authors R. Klein and F. Berger, and was published in [18].

4.4.2.1 Participants travel at the same speed

Let the new meeting location be x . Equation 4.1 can be rewritten as:

$$S(x) = \max_{1 \leq i \leq n} (t_i^{\text{pre}} + ||l_i^{\text{pre}} - x||), \quad (4.3)$$

and Equation 4.2 can be rewritten as:

$$E(x) = \min_{1 \leq i \leq n} (t_i^{\text{sub}} - ||l_i^{\text{sub}} - x||). \quad (4.4)$$

Let $f(x) = E(x) - S(x)$. The objective is to obtain x that can maximize $f(x)$.

Our problem belongs to LP-type maximization problems [73]. The solution to these problems can be obtained by applying the abstract framework of Sharir and Welzl [113]. The LP-type problem is written as (H, w) , where, w is the function which needs to obtain the maximal value ($w(H)$) for, and H is a set of constraints. In our problem, H is the set of participants, i.e., $PRE_i \cap SUB_i$. Without confusion, from here, we let $M = \{PRE_1 \cap SUB_1, \dots, PRE_n \cap SUB_n\}$, $H = M$. The function w is: $w(M) = (f(x), S(x))$, which will be explained later.

To prove that our problem belongs to an LP-type problem, the following two axioms must hold:

1. Axiom 1: For any Q, G with $Q \subset G \subset M$, $w(Q) \geq w(G)$.
2. Axiom 2: For any $Q \subset G \subset M$ with $w(Q) = w(G)$ and any $h \in M$, $w(G \cup \{h\}) < w(G)$ implies that $w(Q \cup \{h\}) < w(Q)$.

For Axiom2, intuitively, suppose that a subset $Q \subset M$ of participants has a longest stick L' inside the convex body formed by the corresponding cones, and a subset $G \subset M$ ($Q \subset G$) of participants has a longest stick L'' . Also suppose that $\|L'\| = \|L''\|$. After a pair of cones are added into G , if the length of the longest stick for G is shortened, then due to the insertion of the pair of cones, the longest stick for Q must also be shortened.

The computation of the longest stick following the framework is simple. There are three primitive operations, starting with the computation of the initial *basis*. A *basis* B is a set of constraints with $w(B') > w(B)$ for all proper subsets B' of B . A *basis* for a subset G of M is a basis $B \subset G$ with $w(B) = w(G)$. The maximum cardinality of any *basis* is called the *dimension* of (H, w) . We say that a constraint $h \in H$ violates a set G if $w(G \cup \{h\}) < w(G)$. In our case, the dimension of the basis is proved to be four by applying Helly's Theorem [46], which will be shown later.

1. Initial basis: in the beginning, compute the length of the longest stick from basis B_0 . Any participant can be used as initial basis. $w(B_0) < \infty$.
2. Violation test: given a basis B , and a pair of cones of a participant, which is a new constraint h , decide whether the pair of cones violate B , i.e., their participation changes the length of the longest stick computed so far.
3. Basis change: after Violation test, return new basis for $B \cup \{h\}$.

There are $O(n)$ participants, and the computation of the length of the longest stick from the initial basis is $O(1)$ by applying the geometric algorithm in Section 4.4.1. Each violation test and basis change can be done in $O(1)$ time.

Theorem 4.2. *The new meeting location with longest duration for n participants can be computed in $O(n)$ expected time.*

Figure 4.6: Axiom 2 does not hold for the participants M_1, M_2, M_3 , if we use $x \mapsto f(x)$ as objective function. For the sets $Q := \{M_1\}$, $G := \{M_1, M_2\}$ and $h := M_3$, we obtain $w(Q) = w(G) = 1$ and $w(G \cup \{h\}) < w(G)$, but $1 = w(Q \cup \{h\}) = w(Q)$.

So far we associate the LP-type problem with ours. Next, we will present the proof that our problem belongs to LP-type problem by correctly selecting the function w .

If we simply consider the set of participants M (i.e., cones) as the set of constraints and the function $w = f(x)$, then Axiom 2 does not hold, and thus not an LP-type problem. Figure 4.6 shows a situation where Axiom 2 does not hold. Our goal is to prove that the problem becomes an LP-type problem if we lexicographically maximize (f, S) instead of just maximizing f , i.e., $w(M) = (f(x), S(x))$. Lexicographically maximize (f, S) means, for two pairs $(f(x), S(x))$ and $(f(y), S(y))$, $(f(x), S(x)) < (f(y), S(y))$ if and only if $f(x) < f(y)$, or $f(x) = f(y)$ and $S(x) < S(y)$. The first inequality only focuses on f , while the second inequality implies that f obtains its maximum in more than one point. So, We have to analyze the second situation first, then see if Axiom 2 holds for $w = (f, S)$.

Lemma 4.4. $|S(x_1) - S(x_2)| = \|x_1 - x_2\|$ holds, if both locations x_1, x_2 maximize f .

Proof. Consider points $(x_1, S(x_1))$, $(x_2, S(x_2))$, $(x_1, E(x_1))$ and $(x_2, E(x_2))$ on the convex body F . Since the set of feasible points F is convex, the line segment B_S from $(x_1, S(x_1))$ to $(x_2, S(x_2))$ is completely contained in F . Analogously, the line segment B_E from $(x_1, E(x_1))$ to $(x_2, E(x_2))$ is in F .

If $(l, t) \in B_S$ were an inner point of F , we could schedule a meeting at l of longer duration than at x_1 . But this would be a contradiction to x_1 maximizing f ,

Figure 4.7: This is a 3-d figure. The four vertices are located on the convex body F intersected by the set of cones. Every vertical segment of maximum length contained in this parallelogram corresponds to a meeting of longest duration.

so $B_S \subset \partial F$ holds. There must be some cone PRE_j whose boundary contains B_S . Otherwise, the line segment B_S could intersect every cone's boundary in at most 2 points. Finally, $B_S \subset \partial PRE_j$ implies that B_S is a line segment of slope 1. $\square$

Lemma 4.5. *The locations maximizing f form a line segment.*

Proof. We have to show that every 3 locations maximizing f are collinear. Let x_1, x_2, x_3 denote locations maximizing f , such that $S(x_1) \leq S(x_2) \leq S(x_3)$. By Lemma 4.4, we conclude $\|x_3 - x_1\| = S(x_3) - S(x_1) = S(x_3) - S(x_2) + S(x_2) - S(x_1) = \|x_3 - x_2\| + \|x_2 - x_1\|$. So, the triangle inequality holds with equality, which means that x_1, x_2, x_3 are collinear. $\square$

Lemma 4.6. *The maximum meeting duration is determined by at most 2 participants, if f obtains its maximum in more than one point.*

Proof. Let x_1 and x_2 denote distinct points where f obtains its maximum and let B_S, B_E and PRE_j be defined as in the proof of Lemma 4.4. An analogous argument to the one showing that $B_S \subset \partial PRE_j$ leads to the fact that there is a downward cone

SUB_k such that $B_E \subset \partial SUB_k$. The optimum meeting duration T is obtained for a subset of the line segment $l_j^{pre} l_k^{sub}$. But even a 2-participant-meeting just of M_j and M_k has maximum duration T . If $j = k$, which is also possible, the maximum meeting duration is determined by only one participant. $\square$

Lemma 4.7. *There is a unique location x , which lexicographically maximizes $(f(x), S(x))$.*

Proof. We define $C := \{x \in \mathbb{R}^2 : \|l_1^{pre} - x\| \leq t_1^{sub} - t_1^{pre} - f(l_1^{pre})\}$, which is a disc. This implies $f(x) < f(l_1^{pre})$ for all $x \in \mathbb{R}^2 \setminus C$. Due to the compactness of C and the continuity of $f|_C$, the function $f|_C$ obtains its maximum T on C . Since $T \geq f(l_1^{pre}) > f(x)$ for all $x \in \mathbb{R}^2 \setminus C$, T is also the maximum of f on $\mathbb{R}^2$. The continuous function $S|_{f^{-1}(T)}$ obtains its maximum t on the compact set $f^{-1}(T)$. Due to Lemma 4.4, there can be at most one point x , such that $f(x) = T$ and $S(x) = t$. Hence the optimum point is unique. $\square$

So far, we have built the uniqueness of the function of w . Next, we need to prove our problem is indeed an LP-type problem. The proof is based on the two Axioms in the LP-type framework.

Theorem 4.3. *To find the optimum meeting for a set of participants is an LP-type problem.*

Proof. Clearly, a meeting, which is possible for a set of participants G , $G \subset M$, is also possible for every subset of G . Thus Axiom 1 holds.

For $M' \subset M$, recall M is the set of participants, let $OPT_{M'}$ be the lexicographically optimum pair $(f(x), S(x))$ for a meeting of all participants in M' . Let $f|_{M'}(x)$ denote the maximum duration for a meeting at location x of all participants in M' . Let $x(M')$ denote the unique location where the optimum meeting is held. We assume $OPT_Q = OPT_G$ and $OPT_{G \cup \{h\}} < OPT_G$ for $Q \subset G \subset M$, $h \in M$. According to Lemma 4.7, every point different from $x(Q)$ leads to a solution worse than $OPT_Q = OPT_G$ for the participants in Q . Hence, every point different from $x(Q)$ leads also for the participants in G to a solution worse than $OPT_Q = OPT_G$. Thus

$x(Q) = x(G)$. The inequality $OPT_{G \cup \{h\}} < OPT_G$ means that participant h cannot stay at the location $x(G)$ for the whole time interval $[S(x(G)), E(x(G))]$. But, this also means that he cannot stay the whole time interval at $x(Q)$ as $Q \subset G$, and we conclude $OPT_{Q \cup \{h\}} < OPT_Q$, thus Axiom 2 holds. $\square$

Now, we have to look at the dimension of the problem. The proof that the dimension of the problem is at most 4 uses Helly's theorem. A similar argument using Helly's theorem for a related question concerning the weighted Euclidean 1-center problem is described by Megiddo [74].

Theorem 4.4 (Helly's Theorem [46]). *Let $C_1, \dots, C_n$ be convex sets in $\mathbb{R}^d$, $n \geq d+1$. Suppose that the intersection of every $d+1$ of these sets is nonempty. Then the intersection of all the C_i is nonempty.*

Theorem 4.5. *The dimension of the meeting scheduling problem is at most 4.*

Proof. This proof is by contradiction. Let us assume that $B = \{M_1, \dots, M_b\}$ is a basis and $b = |B| > 4$ holds (i.e., $B = \{PRE_1 \cap SUB_1, \dots, PRE_b \cap SUB_b\}$). First we prove that for every 4-element subset $W \subset B$, the minimum value f' among all the f_W obtained for all possibilities of W , is also the maximum length f of the vertical line segments contained in the set of feasible points F_B , where, f_W is the maximum length of a vertical segment in the intersection of the cones in W , and F_B is the convex body intersected by the cones in B . Note that, by the assumption that there is a solution with $f(x) > 0$, the intersection of the sets in W is nonempty for every W . Clearly, f' is an upper bound on the length f of the longest stick contained in F_B . To obtain $f = f'$, we have to show that $f \geq f'$. Let SUB'_{i_k} denote the vertical cone with apex angle $\frac{\pi}{2}$ whose uppermost point is its apex $((l_k^{sub})_X, (l_k^{sub})_Y, t_k^{sub} - f')$ for $1 \leq k \leq b$. We observe that all sets in $\{PRE_1 \cap SUB'_1, \dots, PRE_b \cap SUB'_b\}$ are convex and, by definition of f' , the intersection of every 4 of them is nonempty. By Helly's theorem, this implies that the intersection of all sets in $\{PRE_1 \cap SUB'_1, \dots, PRE_b \cap SUB'_b\}$ is nonempty. But a point in this intersection corresponds to a vertical line segment of length at

least f' in the intersection of the original cones $\{PRE_1, \dots, PRE_b, SUB_1, \dots, SUB_b\}$. This means, that $f \geq f'$ and together with $f \leq f'$ we conclude that $f = f'$ holds.

Since 4-element of B can have the same longest length as the one B has, let $V \subset B$ denote a 4-element subset of B such that $f|_V$ and $f|_B$ obtain the same maximum value. Next, we need to get a contradiction with the assumption of B being the base. First we consider the case that $f|_V$ obtains its maximum in only one point, say x^* . This point must be the optimum meeting point for V as well as for B . But in this case $S|_V(x^*) = S|_B(x^*)$ holds, which means $OPT_V = OPT_B$ and we get contradiction to B being a basis.

Second, we consider the case that $f|_V$ obtains its maximum in more than one point. By Lemma 4.6, there is a subset $Z \subset V$ of at most 2 participants such that $f|_Z, f|_V$, and thus $f|_B$, obtain the same maximum value. Let x_Z denote the optimum meeting location for the participants in Z and x_B the optimum meeting location for the participants in B . If $x_Z = x_B$, same as above, we get a contradiction, and we are done.

So, now we have $x_Z \neq x_B$, and $f|_Z, f|_V$ and $f|_B$ obtain the same maximum value at both x_Z and x_B . By Lemma 4.5, the participants in Z can schedule meetings of maximum duration only in locations on the line through x_B and x_Z . This means, we can restrict our examination to the plane containing the vertical segments at x_B and x_Z . Figure 4.8 shows the curves resulting from intersecting this plane with the downward cones corresponding to participants in $B \setminus Z$. Since all participants in B can reach x_B at $S|_Z(x_B)$, they can also reach x_Z at $S|_Z(x_B) + \|x_Z - x_B\|$, which equals $S_Z(x_Z)$ by Lemma 4.4. Hence, the upward cones corresponding to participants of $B \setminus Z$ are not important for Figure 4.8. I.e, if we draw a vertical line from x_B , then the intersection points of the vertical line and all the upward cones in $B \setminus Z$ are lower than $(x_B, S_Z(x_B))$.

The optimality of x_B , i.e., by assumption that B is the basis but Z is not, ensures that there is a downward cone PRE_{i_j} which contains the point $(x_B, E|_Z(x_B))$ on its boundary and prevents improving the solution by moving from x_B into the direction

Figure 4.8: The longest meeting duration that is possible for the participants in Z is the same as for the participants in B . The dotted curves correspond to downward cones in $B \setminus Z$.

of x_Z . We conclude, that the optimum meeting for the at most 3 participants in $Z \cup \{M_{i_j}\}$ is as good as for all participants in B , which is a contradiction to B being a basis. $\square$

While Theorem 4.5 provides an upper bound on the dimension of the meeting scheduling problem, Figure 4.9 shows an example that the dimension is at least 4.

4.4.2.2 Allowing participants to travel at different speeds

Now, let us assume that the i th participant M_i travels at speed α_i , where $0 < \alpha_i < \infty$. Then the definition of Equation 4.3, which expresses the time of the arrival of the last participants at some location x has to be changed to

$$S(x) := \max_{1 \leq i \leq n} \left(t_i^{pre} + \frac{1}{\alpha_i} \|l_i^{pre} - x\| \right)$$

The function E in Equation 4.4 has to be changed accordingly.

For the geometric interpretation of the problem, the set of feasible points can still be described as the intersection of upward and downwards cones. The only difference to the unit speed case is that the apex angle of the cones belonging to participant M_i

Figure 4.9: Set of 4 participants $\{M_1, M_2, M_3, M_4\}$ and their way to the optimum meeting location. The start of their optimum meeting is determined by participants M_1 and M_3 , while the end is determined by M_2 and M_4 . Removing any participant enables a meeting with longer duration. If one, for example, removes participant M_1 , the optimum meeting location becomes $(1, 1)$ and the duration of the optimum meeting increases by $\sqrt{2} - 1$.

traveling at speed α_i becomes $2 \arctan(\alpha_i)$. Due to the different speeds, the projection of the cones in PRE on the XY -plane is an additively multiplicatively weighted furthest site Voronoi diagram of circles. We do not know whether our $O(n \log n)$ algorithm presented in [62] can be generalized to this case. But it is important for the LP-approach that it is still possible to find the optimum meeting location for at most 5 participants in time $O(1)$ by inspecting every cell of the overlay of the Voronoi diagrams belonging to the projections of the cones in PRE and SUB on the XY -plane.

Another problem for generalizing our results to participants traveling at different speeds is that Lemma 4.4 is no longer true. The following lemma serves as a substitute.

Lemma 4.8. *If $x_1 \neq x_2$ and both locations x_1, x_2 maximize f , then $S(x_1) \neq S(x_2)$, even if participants travel at different speeds.*

Figure 4.10: The longest meeting duration that is possible for the participants in Z is the same as for the participants in B . The dotted curves d_1, d_2 correspond to downward upward cones in $B \setminus Z$, while the dotted curve u_1 corresponds to an upward cone.

Proof. As shown in the proof of Lemma 4.4, the line segment from $(x_1, S(x_1))$ to $(x_2, S(x_2))$ is completely contained in the boundary of a cone, which implies that it has slope > 0 . $\square$

For the proof of Lemma 4.7, the only necessary change is to replace the use of Lemma 4.4 by the use of Lemma 4.8. The proof of Lemma 4.6 as well as of Lemma 4.5 directly translate to our generalized problem setting. There is also a minor change with respect to the proof of Theorem 4.5 and Figure 4.8. Now, with the cones having different angles, even upward cones can prevent us from improving the solution, see Fig. 4.10. Finally, we conclude with the generalized result.

Theorem 4.6. *We can compute a meeting of longest duration (or determine that no meeting is possible) for n participants traveling at different speeds in expected time $O(n)$.*

4.5 Maximizing the number of participants in Euclidean space

When no new meeting can be scheduled for the set of n participants, i.e., $F = \emptyset$, we want to find the maximal number of participants that can schedule a new meeting with a meeting time ≥ 0 . Still, all the participants travel at the same speed.

Recall that each participant's time dependent location corresponds to an upper cone and a lower cone, and each cone has an opening angle 90° . This problem can be solved in a brute force way by sweeping the horizontal plane (time plane) H through the two sets of cones — upper cones and lower cones. Each time plane cuts the cones into two sets of circles. Then, at each time plane, we find the maximal number of participants that contribute to a common intersection area of the two sets of circles, and the final maximal number of participants in all the time planes is the result.

In order to use the sweep plane technique, we have to find the event points (planes), i.e., the time events, when the upper and lower cones are cut. Suppose that we have the final maximal number of participants K , $2 \leq K \leq n$. Since the K participants can schedule a new meeting, there is an intersection area of the K upper cones and K lower cones corresponding to the K participants. Let the upper cones and the lower cones intersect in a convex body P . In the degenerate case, the convex body may be a single point.

Lemma 4.9. *If p is a point that maximizes the number of participants, then there exists a meeting point q , such that (1) q is the lowest point on the convex body P ; (2) q is a meeting point for the same set of participants (q could be p).*

Proof. As the convex body $P = (\cap \text{lower cones}) \cap (\cap \text{upper cones}) \neq \emptyset$, and each cone has an opening angle 90° , there must be a lowest point in P . Since $p \in P$, all the points in P are determined by the same participants as p . Thus, the lowest point must be determined by the same participants as p as well. $\square$

The following lemma is based on the fact that the upper and lower cones have the

same opening angle 90° .

Lemma 4.10. *The lowest point low_p of the intersection of the K lower cones and the highest point $high_q$ of the intersection of the K upper cones are the lowest and highest point of the convex body P , respectively.*

The computation and relationship of the smallest enclosing circle of a set of sites, and the arrangement of a set of cones have been introduced in Lemma 2.1. Besides Theorem 2.1, there is also a property of the MEC of a set of points related to the furthest Voronoi diagram.

Property 4.1. *Each vertex of the furthest Voronoi diagram for an n point set is determined by three points, and the circle passing through the three points covers all the n points. The MEC of the points is the circle with the smallest radius.*

The following lemma presents the three cases for the lowest (highest) point on the convex body P .

Lemma 4.11. *Point low_p ($high_q$) is determined by the lower cones (upper cones) as per one of the following three cases:*

1. *One of the apices of the lower (upper) cones;*
2. *The lowest (highest) intersection point of two lower cones (upper cones);*
3. *The intersection point of three lower cones (upper cones).*

Proof. By Lemma 4.10, the lowest point of the convex body must be the lowest intersection point of the set of lower cones. By Lemma 2.1, the lowest intersection point is the center of the MEC of the set of circle sites obtained by cutting the lower cones at a horizontal plane. There are three cases for the center of MEC. Correspondingly, there are three cases for the location of the lowest intersection point of the set of lower cones. □

Algorithm 6 Finding-Maximal-Participants

Input: Cone set $\{PRE_1, \dots, PRE_n\}$ and $\{SUB_1, \dots, SUB_n\}$. $PRE := \cap_{i=1}^n PRE_i$, $SUB := \cap_{i=1}^n SUB_i$ and $F := PRE \cap SUB$. The apex angle of PRE_i (resp. SUB_i) is $\frac{\pi}{2}$, and the bottommost (resp. topmost) point is its apex $((l_i^{pre})_X, (l_i^{pre})_Y, t_i^{pre})$ (resp. $((l_i^{sub})_X, (l_i^{sub})_Y, t_i^{sub}))$.

Output: the maximal number of participants that can schedule a new meeting with duration ≥ 0 .

- 1: /*Event points:*/ compute the intersection points of every two lower cones
- 2: compute the intersection points of every three lower cones
- 3: **for** each event point computed above and each apex of the lower cones **do**
- 4: count the number of matching circle pairs cut by the time plane at the height of the event point that enclose the event point
- 5: **end for**
- 6: output the maximal number computed in above step

Recall that the maximal number of participants is K , and the K upper cones and K lower cones belonging to the K participants intersect at the convex body P . Then, the lowest (highest) point of P is determined by one of the cases in Lemma 4.11.

If we compute all the lowest intersection points of every two cones, and the intersection points of every three cones, then the lowest (highest) point low_p ($high_q$) of P must belong to one of the intersection points or the apices of the lower (upper) cones. Also, if we cut the cones at a horizontal plane at the position of low_p , then all of the K upper circles obtained must include low_p . What remains is to find the maximal value K . Thus, we obtain the following brute force algorithm to find the maximal number of participants K . In the algorithm, a matching circle pair is defined as a pair of circles obtained by cutting the lower and upper cone of a participant at a horizontal plane. Similarly, a matching cone pair is the lower and upper cone of a participant.

Lemma 4.12. *Algorithm 6 can be executed in $O(n^4)$ time.*

Proof. There are, in total $O(n^3)$ event points. For each event point, counting the number of pairs of circles enclosing the event point takes $O(n)$ time. $\square$

Can we improve upon this? We next show that our problem is a 3SUM-hard problem through the following problem (3SUM-hard problems are introduced in Section 2.2.3). Boris and Har-Peled [9] studied the *maximum depth* problem in a disc set, where, *maximum depth of a point* is the largest number of discs covering it. They establish the following lemma:

Lemma 4.13. [9] *Determining a point of maximum depth in an arrangement of n discs is 3SUM-hard.*

In the proof they showed that the POINT-ON-3-LINES problem, (i.e., given a set of lines in the plane, are there three that pass through the same point?), can be reduced to determining a point of maximum depth in an arrangement of n discs. By that argument, computing the depth in $O(n^2)$ time exactly appears difficult.

Lemma 4.14. *Finding the maximal number of participants traveling at the same speed that can schedule a new meeting with a meeting time ≥ 0 , is 3SUM-hard.*

Proof. We reduce the problem in Lemma 4.13 to the problem of finding maximal number of participants problem, and thus can prove that our problem is 3SUM-hard. Consider constructing a set of cones from the n discs in set A . The n discs are located at time plane H . For a disc $C \in A$, there is one upper cone and one lower cone corresponding to it. The apices of the upper and lower cones are located at the center of C , and have the same vertical distance to H . The interior angle of each cone is 90° . In this setting, a point p which has the maximum depth K in the arrangement of the n discs exists, if and only if point p is included in the upper and lower cones built from the discs which cover p . $\square$

It is clear that a significant gap exists between n^2 and our result n^4 . It would be interesting to see if a better algorithm exists.

Another interesting question is the following: Given a set of n participants and a parameter k , find a meeting point which maximizes the duration for a meeting of at least $n - k$ participants. For example, we could think of a situation where half the

participants have a quorum. An optimum meeting for all except k participants can be found by inspecting the optimum meeting point for every subset of 4 participants (according to Theorem 4.5), but we would like to find a fast algorithm. However, according to the following lemma, a subquadratic solution in the full range of k seems unlikely.

Lemma 4.15. *Given a set of n participants and a parameter k , finding a meeting point which maximizes the duration for a meeting of at least k participants is a 3SUM-hard problem.*

Proof. According to Matoušek [73], the problem of finding the smallest circle enclosing at least q points of a set of n points in the plane, is 3SUM-hard. We show that our problem is $\lll_n$ finding the smallest circle enclosing at least q points of a set of n points in the plane, see the definition of $\lll$ given in Section 2.2.3.

Let our problem be B , and let the problem of finding the smallest circle enclosing at least k points of a set of n points in the plane be A . For each element $a \in A$, we construct a cone cone_a and cone'_a , where, cone_a (resp. cone'_a) is an upwardly (resp. downwardly) directed cone in $\mathbb{R}^3$, with apex a (resp. a'), and with an interior angle of 90° . All the apices of the upward cones are located at the same plane L , and the apices of the downward cones are located at the same plane L' which is above L . Point a' and a are located on a vertical line. Each participant corresponds to a cone cone_a and cone'_a . We claim that a meeting point p maximizes the duration for a meeting of at least k participants, if and only if p is the center of the smallest circle enclosing at least k points of the set of n points, where, these k points are the apices of the set of k cones corresponding to the k participants.

Proof of “only if part”: We claim that the lowest point p of the intersection of the k upward cones maximizes the duration for a meeting of k participants. This is because p' , the highest intersection point of the k downward cones, is on the same vertical line as p . So, p and p' determine the meeting location with the longest duration. By Lemma 2.1, p is the center of the smallest circle enclosing the k apices.

The proof of “if part” is similar. □

4.6 Meeting scheduling in a transportation network

The geometric model we considered in Section 4.4 can be modified in many ways. For example, in a graph-based model, one could assume that travelers use a transportation network that combines highway, railway, and air line connections. In this section, we can consider a situation where

- participants do not move as the crow flies; instead, they travel along the edges of a given geometric network in the plane
- both the participants' original locations and the meeting locations can be arbitrary points on some edges of the network
- each participant has an individual weight; it can be used in modeling the individual importance or the traveling speed of each participant
- participants come in different groups; for the meeting, only one member out of each group is required

In other words, let the geometric network be a graph G . Let the n participants belong to k different groups. Let each participant have an individual weight. Also, one participant in a group is needed for the new meeting. We want to find a new meeting location with the longest duration for the k representatives.

In this situation, the distance between two locations in the network is given by the minimum Euclidean length of a connecting path in the network. We use *weighted furthest color Voronoi diagrams on graphs* to solve the problem, where, a *color* represents a group. Given a candidate location l on the network, such a diagram tells us which one, among those k group representatives that have shortest travel time to l within their groups, would be the latest to arrive there. The resulting Voronoi regions are bounded by marks on the network's edges. I.e., the weighted furthest color Voronoi diagram on the graph G for the n weighted participants in the k groups is a division

Algorithm 7 Computing a new meeting location with the longest duration in a graph model

- 1: construct the two weighted furthest color Voronoi diagrams, where the sites are the previous meeting locations and the subsequent meeting locations of the participants with individual weights and within different groups, respectively
 - 2: overlay the two Voronoi diagrams
 - 3: **for** each edge on the overlay **do**
 - 4: compute a meeting location with the longest duration
 - 5: **end for**
 - 6: find the new meeting location with the longest duration among all the computed meeting locations
-

of the edges of G , such that each piece on an edge belongs to a participant. See Hurtado et al. [49] for the details. Refer to Figure 2.11 for a furthest color Voronoi diagram in the plane, and Figure 2.12 for the weighted Voronoi diagram on a line for three sites. Both of the two figures are introduced in Section 2.2.1.4.

Since each participant has a previous meeting location and a subsequent meeting location, similar to the geometric setting, correspondingly there are two weighted furthest color Voronoi diagrams for the participants. We overlay them, and then on each edge of the overlay, we compute a meeting location with the longest duration. Then, we pick the one with the longest meeting duration. In the following, α denotes the extremely slowly growing inverse Ackermann function, that can be considered as a constant for practical purposes.

Theorem 4.7. *Let G denote a geometric network of m vertices and e edges in the plane. Suppose that there are k groups of, in total, n participants with individual traveling speeds. Then, using Algorithm 7, a meeting of longest possible duration can be scheduled in time $O(en\alpha(k)\log k + n\log n + nm\log m)$.*

Proof. We analyze Algorithm 7. When superimposing the two diagrams, their complexities simply add up, as they are on the geometric network. In one pass over all edge segments thus obtained, we can compute the optimum solution. Since we can construct the Voronoi diagrams in time $O(en\alpha(k)\log k + n\log n + nm\log m)$, using Theorem 8 of Hurtado et al. [49], our result is proven. $\square$

4.7 Conclusion

We conclude that the meeting scheduling problem, as stated here, can be solved in $O(n \log n)$ time by an algorithm that is based on geometrical observations and by an expected linear-time randomized algorithm within the LP-type framework.

Many interesting questions are still under investigation. For example we want to generalize our results to other metrics. Another question is whether a deterministic linear-time algorithm for LP-type problems can be applied to our problem. In principle such an algorithm has been described by Chazelle and Matoušek [73], but it needs more restrictive assumptions than the algorithm by Sharir and Welzl we have used in this chapter.

Chapter 5

Shortest Paths among Growing Discs

In this chapter, we design efficient algorithms for one of the shortest path applications: finding a shortest path for a robot among a set of obstacles, where, the obstacles are growing discs. The robot travels at a steady speed, and the discs grow either at the same speed or at various speeds. We state the solution to the problem in Euclidean metric.

5.1 Introduction

A well-studied and fundamental problem in computational geometry is the determination of shortest paths (see e.g., [80] for a survey). It finds applications in a number of key areas such as robotics, GIS, graphics, computer games, and VLSI design. In robotics, the problem arises in the context of planning collision-free motions for a robot operating in an environment containing obstacles. An important algorithmic challenge arises when the obstacles themselves move, as is the case in dynamic environments. This challenge is increased further, if the obstacle motion is unpredictable.

A recently introduced **model** [114, 115], presents a first step to incorporate such unpredictability in path planning. The unpredictability of obstacle locations is modeled by discs growing over time at some (maximum) speed. The velocity by which

the discs grow is assumed to be a constant and to be less than the maximum velocity of the robot. Assume that the robot's speed exceeds that of a disc and that the robot comes in contact with that disc during its motion. Then, when the robot arrives at the disc, the disc will act as an obstacle and the robot is unable to leave the disc as it grows. If all disc speeds exceed that of the robot, then, the robot must reach the target as fast as possible by following a straight line from the starting point s to the target point g . This can be detected in $O(n)$ time, by checking if g is covered by any growing disc at the time when the robot arrives at g . In case some discs, say set D_1 , have smaller speeds than that of the robot, and the remaining discs $D_2 = \mathcal{D} - D_1$ have equal or large speeds than that of the robot, we solve the problem for D_1 , and subsequently apply the above test for D_2 . Under the condition that the velocity of the robot is bigger than the one of any disc, solutions are typically given in configuration space, where a circular robot shrinks to a point and obstacles are augmented appropriately by the radius of the robot [30]. One is therefore interested in finding shortest paths which are guaranteed to be collision-free for a point robot operating in an unpredictable environment.

We state the following geometric problem, called **shortest path among growing discs (SPAGD)** to be addressed in this chapter: given a point robot R , moving at a maximum velocity V_R , and a set of discs $\mathcal{D} = \{C_0, C_1, \dots, C_n\}$ in Euclidean space, where the radius of each disc C_i grows at speed V_i , with $V_i < V_R$, find a shortest path from a start location s to a goal location g for the robot that avoids the growing discs. When discs are growing at the same speed (same-speed discs), the SPAGD problem is written as C-SPAGD, and V-SPAGD when discs are growing at different speeds (various-speed discs). This SPAGD problem was introduced by van den Berg and Overmars [114, 115], and was motivated in the context of motion planning in virtual environments such as commercial video games. They presented an $O(n^3 \log n)$ solution for the case when discs grow at the same speed. They stated an algorithm but did not derive the time complexity for the case when discs grow at different speeds.

In this chapter we present an $O(n^2 \log n)$ algorithm for the C-SPAGD problem.

Our algorithm thus improves upon the previous known result by a factor of n . We achieve our result by deriving some geometric insights into shortest paths among growing discs and, as tools, using a circular sweep technique and additively weighted Voronoi diagrams. With various-speed discs, our algorithm has $O(n^3 \log n)$ time complexity. In [59], an $O(n^2 \log n)$ algorithm to find a shortest path from s to g among a set of discs is presented where the discs are static. Our solution for the growing disc at the same speed problem instance therefore matches the complexity derived for the simpler static (fixed disc) case [26].

Section 5.2 presents preliminaries, Section 5.3 outlines our algorithms, Section 5.4 presents the preprocessing step, and the key algorithms are described in Sections 5.5 and 5.6. Section 5.7 briefly introduces another key part in the overall algorithm. We conclude with Section 5.9.

5.2 Preliminaries

In this section, we introduce some terminology beginning with a **formal definition** for the shortest path among growing discs problem.

Let $\mathcal{D} = \{C_0, C_1, \dots, C_n\}$ be a set of discs, where, each disc $C_i \in \mathcal{D}$ is represented by a center o_i , radius $r_i(t)$, and speed V_i , denoted by $C_i(t) = (o_i, r_i(t), V_i)$. Each disc grows as a function of time t . The radius of C_i at time t is $r_i(t) = r_i(T_0) + V_i t$, where, $r_i(T_0)$ is the radius of $C_i(T_0)$, and T_0 is the starting time. A path π is a function $[t_a, t_b] \rightarrow \mathcal{R}^2$ where $\pi(t_x)$ denotes the location of the robot R , at time t_x , and R is located at $\pi(t_a) = a$ and reaches $\pi(t_b) = b$ at time t_b . A point $\pi(t_x)$ is collision-free with respect to $\mathcal{D}$ at time t_x if $\pi(t_x) \cap (\bigcup_{i=1}^n C_i(t_x) - \partial \bigcup_{i=1}^n C_i(t_x)) = \emptyset$, where, ∂A represents the boundary of object A . A path π from $\pi(t_a)$ to $\pi(t_b)$ is collision-free if every point $\pi(t_x)$ on π is collision-free at time t_x , where $t_x \in [t_a, t_b]$. The problem of finding a **shortest collision-free path** from a start location s , $\pi(t_s)$, to a goal location g , $\pi(t_g)$, in $\mathcal{R}^2$ is to find a collision-free path $\pi: [t_s, t_g] \rightarrow \mathcal{R}^2$ which minimizes $|t_g - t_s|$. We assume that the robot starts from s at time $t_s = T_0$.

Since the discs grow over time, it is easily seen that the robot should always move at maximal velocity. Thus, given a path and a starting time, one can determine the arrival time for any point on the path. The segments on the robot's path are time encapsulated, but, for ease of notation, the time parameter t is omitted when no ambiguity arises.

To solve the problem, van den Berg and Overmars [114, 115] use a 3-dimensional model where the x and y axes describe the location and the z axis represents time. Cones are used to represent the change in obstacles over time. Any shortest path from s to g for the robot is composed of alternating straight line segments of slope $\frac{1}{V_R}$ which are tangent to pairs of cones and logarithmic spiral segments which lie on the surface of each individual cone.

We solve the problem in 2- d thereby avoiding 3-dimensional geometric objects, operations and primitives. We look at tangents between growing discs, and spiral segments around discs instead of cones. A tangent line segment l , also called *tangent*, between two discs $C_i(t_p)$ and $C_j(t_q)$ is a line segment pq ($p \in \partial C_i(t_p), q \in \partial C_j(t_q)$), such that $pq \perp o_i p$, $pq \perp o_j q$, and $|pq| = V_R(t_q - t_p)$. The *direction* of l is from p to q .

A tangent $l(= pq)$ from $p \in C_i$ to $q \in C_j$ that is collision free during time $[t_p, t_q]$ with respect to other growing discs is called a *valid tangent*. The formal definition is given below.

Definition 5.1. Valid tangent: Let l be a tangent from $C_i \in \mathcal{D}$ to $C_j \in \mathcal{D}$. For a disc $C_k \in \mathcal{D}$, $k \neq i$ and $k \neq j$, let $p_k \in l$ be the point closest to the center o_k of C_k among all the points on l . Tangent l is called a *valid tangent*, if $(C_k(t_{p_k}) - \partial C_k(t_{p_k})) \cap p_k = \emptyset$, where t_{p_k} is the robot's arrival time at p_k .

There is a relationship between the endpoints of a valid tangent and the Voronoi diagram of the discs in $\mathcal{D}$. It is well known that the Voronoi diagram of a set of discs does not change when discs are allowed to grow [90]. The Voronoi diagram of same-speed discs is called additively weighted Voronoi diagram. The Voronoi diagram of various-speed discs is called additively, multiplicatively weighted Voronoi diagram. Let V_{C_i} be the Voronoi cell of C_i .

Figure 5.1: Right-right, right-left, left-right, left-left tangents from disc $C_i(T_0)$ to disc C_j , which is originally $C_j(T_0)$.

Lemma 5.1. *Let l be a valid tangent from C_i to C_j . Then the tangent point of l on C_i (resp. C_j) is located inside V_{C_i} (resp. V_{C_j}).*

Proof. Let l be tangent to disc C_i (resp. C_j) at point p . Since l is valid, point p is not inside any disc C_k other than C_i (resp. C_j). I.e., p is closer to C_i (resp. C_j) than to C_k , or p has the same distance to C_i (resp. C_j) as to C_k , where, p is located on the Voronoi edge between V_{C_i} (resp. V_{C_j}) and V_{C_k} . $\square$

Note that when we say that point p is located inside V_{C_i} , p is either inside the Voronoi cell of C_i , or p is located on a Voronoi edge of V_{C_i} .

There are four types of tangents between a pair of discs C_i and C_j . These are: right-right, right-left, left-right, and left-left tangents, denoted by $l_{ij}^{rr}(t)$, $l_{ij}^{rl}(t)$, $l_{ij}^{lr}(t)$, and $l_{ij}^{ll}(t)$, respectively. Knowing that these are time-encapsulated, we often use the short forms l_{ij}^{rr} , l_{ij}^{rl} , l_{ij}^{lr} , and l_{ij}^{ll} , refer to Figure 5.1. The right-right and right-left tangents are reflections of left-left and left-right tangents. Therefore, properties derived for the right-right and right-left tangents are also applicable to left-left and left-right tangents.

A departure curve for a particular tangent is the trace, over time, of the tangent's endpoints that are located on the starting disc as the disc grows. Similarly, an arrival curve for a tangent is the trace of the tangent's endpoints on the growing arrival disc. Refer to Figure 5.3 illustrating the right-right departure curve when the discs grow

Figure 5.2: For the proof of Lemma 5.2. Disc C_i and C_j grow at the same speed. Tangent $l_{ij}^{rr}(T_0)$ and $l_{ij}^{rr}(t)$ with $t > T_0$.

at the same speed. We denote the departure curve of the right-right tangent l_{ij}^{rr} as $DC(l_{ij}^{rr})$ and $DC(l_{ij}^{rl})$ for the right-left departure curve. When no ambiguity arises, we use $DC(l_{ij})$ to represent either $DC(l_{ij}^{rr})$, or $DC(l_{ij}^{rl})$, or both. We also denote the right-right arrival curve as $AC(l_{ij}^{rr})$ and $AC(l_{ij}^{rl})$ for the right-left arrival curve.

Assume for the following discussion, a polar coordinate system with coordinate axes: time, T , and angle, Θ . (Actually polar coordinates are typically radius and angle. In our case, we define that the speed of the starting disc equals to time, therefore, the change in radius of the disc is precisely the change in time.) When discs grow at different speeds, $DC(l_{ij}^{rr})$ and $DC(l_{ij}^{rl})$ (resp. $AC(l_{ij}^{rr})$ and $AC(l_{ij}^{rl})$), are sine curves [115]. When discs grow at the same speed, $DC(l_{ij}^{rl})$ (resp. $AC(l_{ij}^{rl})$) is a sine curve, while $DC(l_{ij}^{rr})$ (resp. $AC(l_{ij}^{rr})$) is a straight line. This is stated in the following lemma.

Lemma 5.2. *Let disc C_i and C_j have equal growing speed $V_i = V_j$. Then, for any $t > T_0$, $l_{ij}^{rr}(t) \parallel l_{ij}^{rr}(T_0)$, and $DC(l_{ij}^{rr})$ is a straight line.*

Proof. Let the two discs be $C_i(T_0) = (o_i, r_i, V_i)$ and $C_j(T_0) = (o_j, r_j, V_j)$, $V_i = V_j$. Refer to Figure 5.2, $l_{ij}^{rr}(T_0)$ is tangent to C_i at point p , and tangent to C_j at point q . For a tangent $l_{ij}^{rr}(t)$, $t > T_0$, $l_{ij}^{rr}(t)$ is tangent to C_i at point p_t , and tangent to C_j

at point q_t . $o_i p_t \perp l_{ij}^{rr}(t)$, $o_j q_t \perp l_{ij}^{rr}(t)$. From o_j , draw $o_j p'' \perp o_i p_t$. $o_j p'' \parallel l_{ij}^{rr}(t)$, and $|o_j p''| = |l_{ij}^{rr}(t)|$. We have:

$$|o_i o_j|^2 = [r_j + t + |l_{ij}^{rr}|/V_R - (r_i + t)]^2 + |l_{ij}^{rr}|^2$$

Solving this for $|l_{ij}^{rr}|$, we obtain:

$$|l_{ij}^{rr}| = \frac{\pm \sqrt{(V_R^2 + 1)|o_i o_j|^2 - V_R^2(r_j - r_i)^2} - (r_j - r_i)}{V_R^2 + 1},$$

and $|l_{ij}^{rr}|$ is a constant. From o_j , draw $o_j p' \perp o_i p$. The line segment $o_j p' \parallel l_{ij}^{rr}(T_0)$, and $|o_j p'| = |l_{ij}^{rr}(T_0)|$. Since $|l_{ij}^{rr}|$ is a constant, $|o_j p''| = |o_j p'|$. But there is only one right triangle $o_i \hat{p} o_j$, with $\hat{p}$ located on $o_i p$, and $|\hat{p} o_j| = |l_{ij}^{rr}|$. Point p' , p'' and $\hat{p}$ are the same point. Thus, o_i , p and p_t are co-linear. Therefore, $l_{ij}^{rr}(t) \parallel l_{ij}^{rr}(T_0)$ and $DC(l_{ij}^{rr})$ is a straight line. $\square$

For a pair of discs C_i and C_j , a *valid buffer* is an interval I on $DC(l_{ij})$ such that for any point $p \in I$, $l_{ij}(t_p)$ is a valid tangent, where, t_p is the starting time of l_{ij} at point p . Refer to Figure 5.3. We will use time interval $[t_a, t_b]$, called *life span*, or the geometric description $[a, b]$ to represent a segment on $DC(l_{ij})$, where, a and b are the two endpoints of the segment.

Corollary 5.1. *A valid buffer $[t_1, t_2] \in DC(l_{ij})$ must be located inside V_{C_i} . The ending point of $l_{ij}(t)$ with $t_1 \leq t \leq t_2$, must be located inside V_{C_j} .*

Proof. According to Lemma 5.1, the two endpoints of the valid l_{ij} must be located inside V_{C_i} and V_{C_j} . Since $l_{ij}(t)$ is valid, $t \in [t_1, t_2]$, the valid buffer $[t_1, t_2] \in DC(l_{ij})$ must be located inside V_{C_i} , and the ending point of $l_{ij}(t)$ must be located inside V_{C_j} . $\square$

Lemma 5.3. *Let l be a valid tangent from C_i to C_j at time t_l . Let T_p , $T_p > t_l$, be the earliest time that $DC(l)$ intersects the boundary of V_{C_i} , and $T_{p'}$, $T_{p'} > t_l$, be the earliest starting time of l such that the ending point of $l(T_{p'})$ is on an edge of V_{C_j} . Then, a valid buffer $[t_l, t_r] \in DC(l)$ exists, where, $t_r \leq \min\{T_p, T_{p'}\}$, refer to Figure 5.3.*

Figure 5.3: The Voronoi diagram of a set of discs growing at the same speed. $DC(l_{ij}^{rr})$, $AC(l_{ij}^{rr})$ and tangent $l_{ij}^{rr}(T_0)$. Tangent $l_{ij}^{rr}(T_p)$ intersects V_{C_i} with p . Tangent $l_{ij}^{rr}(T_{p'})$ intersects V_{C_j} with point q . $p \in DC(l_{ij}^{rr})$, $p' \in DC(l_{ij}^{rr})$, $q \in AC(l_{ij}^{rr})$. The bold segment on $DC(l_{ij}^{rr})$ is the valid buffer on $DC(l_{ij}^{rr})$, determined by disc C_R .

Figure 5.4: Tangent l from C_i to C_j . (a) C_k projects on l , and C_k and $C_{k'}$ are lower than C_j along l . (b) C_k and C_r are two right-blocks of l . C_f is the left-closest-block of l .

Proof. According to Corollary 5.1, the endpoints of $l(t)$, $t \in [t_l, t_r]$ must be located inside V_{C_i} and V_{C_j} . Tangent $l(t_r)$ either intersects an edge of V_{C_i} or V_{C_j} . $\square$

We also need the following definitions in the next sections.

The tangents originating from C_i are sorted by angles in counterclockwise (CCW) order. These angles are between the horizontal line passing through the center of C_i (o_i) in direction $+\infty$ and each tangent in CCW order. The tangent points on C_i have the same order as the order of the tangents. Define *slope* as the angle of a tangent. Around C_i , starting from direction $+\infty$, in CCW order, the slopes of the tangents increase.

Considering the tangents originating from C_i . Let l be a valid tangent from C_i to C_j . Refer to Figure 5.4. We say that disc C_k **projects on** l , if there exists a point p , $p \in l$, and $o_k p \perp l$. We say that C_k is **lower** than C_j along l if: (1) C_k projects on l , or (2) the projection point is not on l , but it is closer (has shorter distance) to

the starting point of l than to the ending point of l . Otherwise, C_k is *higher* than C_j along l . (When the projection point is the ending point of l , which is a trivial case, we use the term “not higher” or “not lower” according to the context in the later sections.) We say that C_k is a *left-block* (resp. *right-block*) of l , if C_k projects on l and $\text{slope}(l_{ik}^{rr}) \geq \text{slope}(l)$ (resp. $\text{slope}(l_{ik}^{rl}) \leq \text{slope}(l)$). C_k is the *left-closest-block* of l , if C_k is a left-block of l , and l_{ik}^{rr} has the smallest slope among all the right-right tangents of discs that are left-blocks of l . Similarly, C_k is the *right-closest-block* of l , if C_k is a right-block of l , and l_{ik}^{rl} has the biggest slope among all the right-left tangents of discs that are right-blocks of l .

5.2.1 Departure curves computation

In this section, we establish an important property of departure curves for which we need to derive some formulas first. Let $C_i = (o_i, r_i(\bar{t}), V_i)$ and $C_j = (o_j, r_j(\bar{t}), V_j)$ be two discs which may grow at different speeds. To simplify the computation of departure curves, we normalize V_i to be 1 and consider the cases when $V_j > 1$ and $V_j < 1$, respectively. When $V_j = 1$, the computation is simpler, and is presented at the end of this section.

First we will show that:

Lemma 5.4. *When discs grow at different speeds, $DC(l_{ij}^{rr})$ (resp. $DC(l_{ij}^{rl})$) exists for a particular time range.*

Proof. Let $\bar{t}$ be the starting time of l_{ij}^{rr} , and t be the time spent by the robot on l_{ij}^{rr} . Refer to Figure 5.5 (a). Let $|o_i o_j| = s$. We intend to compute a range for the variable $\bar{t}$ on $DC(l_{ij}^{rr})$.

$$s^2 = [r_j + V_j(\bar{t} + t) - r_i - \bar{t}]^2 + (V_R t)^2$$

Solving this equation, we obtain:

$$t = \frac{A(\bar{t}) \pm \sqrt{D(\bar{t})}}{V_j^2 + V_R^2}$$

Figure 5.5: Two discs C_i and C_j , with $|o_i o_j| = s$. The speeds of C_i , C_j and the robot are $V_i (= 1)$, V_j , and V_R , respectively. The robot starts to travel from C_i to C_j along the bold tangents at time $\bar{t}$. In (a) the bold tangent is a right-right tangent, and in (b), it is a right-left tangent.

where,

$$\begin{cases} A(\bar{t}) = -V_j[(V_j - 1)\bar{t} + (r_j - r_i)] \\ D(\bar{t}) = -V_R^2(V_j - 1)^2\bar{t}^2 - 2V_R^2(V_j - 1)(r_j - r_i)\bar{t} + s^2(V_j^2 + V_R^2) - V^2(r_j - r_i)^2 \end{cases}$$

For a right-right tangent from C_i to C_j , the following inequalities must be satisfied:

$$\begin{cases} D(\bar{t}) \geq 0 & (1) \\ \bar{t} \geq 0 & (2) \end{cases} \quad (5.1)$$

Solving Equation 5.1, and because $\bar{t} \geq 0$, for the starting time $\bar{t}$ of l_{ij}^{rr} , it holds:

$$(a) \begin{cases} 0 \leq \bar{t} \leq \frac{s\sqrt{V_R^2 + V_j^2} - V_R(r_j - r_i)}{V_R(V_j - 1)} \\ \text{when } t \geq 0 \text{ and } V_j > 1 \end{cases} \quad (b) \begin{cases} 0 \leq \bar{t} \leq \frac{-s\sqrt{V_R^2 + V_j^2} - V_R(r_j - r_i)}{V_R(V_j - 1)} \\ \text{when } t \geq 0 \text{ and } V_j < 1 \end{cases} \quad (5.2)$$

We therefore obtain a time range for $DC(l_{ij}^{rr})$. Similarly, we can obtain a time range for a right-left departure curve (this is also suitable for the case when $V_i = V_j$).

Figure 5.6: The logarithmic spiral around a growing disc which grows at speed 1. The speed of the spiral is V_R . A point on the spiral is specified as $(r(t), \Theta(t))$, and $r(t) = t$.

The details of the computation are in Section A.1 in Appendix A.

$$0 \leq \bar{t} \leq \frac{s\sqrt{V_R^2 + V_j^2} - V_R(r_j + r_i)}{V_R(V_j + 1)}. \quad (5.3)$$

□

So far, we have shown that when $V_j > V_i$ and $V_j < V_i$, $DC(l_{ij}^{rr})$ (resp. $DC(l_{ij}^{ll})$) is a continuous curve with respect to time. We also extend the departure curve computation of van den Berg and Overmars, and obtain the formulas of $DC(l_{ij}^{rr})$ and $DC(l_{ij}^{ll})$ in the polar coordinate system. The details of the computation are presented in Section A.2.

Let C_i be centered at $(0, 0)$, with radius 0 and speed 1, and C_j be centered at (a, b) , with radius r_j and speed V_j , $V_j \neq 1$. The trace of the robot travelling around disc C_i at speed V_R is a logarithmic spiral. The equation of spiral is represented in the polar coordinates $(r(t), \Theta(t))$, with parameter t , refer to Figure 5.6 for an example.

We have: $r(t) = t$. For the $\Theta(t)$, we have:

$$\sqrt{x'(t)^2 + y'(t)^2} = V_R \quad (5.4)$$

$$\begin{cases} x(t) = r(t) \cos \Theta(t) \\ y(t) = r(t) \sin \Theta(t) \end{cases} \quad (5.5)$$

$$\begin{cases} \Theta(t) = \pm \sqrt{V_R^2 - 1} \log t + \Theta_0 \\ r(t) = t \end{cases} \quad (5.6)$$

The $\pm$ in Equation 5.6 indicates the spiral is counterclockwise (+), or clockwise (−) around the growing disc. Θ_0 is the starting angle of the spiral. The growing disc C_i can be represented by two parameters time T and Θ :

$$C_i(T, \Theta) \longrightarrow \{T \cos \Theta, T \sin \Theta, T\}$$

Considering the CCW spirals around C_i . Each of the spirals is uniquely defined by the initial angle Θ_0 . Each point (T, Θ) on the growing disc C_i has a unique spiral that goes through that point. This spiral can be found by solving $\Theta(T) = \Theta$ for Θ_0 .

$$\begin{aligned} \Theta(t) &= \sqrt{V_R^2 - 1} \log t + \Theta_0 \\ \Theta_0 &= -\sqrt{V_R^2 - 1} \log T + \Theta \end{aligned}$$

Hence, the spiral through (T, Θ) is described in Euclidean coordinate

$$\begin{cases} x(t) = t \cos(\sqrt{V_R^2 - 1} \log t - \sqrt{V_R^2 - 1} \log T + \Theta) \\ y(t) = t \sin(\sqrt{V_R^2 - 1} \log t - \sqrt{V_R^2 - 1} \log T + \Theta) \end{cases} \quad (5.7)$$

If we walk along this spiral, we can depart for another disc C_j , if the straight line segment tangent to this spiral is tangent to C_j as well. The straight line segment l tangent to the spiral at point (T, Θ) is represented by:

$$\begin{aligned} l(t) &= \{x(T) + (t - T)x'(T), y(T) + (t - T)y'(T)\} \\ l(t) &= \{T \cos \Theta - (t - T)\sqrt{V_R^2 - 1} \sin \Theta, T \sin \Theta + (t - T)\sqrt{V_R^2 - 1} \cos \Theta\} \end{aligned} \quad (5.8)$$

The tangent $l(t)$ must be tangent to disc $C_j = (o_j, r_j, V_j)$, in order for point (T, Θ) to be on a departure curve $DC(l_{ij})$. The function of C_j is:

$$C_j : (x - a)^2 + (y - b)^2 = (V_j t + r_j)^2, \text{ where, } o_j = (a, b).$$

We fill the above equation by substituting $x = l_x(t)$ and $y = l_y(t)$ in Equation 5.8, and solve for t . We get a solution of the following form:

$$t_{1,2} = A(T, \Theta) \pm \sqrt{D(T, \Theta)}$$

When $D(T, \Theta) = 0$, l is tangent to C_j . By solving $D(T, \Theta) = 0$ for T , we obtain the functions of the departure curves. (The arrival time on C_j is given by $A(T, \Theta)$.) The functions of the departure curves are as the following (see Section A.2 for the details):

$$\begin{cases} \bar{t} = AE \sin(\theta + \alpha + \beta) + \frac{\sqrt{V_R^2 - V_j^2} - V_j \sqrt{V_R^2 - 1}}{\sqrt{V_R^2 - 1}(V_j^2 - 1)} r_j \\ A = \sqrt{a^2 + b^2}, \sin \alpha = \frac{a}{A}, \cos \alpha = \frac{b}{A} \\ E = \left| \frac{\sqrt{V_R^2 - V_j^2} - V_j \sqrt{V_R^2 - 1}}{\sqrt{V_R^2 - 1}(V_j^2 - 1)} \right|, \cos \beta = \frac{V_j \sqrt{V_R^2 - V_j^2} - \sqrt{V_R^2 - 1}}{\sqrt{V_R^2 - 1}(V_j^2 - 1)}, \sin \beta = \frac{1}{E} \end{cases} \quad (5.9)$$

and

$$\begin{cases} \bar{t} = AE \sin(\theta + \alpha + \beta) + \frac{\sqrt{V_R^2 - V_j^2} - V_j \sqrt{V_R^2 - 1}}{\sqrt{V_R^2 - 1}(V_j^2 - 1)} r_j \\ A = \sqrt{a^2 + b^2}, \sin \alpha = \frac{a}{A}, \cos \alpha = \frac{b}{A} \\ E = \left| \frac{\sqrt{V_R^2 - V_j^2} - V_j \sqrt{V_R^2 - 1}}{\sqrt{V_R^2 - 1}(V_j^2 - 1)} \right|, \cos \beta = \frac{-V_j \sqrt{V_R^2 - V_j^2} - \sqrt{V_R^2 - 1}}{\sqrt{V_R^2 - 1}(V_j^2 - 1)}, \sin \beta = \frac{1}{E} \end{cases} \quad (5.10)$$

In case $V_i = V_j$, i.e., $V_j = 1$, we obtain $DC(l_{ij}^{rr})$:

$$\begin{cases} \bar{t} \geq 0 \\ a \cos \theta + b \sin \theta + r_j = 0 \\ AB \sin(\theta + \alpha + \beta) \geq 0, \text{ where, } A = \sqrt{a^2 + b^2}, B = V_R, \cos \beta = 1/V_R, \sin \beta = \frac{\sqrt{V_R^2 - 1}}{V_R} \end{cases}, \quad (5.11)$$

which implies that $DC(l_{ij}^{rr})$ is a straight line, where $\bar{t}$ is increasing and θ is a constant. We also obtain $DC(l_{ij}^{rl})$:

$$\begin{cases} \bar{t} \geq 0 \\ \bar{t} = A \cos \theta + B \sin \theta + C, \text{ where, } A = \frac{a(V_R^2 - 2) - 2b\sqrt{V_R^2 - 1}}{2(V_R^2 - 1)}, B = \frac{b(V_R^2 - 2) + 2a\sqrt{V_R^2 - 1}}{2(V_R^2 - 1)}, C = \frac{-V_R^2 r_j}{2(V_R^2 - 1)} \end{cases} \quad (5.12)$$

In the following, we use $DC(l_{ij})$ to represent the portion on $DC(l_{ij})$ where l_{ij} exists.

Property 5.1. *Let C_i and C_j in $\mathcal{D}$ grow either at different speeds or at the same speed. Then, $DC(l_{ij})$ is a monotone function both in polar coordinate time T and angle Θ . Let C_k be a third disc in $\mathcal{D}$. If $DC(l_{ij})$ and $DC(l_{ik})$ intersect each other, then they either have one common intersection point, or have identical functions.*

Proof. By Lemma 5.4, each departure curve exists in a time range. By Equations 5.9, 5.10, 5.11, 5.12, and the fact that time $\bar{t}$ increases, $DC(l_{ij})$ is therefore monotone. When $DC(l_{ij})$ and $DC(l_{ik})$ have one segment in common, the two functions are the same. This is because that the departure curves are sine curves, and each of them has a period 2π . Suppose that the two functions are different. Let the function of $DC(l_{ij})$ be $\bar{t}_1 = A_1 E_1 \sin(\theta + \alpha_1 + \beta_1) + c_1 = a_1 \sin \theta + b_1 \cos \theta + c_1$, and the function of $DC(l_{ik})$ be $\bar{t}_2 = A_2 E_2 \sin(\theta + \alpha_2 + \beta_2) + c_2 = a_2 \sin \theta + b_2 \cos \theta + c_2$, where, a_1, b_1, a_2, b_2, c_1 and c_2 are constants. We take the negative value of $\bar{t}$ for $DC(l_{ik})$, then simply overlay the two functions. We obtain a new sine curve function as: $\bar{t}_3 = a_3 \sin(\theta + \gamma) + c_3$. We claim that this curve intersects with the Θ -axis only once, as the three functions have the same $\sin(\theta)$ part, and the period of each of them is 2π . $\square$

In this chapter, we assume that no two departure curves have the same function. Otherwise, we can always break the tie by processing them one by one.

5.2.2 Nearest-block disc problem

In this section, we present a pre-processing step which is required for Section 5.6. The discs in $\mathcal{D}$ grow at the same speed. Suppose that we are given all tangents originating at a common disc, say C_0 . Let $L = pq$ be tangent to C_0 at point $p \in C_0$, such that $L \parallel l_{0i}^l(t_1)$, and $|l_{0i}^l(t_1)| = |L|$, $t_1 > T_0$. Refer to Figure 5.7. The tangent $l_{0i}^l(t_1)$ has the following property: the ending point w of $l_{0i}^l(t_1)$ is located inside V_{C_i} . We call L the *supporting tangent* of $l_{0i}^l(t_1)$, written as $(L, l_{0i}^l(t_1))$. Assume that L is always valid. We aim to compute the *nearest-block* C_R for $(L, l_{0i}^l(t_1))$: C_R is a right-block of

Figure 5.7: L is tangent to C_0 , $p \in C_0$, $O_i q \perp L$. The ending point w of $l_{O_i}^l(t_1)$ is located inside V_{C_i} , where $|l_{O_i}^l(t_1)| = |L|$. The dotted curves are Voronoi edges. The dashed disc is centered at w , and is tangent to C_i .

L , and C_R is either (1) a left-block of $l_{0i}^l(T_0)$, or (2) an obstacle on $l_{0i}^l(T_0)$. It is easy to see that the nearest-block C_R for $(L, l_{0i}^l(t_1))$ can be obtained by walking along the tangents around C_0 from L in CW order. But this may not be efficient. For each valid $l_{0i}^l(T_0)$, we compute its left-closest-block C_f . This can be computed in $O(n \log n)$ time by adapting the solution to all-nearest-smaller-value problem described in [20]. Based on this, in $O(\log n)$ time, we can detect if C_R exists for $(L, l_{0i}^l(t_1))$. In case l_{0i}^l in the pair $(L, l_{0i}^l(t_1))$ is invalid at time T_0 , we find the valid CCW neighbor l_{0u}^l of l_{0i}^l instead, with $\text{slope}(l_{0u}^l) > \text{slope}(l_{0i}^l)$. We compute the left-closest-block C_f of l_{0i}^l with $\text{slope}(l_{0f}^{rr}) \geq \text{slope}(l_{0u}^l)$. Thus, we can always treat l_{0i}^l in the pair $(L, l_{0i}^l(t_1))$ as being valid at T_0 . In the following, when we talk about l_{0i}^{rr} , l_{0i}^l and the CCW/CW order, the order is understood to be around C_0 .

We consider all valid tangents around C_0 . Let $\mathbb{B}$ be the set of valid right-right tangents around C_0 . Without loss of generality, assume that $l_{01}^{rr} \in \mathbb{B}$ has minimum length among all tangents in $\mathbb{B}$. In CW order around C_0 , for each l_{0i}^{rr} , compute l_{0j}^{rr} , where, C_j is a left-closest-block of l_{0i}^{rr} . A stack S is used during the computation. At any time, the stack holds the right-right tangents, from l_{01}^{rr} to l_{0m}^{rr} . Tangent l_{0m}^{rr} is the top element in current S . For any $l_{0k}^{rr} \in S$, $2 \leq k \leq m$, C_{k-1} is the left-closest-block of l_{0k}^{rr} (providing that C_{k-1} is a left-block of l_{0k}^{rr}). (Note, for simpler illustration, we just give indices 1 to m to the tangents in the stack S in order. In reality, the tangents may have different indices.) The procedure is as follows. First, l_{01}^{rr} is pushed onto the stack, which has minimum length among all tangents in $\mathbb{B}$. If C_2 , where, l_{02}^{rr} is the CW neighbor of l_{01}^{rr} , is not lower than C_1 along l_{01}^{rr} , then push l_{02}^{rr} onto S , and so on. If at some moment, C_i is lower than C_m along l_{0m}^{rr} , then pop l_{0m}^{rr} off. This step is repeated until l_{0i}^{rr} is pushed onto S . As all valid tangents are processed in CW order, a valid l_{0i}^l may be processed at a moment. We treat it in the same way as processing a valid right-right tangent, but without pushing l_{0i}^l onto S . (The left-closest-block of l_{0i}^l is disc C_m , providing that C_m projects on l_{0i}^l , where, l_{0m}^{rr} is the top element in current S .) We then compute C_R for $(L, l_{0i}^l(t_1))$ in the current stack S , and the details of this step are introduced later. The algorithm is presented in Algorithm 8.

Figure 5.8: (a): $\text{slope}(l_{0j}^l) > \text{slope}(l_{0i}^r)$, and C_j is lower than C_i along l_{0j}^l . (b): $\text{slope}(l_{0j}^l) < \text{slope}(l_{0i}^r)$, and C_j is lower than C_i along l_{0i}^r . (c): $\text{slope}(l_{0j}^l) < \text{slope}(l_{0i}^r)$, and C_j is higher than C_i along l_{0i}^r .

Lemma 5.5. *Let S be the current stack in Algorithm 8 during the process of l_{0j}^l . If the left-closest-block C_f of l_{0j}^l exists, then $l_{0f}^r \in S$.*

Proof. We first claim that once a tangent l_{0h}^r is popped off the stack S while processing a valid tangent l_{0i} , C_h is not the left-closest-block C_f of any l_{0j}^l processed later, where, $\text{slope}(l_{0j}^l) < \text{slope}(l_{0i})$.

Without loss of generality, we assume that $l_{0i} = l_{0i}^l$ (when $l_{0i} = l_{0i}^r$, analogous argument can be applied). To simplify the proof, assume that no two tangents have the same slope. Let the tangents popped off from the stack be $\mathbb{A} = \{l_{0m}^r, l_{0(m-1)}^r, \dots, l_{0k}^r\}$ in order, $m > k$, when processing a valid l_{0i}^l , $h \in \{m, m-1, \dots, k\}$. Then, $\text{slope}(l_{0h}^r) > \text{slope}(l_{0i}^l)$, as l_{0h}^r is processed before l_{0i}^l . There are several possible cases for the location of a valid l_{0j}^l with $\text{slope}(l_{0j}^l) < \text{slope}(l_{0i}^l)$. Refer to Figure 5.8.

(1): $\text{slope}(l_{0j}^l) > \text{slope}(l_{0i}^r)$, and C_j is lower than C_i along l_{0i}^l . Since C_i is lower than C_h along l_{0h}^r , C_j must be lower than C_h along l_{0h}^r , $C_h \neq C_f$.

(2): $\text{slope}(l_{0j}^l) < \text{slope}(l_{0i}^r)$, and C_j is lower than C_i along l_{0i}^r . No matter if l_{0i}^r is valid or invalid, by the same argument as made in Case (1), C_j must be lower than

C_h .

(3): $\text{slope}(l_{0j}^{rl}) < \text{slope}(l_{0i}^{rr})$, and C_j is higher than C_i along l_{0i}^{rr} . If l_{0i}^{rr} is valid, then $C_i = C_f$ ($C_i \neq C_h$). If l_{0i}^{rr} is invalid, then there exists a disc C_z , with $\text{slope}(l_{0j}^{rl}) < \text{slope}(l_{0z}^{rr}) < \text{slope}(l_{0i}^{rr})$, and C_z is lower than C_i along l_{0i}^{rr} , with valid l_{0z}^{rr} . Thus, before processing the valid l_{0j}^{rl} , l_{0z}^{rr} must have been processed. By the same argument as in Case (1), C_z must be lower than C_h . $C_z = C_f$.

Let B be a list of tangents popped off the stack S during l_{0j}^{rl} is processed. For any $l' \in B$, the disc corresponding to l' is not the left-closest-block of l_{0j}^{rl} , as otherwise, l' would have been kept in S . Combining the above with the claim made at the beginning of the proof, the lemma is derived. $\square$

We obtain the right-right tangents in stack S after processing each l_{0i}^{rl} . Now we compute the nearest-block C_R for $(L, l_{0i}^{rl}(t_1))$ using the information in S . We search the tangents in S for the CW neighbor of L , say $l_{0v}^{rr} \in S$, with $\text{slope}(l_{0i}^{rl}(T_0)) < \text{slope}(l_{0v}^{rr}) < \text{slope}(L)$.

Lemma 5.6. *If the nearest-block C_R for $(L, l_{0i}^{rl}(t_1))$ exists, then C_R is in the current stack S right after processing l_{0i}^{rl} , and $C_R = C_v$, where, l_{0v}^{rr} is the CW neighbor of L in S .*

Proof. We claim that if l_{0v}^{rr} exists, and C_v projects on L , then $C_v = C_R$. Otherwise, $C_R = \emptyset$.

The case when C_v does not exist is easy. If C_v exists, $l_{0v}^{rr} \in S$ is angularly closer to L in CW order than any other tangents in the current stack S . If C_v projects on L , by the definition of C_R , C_v is C_R for $(L, l_{0i}^{rl}(t_1))$. If C_v does not project on L , then C_h does not project on L , where, $l_{0h}^{rr} \in S$, and $\text{slope}(l_{0i}^{rl}) < \text{slope}(l_{0h}^{rr}) < \text{slope}(l_{0v}^{rr})$. This is because the ending point w of $l_{0i}^{rl}(t_1)$ is located inside V_{C_i} . Since $|l_{0i}^{rl}(t_1)| = |L|$, $L \parallel l_{0i}^{rl}(t_1)$, $|o_i w - r_i(T_0)| > |wq|$. No disc in $\mathcal{D}$ is inside disc C centered at w , with radius $|o_i w - r_i(T_0)|$. The disc corresponding to a tangent in S which is higher than C_h along l_{0h}^{rr} is outside C and above $o_i q$. Refer to Figure 5.7. Furthermore, since $o_i q \perp L$, the discs corresponding to the tangents which are popped off S , are above $o_i q$. Thus, $C_R = \emptyset$. Therefore, once C_R exists, $C_R = C_v$. $\square$

Lemma 5.7. *For all $(L, l_{0i}^l(t_1)) \in \mathbb{L}$, the nearest-block C_R can be computed in $O(n \log n + |\mathbb{L}| \log n)$ time in total by Algorithm 8.*

Proof. First we establish the correctness of Algorithm 8. By Lemma 5.5, stack S holds the valid right-right tangents in CW order with the following property: for every two neighboring tangents, say, l_{0h}^{rr} and $l_{0(h-1)}^{rr}$ in S , C_{h-1} is the left-closest-block of l_{0h}^{rr} . S is built by Steps 4-9. By definition of C_R , C_R for $(L, l_{0i}^l(t_1))$ exists only between L and $l_{0i}^l(T_0)$ in CW order. Since after processing l_{0i}^l , stack S holds all tangents which have the above relationship, by Lemma 5.6, if C_R exists, then l_{0R}^{rr} is in S . Steps 10-12 compute C_R for $(L, l_{0i}^l(t_1))$ according to Lemma 5.6.

Next we give the time complexity analysis. Each valid tangent is pushed onto and popped off the stack at most once. Since there are $O(n)$ tangents, sorting the tangents takes $O(n \log n)$ time. Searching the tangents in S for $l_{0w}^{rr} \in S$ in Step 11 takes $O(\log n)$ time. The number of the nearest-blocks to be computed is $|\mathbb{L}|$. $\square$

5.3 An overview of our approach

Our approach to solving the SPAGD problem is based on constructing a graph which, during run-time, contains all information required to solve the problem. In contrast to [114, 115] though, our graph has substantially fewer vertices and edges, for the same-speed case. We first describe the graph construction of van den Berg and Overmars. In their cone-based approach, a cone in 3-dimension represents a disc growing over time, where, time is the third dimension. Each departure curve is cut by at most $n-1$ cones thus defining $O(n)$ intervals. For each interval, only the collision free path that arrives at the interval earliest is kept. Each interval becomes a *node* in a graph G . Each node is incident to two *edges*: a collision free tangent between two cones (possibly followed by a spiral), and a spiral between two neighbouring departure curves on a cone. The edges in G are obtained by computing the arrangement of the cones and the departure curves, and the trapezoidal map (i.e., the sides of the trapezoids are spirals, and the top and bottom edges are departure curves) for the arrangement.

There are $O(n^3)$ nodes and $O(n^3)$ edges in the graph G , thus applying Dijkstra's algorithm they obtain an $O(n^3 \log n)$ solution.

Our technique is based on the observation that the projection of the intersection of the set of cones, is a Voronoi diagram of a set of discs in 2- d . By using the Voronoi diagram, we avoid the computation costs of the arrangement of the set of cones. For a departure curve, we compute its valid buffer(s) which has the property that any shortest path passing through a departure curve must pass through a valid buffer. We efficiently pre-compute the valid buffers on each departure curve. By employing valid buffers, we can decrease the number of nodes and edges in the graph G to $O(n^2)$, and thus are able to reduce the existing $O(n^3 \log n)$ solution to $O(n^2 \log n)$, for the same-speed discs case. With the various-speed discs, we can still construct the graph G , with $O(n^3)$ nodes and edges. The main steps of our algorithm are presented in Algorithm 9.

Since the spirals computation is the same as the ones of van den Berg and Overmars, we omit it here. We focus on the computation of valid buffers. Recall that there are four types of tangents between any two discs. We will describe the computation of valid buffers on right-right and right-left departure curves (left-right and left-left departure curves are handled analogously).

We call C_0 the disc which contains the starting points of the tangents. Around disc C_0 , we compute the arrangement of the departure curves in polar coordinates. Each departure curve is part of a sine curve, composed of the starting points of a right-right (resp. right-left) tangent from disc C_0 , to disc C_i , at different time t . Each departure curve is monotone, and every two departure curves can intersect at most once. After computing the arrangement of the departure curves, the valid buffers on a departure curve are determined either (1), by the intersection points on the departure curve in the arrangement, or (2), by the Voronoi edges which cut the departure curve, or (3), by the Voronoi edges which cut the corresponding arrival curve. Thus, at each time-sorted intersection point q on the departure curve, we test whether the tangent is valid or not with starting point q . If q is valid, then a valid buffer exists around q .

By Lemma 5.3, at any time, the Voronoi edges play a role in determining the valid buffers. With the same-speed discs, we do not check each intersection point on a departure curve. We stop the valid buffer computation on a departure curve segment $[t_1, t_2]$ at a crucial time, say t_q , such that after t_q , there is no valid tangent with starting point located on $[t_1, t_2]$. Thus, there are less event points on each departure curve in the same speed case.

Each valid buffer corresponds to a tangent portion in E_2 . How are the valid buffers used during running Dijkstra shortest path algorithm? It is adapted from van den Berg and Overmars' variation of Dijkstra's algorithm. The following is the outline of running Dijkstra's algorithm. The details are presented in Section 5.8.

Starting from s , following a tangent (possibly followed by a spiral), we pick the departure curve μ on a disc C at which the robot has arrived earlier. There are two edges originating from the arriving point p at time t_p , $p \in \mu$: one is a piece of spiral ending at the next departure curve on C , if the piece of spiral is totally located inside the Voronoi cell of C ; one is a tangent (possibly followed by a spiral) ending at a departure curve on another disc C' , if μ is associated with a valid buffer, and t_p is inside the buffer. An arriving time is associated with the new departure curve reached so far. Each time, the node (departure curve) which is arrived earliest is picked, and so on. To compute the next departure curve on C reached by a spiral, or the next departure curve on C' reached by a spiral on C' following a tangent between C and C' , we apply the step of locating the arriving point in a trapezoid stated in [115], summarized in Section 2.2.2. The trapezoid is built upon the arrangement of the departure curves on C , or C' . The tangents originate from the points on the valid buffers we compute. With the same-speed discs, we use valid buffers instead of the departure curves when computing the arrangement of the departure curves.

In summary, the C-SPAGD problem can be solved in $O(n^2 \log n)$ time, and the V-SPAGD problem can be solved in $O(n^3 \log n)$ time, where, n is the number of discs. We will prove this statement in the remainder of this chapter.

Figure 5.9: (a) $\text{slope}(l) < \text{slope}(l_{0j}^{rr})$. (b) Tangent $l = l_{0i}^l$ is located between l_{0j}^{rr} and l_{0j}^l in CCW order, C_j is lower than C_i along l , l is invalid. The growth of the discs is not shown in the figure.

5.4 Valid tangent computation

In this section, we present our algorithm for computing all valid tangents originating from a particular disc C_0 , when discs grow at the same speed. We then repeat the same procedure for all other discs in the set $\mathcal{D}$. We aim to compute all valid tangents around C_0 in $O(n \log n)$ time. (For the various-speed case, we check the validity of a tangent with respect to all other discs in $\mathcal{D}$, and thus, the valid tangents around C_0 can be computed in $O(n^2)$ time.)

Lemma 5.8. *Let C_0 , C_i and C_j be three distinct discs in $\mathcal{D}$. Let $l(t)$, be a right-right, or a right-left tangent from C_0 to C_i , with starting time t . Disc C_j has no effect on the validity of $l(t)$ if $\text{slope}(l_{0j}^{rr}(t)) > \text{slope}(l(t))$, or $\text{slope}(l_{0j}^l(t)) < \text{slope}(l(t))$. If $l(t)$ is located between $l_{0j}^{rr}(t)$ and $l_{0j}^l(t)$ in CCW order, where C_j is lower than C_i along $l(t)$, then $l(t)$ is invalid.*

Proof. For simplicity of presentation, we omit t in the proof. We prove that disc C_j has no effect on the validity of l in the case of $\text{slope}(l_{0j}^{rr}) > \text{slope}(l)$. The case of $\text{slope}(l_{0j}^{rl}) < \text{slope}(l)$ can be proved similarly. Refer to Figure 5.9 (a), l is to the right of l_{0j}^{rr} . Without loss of generality, let $l = l_{0i}^{rr}$. If $\text{slope}(l_{0j}^{rr}) = \text{slope}(l)$, then the robot will touch C_j at point p at time t_p , where p is the projection of o_j on l . Therefore, when $\text{slope}(l_{0j}^{rr}) > \text{slope}(l)$, there is no intersection point between C_j and l at time t_p . By Definition 5.1, C_j has no effect on the validity of l . When l is located between l_{0j}^{rr} and l_{0j}^{rl} in CCW order, and C_j is lower than C_i along l , tangent l must intersect disc C_j . Refer to Figure 5.9 (b). $\square$

Note that Lemma 5.8 can also be applied to the case when the discs grow at different speeds.

Next, we present Algorithm 10 which computes all valid tangents. We compute all tangents from C_0 to each disc and sort them in angular order around C_0 . For simplicity, we can assume that no two tangents have the same slope. Otherwise, we break ties by using their lengths as secondary key. The input is the set of tangents, stored in a balanced binary tree according to their slopes. The output is the set of all valid tangents. First we assume that C_0 is a point, $|l_{0i}^{rr}| = |l_{0i}^{rl}|$. Since $r_0 = 0$ when C_0 is a point, $|l_{0i}^{rr}|^2 + (r_i + |l_{0i}^{rr}|/V_R)^2 = |o_i o_0|^2$. This is also the equation for $|l_{0i}^{rl}|$. Then we generalize our arguments and algorithms to the case that C_0 is a disc. We divide the plane into four quadrants around C_0 (if a tangent happens to be a boundary edge of a quadrant, then the validity of the tangent can be detected by checking if the tangent intersects the discs). We consider the discs in each quadrant in increasing order of length of the tangents. Let C_i be the disc under consideration. We remove all tangents that lie between l_{0i}^{rr} and l_{0i}^{rl} , and also decide the validity of l_{0i}^{rr} and l_{0i}^{rl} .

In the following, Lemmas 5.9 and 5.10 establish the correctness and time complexity of Algorithm 10, respectively.

Lemma 5.9. *Algorithm 10 computes the valid tangents around C_0 correctly.*

Proof. Algorithm 10 maintains the following loop invariant:

Let $\mathbb{T}$ be the current tree when processing tangent l_{0i} ($l_{0i} = l_{0i}^{rr}$ or $l_{0i} = l_{0i}^l$). After l_{0i} is processed, the tangents in $\mathbb{T}$ satisfy the following properties: (1) all tangents in $\mathbb{T}$ have lengths $\geq |l_{0i}|$, (2) C_i has no effect on the validity of the tangents in $\mathbb{T}$, (3) L' contains l_{0i} , if l_{0i} was not deleted in the previous events, and the tangent point on C_i is located in V_{C_i} .

At the beginning, all tangents in $\mathbb{T}$ are sorted with respect to slope.

Next, we show that each iteration maintains the loop invariant. (1) The tangents whose lengths are less than $|l_{0i}|$ have been processed and have been removed from the current tree $\mathbb{T}$. (2) Draw a circle C centered at o_0 , with radius $\sqrt{|l_{0i}|}$ ($r_0 = 0$). Any tangent l in the current $\mathbb{T}$, whose ending point $p \notin C$, and for which $\text{slope}(l_{0i}^{rr}) < \text{slope}(l) < \text{slope}(l_{0i}^l)$, must intersect C_i . This is because that all the points on the current sweeping circle that are located between l_{0i}^{rr} and l_{0i}^l in CCW order are located inside C_i , and l passes through one of those points. Hence, l is invalid. After Step 6, the tangents whose validity are affected by C_i have been deleted. (3) If l_{0i} was not deleted before, then the examined event discs are not obstacle discs on l_{0i} , by Lemma 5.8. We have to prove that if the tangent point on C_i is located in V_{C_i} , then the other event discs after processing l_{0i} , are not obstacle discs on l_{0i} . We claim that the projections of the other discs C_j , which are not processed yet, with $\text{slope}(l_{0j}^{rr}) < \text{slope}(l_{0i}) < \text{slope}(l_{0j}^l)$, are not on l_{0i} . Let C be the current sweeping circle passing through the end point q of l_{0i} . Then, o_j , the end points of l_{0j}^l and l_{0j}^{rr} are all located out of C . Thus, the projection of o_j is not on l_{0i} , and q is the closest point among all the points on l_{0i} to o_j . Since q is located in V_{C_i} , by Definition 5.1, l_{0i} is valid. The valid tangents are put in L' in Step 8.

Finally, when $Q = \emptyset$, all tangents have been processed, and L' holds all the valid tangents. $\square$

Lemma 5.10. *Algorithm 10 computes all valid tangents originating from point disc C_0 at time T_0 in $O(n \log n)$ time.*

Proof. There are $O(n)$ events in total in Step 3. At each event, searching and inserting the two tangents take $O(\log n)$ time in Step 5. Each tangent is inserted once, and

Figure 5.10: Sweep circle C passes through event point p , which is the end point of l_{0i}^l . $C \cap C_i = \{p, q\}$. Segment $\hat{l}$ is tangent to C_0 , $|\hat{l}| = d$, with endpoint q . Sweep circle C' passes through the ending point of l_{0i}^{rr} .

deleted at most twice in Step 6 and Step 10. □

Algorithm 10 employs a radial sweep, centered at o_0 , with radius $\sqrt{r_0^2 + |l_{0i}^{rr}|^2} = \sqrt{|l_{0i}^{rr}|}$. We call the circle *sweep circle*. The sweep events are the tangents from C_0 to C_i . When C_0 is a disc, every disc now has two tangent events associated with, with lengths $|l_{0i}^{rr}|$ and $|l_{0i}^l|$, respectively. We claim that $|l_{0i}^{rr}| > |l_{0i}^l|$. This is because that $|l_{0i}^l|^2 = |o_0 o_i|^2 - |r_0 + r_i|^2$, and $|l_{0i}^{rr}|^2 = |o_0 o_i|^2 - |r_0 - r_i|^2$. Let the sweep circles be C and C' at the two tangents events, refer to Figure 5.10. Let $\mathbb{A}$ be the set of tangents, whose slopes are between $\text{slope}(l_{0i}^{rr})$ and $\text{slope}(l_{0i}^l)$. It is possible that the end points of some tangents in $\mathbb{A}$ are located in $C' - C$. Thus, Step 5 and Step 6 cannot be applied directly. We need to find and insert a segment $\hat{l}$, which is tangent to C_0 , between l_{0i}^{rr} and l_{0i}^l in CCW order, such that $\hat{l}$ is the right-left tangent paired with l_{0i}^{rr} , and is the right-right tangent paired with l_{0i}^l . In this way, if there are tangent events

located in $C' - C$, then after sweeping circle C , those tangent events can be captured by sweeping the circle passing through them. The segment $\hat{l}$ is determined as follows:

Let d be the minimum of $|l_{0i}^{rr}|$ and $|l_{0i}^{rl}|$, which is $|l_{0i}^{rl}|$. Let $C_i(t_p)$ be the disc when the robot arrives at the tangent point on C_i along l_{0i}^{rl} . Let q be the other intersection point of $C_i(t_p)$ and C . Recall that C is the circle centered at o_0 , with radius $\sqrt{r_0^2 + |l_{0i}^{rl}|^2}$. The segment $\hat{l}$ is tangent to C_0 , with the other endpoint q , and $|\hat{l}| = d$.

Lemma 5.11. *All tangents that are located between l_{0i}^{rr} and $\hat{l}$ (resp. between $\hat{l}$ and l_{0i}^{rl}) in CCW order, with lengths $\geq |l_{0i}^{rr}|$ (resp. $\geq |l_{0i}^{rl}|$), are invalid.*

Proof. The proof follows the proof of loop invariant (2) in Lemma 5.9. $\square$

Algorithm 10 needs to be changed to accommodate $\hat{l}$. Each disc has two tangent events. Tangent $\hat{l}$ for each disc is inserted into the binary search tree. By Lemma 5.11, Step 5 is changed to locate the two tangents $\hat{l}$ and l_{0i}^{rl} (resp. l_{0i}^{rr} and $\hat{l}$) in the binary tree. There are in total $2n$ tangent events in the binary search tree. All other steps in Algorithm 10 are not changed. Hence, we obtain the following theorem:

Theorem 5.1. *When discs in $\mathcal{D}$ grow at the same speed, by applying Algorithm 10, all valid tangents around disc C_0 at time T_0 can be computed in $O(n \log n)$ time.*

5.5 Valid buffer computation for various-speed discs

In this section, we provide an algorithm that computes valid buffers on the right-left and right-right departure curves, with various-speed discs. We focus on computing valid buffers on the right-left and right-right departure curves around disc C_0 , and then repeat this for each disc in $\mathcal{D}$. The computation has two main steps. The first step computes the arrangement of the departure curves, refer to Figure 5.11 for an arrangement. The second step computes valid buffers on the segments defined by every two consecutive intersection points on a departure curve. Recall that as appropriate for the discussion, we use the starting time t_x of tangent $l_{0i}(t_x)$ to represent

Figure 5.11: An arrangement of the departure curves on disc C_0 is illustrated in the polar coordinate system, with the various-speed discs. The vertical line is $DC(l_{0i}^{rr})$, with $V_0 = V_i$. The rests are right-left or right-right departure curves, each of which is part of a sine curve.

point x on $DC(l_{0i})$. We define the *validity changing point* x of l_{0i} on $DC(l_{0i})$ as: at time $t_x - \epsilon$, l_{0i} is invalid, and at time $t_x + \epsilon$, l_{0i} becomes valid, and vice versa, where, $\epsilon > 0$ is sufficiently small. The intersection points and the valid buffers have the following properties. Unless specified, otherwise, all the departure curves are represented using polar coordinates.

Property 5.2. *Point q is a common point on $DC(l_{0i}) \cap DC(l_{0j})$, if and only if C_0 , C_i and C_j are co-tangent to $l_{0i}(t_q)$.*

Proof. Refer to Figure 5.12. $\Leftarrow$: Point q is the starting point of two tangents $l_{0i}(t_q)$ and $l_{0j}(t_q)$, respectively. Since q is the tangent point on C_0 , $l_{0i}(t_q) \perp o_0q$, and $l_{0j}(t_q) \perp o_0q$. Thus, $l_{0i}(t_q)$ and $l_{0j}(t_q)$ are co-linear.

$\Rightarrow$: Since C_0 , C_i and C_j are co-tangent, Point q is a tangent point of l_{0i} and l_{0j} on C_0 . Thus, point q is on $DC(l_{0i})$ and $DC(l_{0j})$. $\square$

Figure 5.12: For the proof of Property 5.2. The intersection point q of two departure curves using Cartesian coordinates. $DC(l_{0i}) \cap DC(l_{0j}) = q$. Tangent $l_{0i}(t_q)$ and $l_{0j}(t_q)$ are co-linear.

Corollary 5.2. *Let p and q be two consecutive intersection points on $DC(l_{0i})$, with $t_q > t_p$. Then no disc C_j exists, such that C_0 , C_i and C_j are co-tangent at time t , where, $t_p < t < t_q$.*

Proof. Otherwise, there is a third intersection point w corresponding to time t between p and q , by Property 5.2. $\square$

We define $q^*(0, j, i) = DC(l_{0j}) \cap DC(l_{0i})$, where, C_0 , C_j and C_i are co-tangent to $l_{0i}(t_q)$, with C_i higher than C_j along l_{0i} .

Lemma 5.12. *Let t_q be an intersection point of $DC(l_{0i})$ and $DC(l_{0j})$. Let $t_q^+ = t_q + \epsilon$, and $t_q^- = t_q - \epsilon$. (1) If t_q corresponds to $q^*(0, i, j)$, then, C_j has no effect on the validity of $l_{0i}(t_q^+)$. (2) If t_q corresponds to $q^*(0, j, i)$, and disc C_j is an obstacle on $l_{0i}(t_q^-)$,*

then C_j is not an obstacle on $l_{0i}(t_q^+)$; if C_j is not an obstacle on $l_{0i}(t_q^-)$, then C_j is an obstacle on $l_{0i}(t_q^+)$.

Proof. By Property 5.2, q is a common point on $DC(l_{0j}) \cap DC(l_{0i})$. (1) Let the ending point of l_{0i} be w . Since the tangent point on C_j is higher than the tangent point on C_i along $l_{0i}(t_q)$, $w \cap C_j = \emptyset$. Thus, for a sufficient small ϵ , C_j has no effect on the validity of $l_{0i}(t_q^+)$. This is also true for the same-speed discs. (2) Since C_j is an obstacle on $l_{0i}(t_q^-)$, by Lemma 5.8, holds $\text{slope}(l_{0j}^{rr}(t_q^-)) < \text{slope}(l_{0i}(t_q^-)) < \text{slope}(l_{0j}^{rl}(t_q^-))$. Since q is a point in $DC(l_{0j}) \cap DC(l_{0i})$, $\text{slope}(l_{0j}(t_q)) = \text{slope}(l_{0i}(t_q))$. After time t_q , the order of the two tangents, l_{0j} and l_{0i} , on C_0 changes. This is because that $DC(l_{0i})$ and $DC(l_{0j})$ can only intersect once, by Property 5.1. By Lemma 5.8, C_j is not an obstacle on $l_{0i}(t_q^+)$. A similar argument can be applied for the next statement of (2), which is also true for the same-speed discs. $\square$

Corollary 5.3. *Let point q be a validity changing point of l_{0i} on $DC(l_{0i})$. Then, q is either (1) $q^*(0, j, i)$ with some C_j , or (2) a point on a Voronoi edge of C_0 , or (3) the starting point of $l_{0i}(t_q)$, such that the point $C_i \cap l_{0i}(t_q)$ is on a Voronoi edge of C_i .*

Proof. By Lemma 5.12, the validity of l_{0i} changes when C_0 , C_j and C_i are co-tangent to $l_{0i}(t_q)$. Thus, q is an intersection point of two departure curves. By Lemma 5.1, the Voronoi edges of C_0 and C_i may change the validity of l_{0i} . $\square$

We define that point q is determined by a Voronoi edge of C_0 , $q \in DC(l_{0i})$, as: q is located on a Voronoi edge of C_0 . We also define that a point $q \in DC(l_{0i})$ is determined by a Voronoi edge of C_i as: the end point of $l_{0i}(t_q)$ on C_i is located on a Voronoi edge of C_i .

5.5.1 Algorithms

Based on Corollary 5.1 and Corollary 5.3, we are now able to state Algorithm 11.

By Corollary 5.1, every point in a valid buffer on $DC(l_{0i})$ must be located inside V_{C_0} . Furthermore, the corresponding arriving points on C_i of l_{0i} must be located

inside V_{C_i} . Based on Corollary 5.1, we elaborate Step 4 of Algorithm 11 next, and present Algorithm 12.

Lemma 5.13. *Algorithm 12 computes valid buffers correctly.*

Proof. We consider a departure curve $DC(l_{0i})$. In Step 1, the obstacle discs in $\mathcal{D}$ for $l_{0i}(T_0)$ are placed into $\mathbb{M}$. We do this by projecting each disc center onto $l_{0i}^r(T_0)$, and then compute the time needed for the robot to reach the projection point. After that we check if the projection point is located inside the disc. When $|\mathbb{M}| = 0$, l_{0i} is valid. By Corollary 5.3, the event points are (i) in S_{0i} , (ii) determined by the Voronoi edges of C_0 and of C_i . All the event points are stored in Q and time sorted. Steps 4-17 work on every two neighboring event points. Time t_p denotes the previous event, and t_q the current event. If at a previous event point p , no disc is in the middle set $\mathbb{M}$, then a valid buffer $[t_p, t_x]$ is defined. The time of the next intersection point of two departure curves, or as determined by Lemma 5.3, is t_x . The disc C_j , which determines the valid buffer, is put in $\mathbb{M}$. This is performed in Steps 6-8. Steps 10-15 deal with the case when $l_{0i}(t_p)$ is invalid according to Lemma 5.12: (1) if $C_j \in \mathbb{M}$, then C_j is deleted from $\mathbb{M}$; (2) if $C_j \notin \mathbb{M}$, then it is placed into $\mathbb{M}$. Thus, all the discs which may change the validity of l_{0i} , according to Corollary 5.3 are processed. $\square$

5.5.2 Time complexity analysis

In order to establish the time complexity analysis of Algorithm 12, we need to introduce some knowledge first.

5.5.2.1 Plane sweep

The plane sweep technique is applied to the arrangement of departure curves. The general plane sweep algorithm, refer to [111], (also see [36] and [55],) is stated as follows: Let S be a set of planar objects for which we want to solve a given problem using a plane sweep. We sweep a line SL over the elements of S in some specified direction. During the sweep, we dynamically maintain (information based on) the

Figure 5.13: An example of $\Theta(n^2)$ intersection points in the arrangement of the departure curves around disc C_0 . The speed of each disc C_i in the middle is $V_i \rightarrow 0$. $V_R \rightarrow \infty$. The speeds of other discs C_j , $V_j = V_0$. V_0 is big enough such that each C_i will touch l_{0j}^r once at different times without intersecting other C_i s. Imagining that the discs grow at the given speeds. For a specific C_j in the figure, l_{0j}^r is parallel to each other at different times, as $V_0 = V_j$. Since discs C_i grows at a very small speed, l_{0j}^r may be tangent to a C_i from the left side of C_i , and then is tangent to C_i from the right side of C_i .

intersection $SL \setminus S$, which is a one-dimensional scene, and, simultaneously, compute the desired information for parts of the set S . For any position of the sweep line SL , let SSL be the set of elements of S that intersect SL . These elements are maintained in a Y -structure, which depends on the problem to be solved. For many applications, the elements of SSL are maintained sorted with respect in y -coordinates of their intersections with the sweep line, and the Y -structure is kept as a balanced binary search tree storing SSL according to this order. The Y -structure changes if

1. the sweep line encounters an element of S . In that case, it is inserted (together with certain information) into the Y -structure,
2. the sweep line encounters the rightmost part of an element of S . In that case, it is deleted together with certain information from the Y -structure, or,
3. the relative order of the elements of SSL on the sweep line changes. In that case, elements together with the information are interchanged in the Y -structure.

The positions of the sweep line where the Y -structure changes are called the event points of the sweep. At the start of the sweep algorithm, certain transition points are known already. The remaining event points are discovered during the sweep itself. In order to maintain these event points, a dynamic data structure, called the X -structure is needed. This structure must support the following operations.

1. Insertion of a new event point.
2. Delete-min, i.e., find and delete the event point with minimum x -coordinate, which is the next event point of the sweep algorithm.
3. In some applications, it is able to delete an arbitrary event point.

In many applications, the X -structure is a balanced binary search tree or just a linked list or array, containing the event points sorted by their x -coordinates.

According to Property 5.1, the departure curves of the various-speed discs have functions of sine curves, and are monotone both in polar coordinate time T and

angle Θ . Two departure curves can intersect only once. Thus, we can apply plane sweep technique to compute the arrangement of the departure curves in the polar coordinate system as follows: The plane sweep is the along the T direction in the polar coordinate system. Two balanced binary search trees are needed: one is called T data structure, the other one is called Θ data structure. At the beginning, the sweep line is located at $T = 0$. The departure curves are installed in the Θ data structure, according to their Θ values at $T = 0$. The starting and ending points of the departure curves are the event points. All the event points are stored in the T data structure, according to their T values. At each event point, compute the intersection points of the corresponding departure curve and its two neighbors in the T data structure. The intersection points are also event points. At each event point, update the two data structures, e.g., insert or delete the corresponding departure curve in the Θ data structure, or exchange the order of two neighboring departure curves in the Θ data structure. Report the intersection event points.

5.5.2.2 Multiplicatively additively weighted Voronoi diagram

When discs grow at different speeds, the Voronoi diagram is called multiplicatively additively weighted Voronoi diagram. It is well known that this Voronoi diagram is the same as projecting the intersection of a set of cones onto the X - Y plane, see [90]. The cones are obtained as follows: for each disc $C_i \in \mathcal{D}$, cone_i is constructed with apex located at the center of C_i , and open angle $2 \arctan V_i$. At time $t = 0$, the cones are cut to the discs in $\mathcal{D}$. A Voronoi edge is a Descartes oval [31], a line, a hyperbola, or a circle. For details, see Section 2.2.1.3. As far as we know, there exists no efficient algorithm for computing this Voronoi diagram. We provide an $O(n^3)$ algorithm in Algorithm 13 by the incremental construction of the multiplicatively additively weighted Voronoi diagram on the cones.

Lemma 5.14. *Algorithm 13 computes the multiplicatively additively weighted Voronoi diagram of a set of various-speed discs in $O(n^3)$ time.*

Proof. The algorithm computes the Voronoi cell of each disc one by one. For each

Figure 5.14: $cone_0$ intersects $cone_1$ and $cone_2$ with two Descartes curves. Project the Descartes curves onto the $x-y$ plane. It forms the Voronoi cell V_0 for the corresponding disc C_0 , V_1 for C_1 , and V_2 for C_2 .

Figure 5.15: Doubly connected edge list data structure for a polygon with holes.

$cone_i$, other cones C_j intersect it one by one sequentially. For C_i and any other $cone_j$, the region that is closer to the corresponding disc C_i of $cone_i$ (the lower envelope of the arrangement of the cones defines the Voronoi diagram. See the details in Section 2.2.1.4) is computed in Step 3. After computing the intersections of the regions in Step 5, the regions for $cone_i$ are stored in V_i , and V_i is the Voronoi cell of C_i . Refer to Figure 5.14.

We claim that the Voronoi cell of C_i can be computed in $O(n^2)$ time, by using the doubly connected edge list data structure (DCEL, see [30] for a reference), refer to Figure 5.15 for an example. In our case, the boundary of a polygon is composed of curves, and a hole in the polygon may be enclosed by a curve, i.e., a circle or a Descartes curve. In this case, we can add one vertex on the curve if needed. When the projection of the intersection of two cones is a Descartes oval, at most four intersection points of two Descartes ovals (imagine the projections of two egg-like objects intersecting, also refer to Equation 2.1) are generated. Thus, there are $O(n^2)$ intersection points in the arrangement of the regions in Step 5. The intersection points can be computed by plane sweep. By using a DCEL data structure, V_i can be computed in $O(n^2)$ time, as the plane sweep takes $O(n \log n + k)$ time [27], where, k is the number of the intersection points, and n is the number of the vertices in the polygons. In our case, the number of vertices in the polygons in total cannot exceed $O(n)$, as there are $O(1)$ vertices on two cones intersection (two endpoints of a hyperbola or a point on a close curve). Therefore, for all the discs, the Voronoi cells are computed in $O(n^3)$ time. $\square$

5.5.2.3 Time complexity

Theorem 5.2. *Algorithm 12 computes valid buffers in $O(n^3 \log n)$ time.*

Proof. By Lemma 5.14, the additively multiplicatively weighted Voronoi diagram for the set of discs can be computed in $O(n^3 \log n)$ time by the incremental method.

For each C_0 , Step 1 takes $O(n^2)$ time, as each disc is intersected by l_{0i} once. By Property 5.1, every two departure curves intersect at most once (in case when

$DC(l_{0i})$ and $DC(l_{0j})$ have the same function, we can duplicate them, and treat them separately), the arrangement of departure curves can be computed in $O(n \log n + k)$ time, by applying general plane sweep technique along T direction into our polar coordinate system introduced in Section 5.5.2.1, where, k is bounded by $O(n^2)$. Refer to Figure 5.13 for an example.

Each disc may change the validity of l_{0i} at most once, therefore, the number of intersection points in S_{0i} associated with $DC(l_{0i})$ is $O(n)$. Thus, there are in total $O(n^2)$ event points to be considered in Step 2. Initially, the tangents around C_0 are stored in a balanced binary tree according to their slopes. When inserting or deleting a disc in $\mathbb{M}$, the slope of its corresponding tangent always keeps the original value. Thus, each deletion and insertion of a disc in $\mathbb{M}$ takes $O(\log n)$ time. Therefore, Step 6, 10 and 12 will each take $O(\log n)$. The time complexity is thus proven. $\square$

5.6 Valid buffer computation for same-speed discs

In this section, we discuss the problem instance of same-speed discs. We show how to compute valid buffers over the previous solution in $O(n^2 \log n)$ time. We achieve an $O(n)$ factor improvement due to the following facts: (a) the arrangement of partial departure curves around a disc C_0 can be computed in $O(n \log n)$ time, (b) for a valid $l_{0i}^{rr}(T_0)$, there is one valid buffer on its departure curve (no valid buffer exists when $l_{0i}^{rr}(T_0)$ is invalid), (c) an invalid $l_{0i}^{rl}(T_0)$ may become valid at some time t T_0 (refer to Figure 5.16). There are at most $O(\deg(V_{C_0}) + \deg(V_{C_i}))$ valid buffers on $DC(l_{0i}^{rl})$, where, $\deg(V_{C_0})$ and $\deg(V_{C_i})$ is the number of Voronoi edges of C_0 and C_i , respectively, and (d) the right-left valid buffers can be computed in $O(n^2 \log n)$ time.

We first show which disc may determine a right-right valid buffer and a right-left valid buffer. Then, we show when a disc C_j changes the validity of tangent l at time t_q , under what condition, C_j is also an obstacle disc on $l(t)$ with $t > t_q$.

Lemma 5.15. Consider $q^*(0, j, i)$. (1): If $l_{0i} = l_{0i}^{rr}$, then l_{0j} can only be l_{0j}^{rl} . (2): If $l_{0i} = l_{0i}^{rl}$, then l_{0j} can be either l_{0j}^{rl} or l_{0j}^{rr} .

Figure 5.16: An invalid right-left tangent becomes valid again. $l_{0i}^r(T_0)$ is invalid, while $l_{0i}^r(t)$ is valid. (a) $l_{0i}^r(T_0)$ is invalid because of C_j . (b) The departure curve on C_0 . The buffer candidates on the departure curve are located inside V_{C_0} .

Proof. (1) By contradiction. If $l_{0j} = l_{0j}^{rr}$, then by Lemma 5.2, C_0 , C_j and C_i are co-tangent to $l_{0i}(t)$, for any $t > t_q$. $DC(l_{0i})$ and $DC(l_{0j})$ are co-linear, contradicting that $q = DC(l_{0i}) \cap DC(l_{0j})$. (2) By Property 5.1, both $DC(l_{0j}^{rl})$ and $DC(l_{0i}^{rl})$ can intersect $DC(l_{0i}^{rl})$ in a point. $\square$

Lemma 5.16. *Given a point $q^*(0, j, i)$. If C_j projects on $l_{0i}(t_1)$, $t_1 > t_q$, then C_j is an obstacle disc on $l_{0i}(t_1)$.*

Proof. Assume that $l_{0i} = l_{0i}^{rl}$. By Corollary 5.3, C_j is an obstacle disc on $l_{0i}^{rl}(t_q + \epsilon)$, $\text{slope}(l_{0j}^{rr}) < \text{slope}(l_{0i}^{rl}) < \text{slope}(l_{0j}^{rl})$ at time $t_q + \epsilon$. We have to prove that the slope relationship is satisfied at any time $t_q < t \leq t_1$. Since the discs grow at the same speed, $l_{0j}^{rr}(t) \parallel l_{0j}^{rr}(t_q)$, and $l_{ji}^{ll}(\bar{t}) \parallel l_{ji}^{ll}(t_q)$, for $\bar{t} > t_q$. Thus, the CCW angle formed by $l_{0j}^{rr}(t)$ and $l_{0i}^{rl}(t)$, and the CCW angle formed by $l_{0i}^{rl}(t)$ and $l_{ji}^{ll}(\bar{t})$ is getting larger with t and $\bar{t}$ increasing, respectively. For the latter case, the robot arrives at the same C_i following $l_{0i}^{rl}(t)$ and $l_{ji}^{ll}(\bar{t})$, respectively. Therefore, we obtain $\text{slope}(l_{0j}^{rr}(t)) < \text{slope}(l_{0i}^{rl}(t))$. Also $\text{slope}(l_{0i}^{rl}(t)) < \text{slope}(l_{0j}^{rl}(t))$, as $l_{ji}^{ll}(\bar{t}) \parallel l_{ji}^{ll}(t_q)$ for the same-speed discs. Since C_j

projects on $l_{0i}(t_1)$, the order of the projection points of C_0 and C_j on $l_{0i}^l(t)$, does not change at different t . Therefore, C_j is an obstacle disc on $l_{0i}^l(t_1)$. A similar argument can be made for the case when $l_{0i} = l_{0i}^{rr}$. $\square$

Define *minimum segment* $[t_1, t_2]$ as a segment on $DC(l_{0i})$ with the following property: all the points on $[t_1, t_2]$ are located inside V_{C_0} , and the corresponding arrival curve of $[t_1, t_2]$ is located inside V_{C_i} . Furthermore, t_1 and t_2 are determined either by a Voronoi edge of C_0 or a Voronoi edge of C_i , or $t_1 = T_0$.

Lemma 5.17. *Let $[t_1, t_2] \in DC(l_{0i})$ be a minimum segment. (1) When $l_{0i}(t_1)$ is valid, there is exactly one valid buffer on $[t_1, t_2]$. (2) When $l_{0i}(t_1)$ is invalid, no valid buffer on $[t_1, t_2]$ exists.*

Proof. (1): Suppose by contradiction that there are at least two valid buffers $[t_1, t_3]$ and $[t'_1, t'_2]$ on $[t_1, t_2]$, $t'_1 > t_3$. Every $l_{0i}(\hat{t})$, $\hat{t} \in (t_3, t'_1)$, is invalid. By Corollary 5.3, a disc C_j whose departure curve intersects $DC(l_{0i})$, changes the validity of $l_{0i}(t_3)$. Let C_j be an obstacle disc on $l_{0i}(t_3 + \epsilon)$. We claim that C_j projects on $l_{0i}(t_2)$. This follows because by Lemma 5.2, if $l_{0i} = l_{0i}^{rr}$, then $l_{0i}^{rr}(t_3 + \epsilon) \parallel l_{0i}^{rr}(t_2)$. This also holds if $l_{0i} = l_{0i}^l$. Otherwise, a time $\bar{t} \in [t_1, t_2]$ exists, such that for $t < \bar{t}$, C_j projects on $l_{0i}(t_2)$, and for $t > \bar{t}$, C_j does not project on $l_{0i}(t_2)$. Then, o_0 and o_j are co-linear with the point $q_{\bar{t}} \in DC(l_{0i}^l)$ corresponding to $\bar{t}$. By Lemma 5.16, C_j is an obstacle disc on $l_{0i}(\bar{t})$, and C_j covers $q_{\bar{t}}$. Thus $q_{\bar{t}}$ is closer to C_j than to C_0 , and $q_{\bar{t}}$ is not located inside V_{C_0} , which is a contradiction. By Lemma 5.16, C_j is an obstacle disc on $l_{0i}(t)$ for any $t_3 < t \leq t_2$.

(2): Let C_j be the obstacle disc on $l_{0i}(t_1)$, where, C_j is neither a Voronoi neighbor of C_0 nor a Voronoi neighbor of C_i . A similar argument as above shows that there is no valid buffer on $[t_1, t_2]$. $\square$

For each segment $[t_1, t_2]$, Lemmas 5.16 and 5.17 show that if we know that $l_{0i}(t_1)$ is valid, and t_q is the minimum time on $[t_1, t_2]$, such that C_0 , C_j and C_i are co-tangent to $l_{0i}(t_q)$, then we will show in the remainder of this section how to obtain a valid buffer $[t_1, t_q] \in [t_1, t_2]$.

Figure 5.17: For the proof of Lemma 5.18 (2). Assume that $DC(l_{0i}^{rr})$ intersects the Voronoi edge between C_0 and C_j with p and q .

Next we estimate the number of segments $[t_1, t_2]$.

Lemma 5.18. *For the same-speed discs, (1) a $DC(l_{0i}^l)$ has $O(\deg(V_{C_0})) + O(\deg(V_{C_i}))$ minimum segments, (2) a $DC(l_{0i}^{rr})$ has at most one minimum segment, (3) there are in total $O(n^2)$ minimum segments on $DC(l_{ij})$, for all $C_i, C_j \in \mathcal{D}$.*

Proof. As definition, the minimum segments must be located inside V_{C_0} , and the corresponding arrival curve must be located inside V_{C_i} . (1) The departure curve of l_{0i}^l may intersect V_{C_0} in at most $\deg(V_{C_0})$ segments. Similarly, for the arrival curve, at most $\deg(V_{C_i})$ segments are located inside V_{C_i} . Thus, in total, $O(\deg(V_{C_0})) + O(\deg(V_{C_i}))$ segments on the departure curve are located inside V_{C_0} , and the corresponding segments on the arrival curve are located inside V_{C_i} .

(2) By Lemma 5.2, $DC(l_{0i}^{rr})$ is a straight line passing through the center of C_0 . We claim that $DC(l_{0i}^{rr})$ intersects V_{C_0} in one segment. Suppose by contradiction that $DC(l_{0i}^{rr})$ intersects V_{C_0} in two segments. Let $DC(l_{0i}^{rr})$ pass through V_{C_j} , where, C_j is a Voronoi neighbor of C_0 , and p and q be two intersection points on the Voronoi edges of C_j . Let p be the ending point of a segment, q be the starting point of another segment on $DC(l_{0i}^{rr})$, and $t_q > t_p$. Let $q' \in DC(l_{0i}^{rr})$ be another point with $t_{q'} > t_q$. Refer to Figure 5.17. By triangle inequality, $\|pC_j\| + \|pq'\| > \|q'C_j\|$, where, $\|pC_j\| = \|po_j\| - r_j$, and $\|q'C_j\| = \|q'o_j\| - r_j$. Since $\|pC_0\| = \|pC_j\|$, q' has shorter distance to C_j than to C_0 . Thus, no point on $DC(l_{0i}^{rr})$ after t_q is located inside V_{C_0} . This results a contradiction. Since the arrival curve on C_i is a straight line, it has the

same property as the departure curve, and it intersects V_{C_i} in one segment. Thus, $DC(l_{0i}^{rr})$ has at most one minimum segment.

(3) By the first statement, there are at most $\sum_{\forall C_i \in \mathcal{D}, \forall C_j \in \mathcal{D}} (deg(V_{C_i}) + deg(V_{C_j})) \cdot n = O(n^2)$ minimum segments on the right-left departure curves. By the second statement, there are at most $O(n^2)$ minimum segments on the right-right departure curves. $\square$

Based on the geometric locations of the minimum segments with respect to the Voronoi neighbors of C_0 , we divide the discs in $\mathcal{D}$ into $deg(V_{C_0})$ groups. For each minimum segment $[t_1, t_2] \in DC(l_{0i})$, $t_1 > T_0$, we find a Voronoi neighbor C_l of C_0 , such that C_l is a right-block of $l_{0i}(t_1)$, and is the lowest Voronoi neighbor of C_0 along $l_{0i}(t_1)$. Imagine that we rotate the positive x -axis around o_0 to o_0o_l , then the group of discs, say $\mathbb{D}$, includes all the discs C_j which are located in the North-East, North-West, and South-West quadrants around o_0 , where o_0o_l is the x -axis. This division guarantees that the discs which project on $l_{0i}(T_0)$ but do not project on $l_{0i}(t_1)$ are out of consideration when computing a valid buffer on $[t_1, t_2] \in DC(l_{0i})$. Also, all discs which may have an effect on the valid buffer computation on $[t_1, t_2]$ are in $\mathbb{D}$. Next, we focus on the discs in $\mathbb{D}$.

So, we have two tasks when computing a valid buffer on $[t_1, t_2] \in DC(l_{0i})$: (1) for each minimum segment $[t_1, t_2]$, compute the validity of $l_{0i}(t_1)$, (2) compute $q^*(0, j, i)$ with some C_j , $t_q \in [t_1, t_2]$, and the validity of $l_{0i}(t_1)$ changes after t_q .

We pick a disc as C_0 . Suppose that all minimum segments $[t_1, t_2] \in DC(l_{0i})$, for all $C_i \in \mathcal{D}$, are placed into a list S associated with C_0 .

Since it is not easy to efficiently detect the validity of $l_{0i}^l(t_1)$, $t_1 > T_0$, for all $C_i \in \mathbb{D}$, we construct a right-right tangent L , called *supporting tangent* for $l_{0i}^l(t_1)$, which is tangent to $C_0(T_0)$, and is parallel to $l_{0i}^l(t_1)$, with length $|l_{0i}^l(t_1)|$. Refer to Figure 5.7 in Section 5.2.2. The validity of a supporting tangent is comparatively easier to compute, as it is determined by the discs from only one side of the supporting tangent, while for a right-left tangent, the discs on both sides of it should be considered. Also, the property that a right-right tangent l at different times is parallel to each other helps to reduce the computation. Define $L(t)$ (L at time t) as: $L(t)$ is tangent to $C_0(t)$,

and is parallel to L , $|L(t)| = |L|$. We will show that it is easier to detect the validity of $L(t_1)$, because the validity of $L(t_1)$ is determined only by two crucial discs. $L(t_1)$ and $l_{0i}^l(t_1)$ have the following relationship.

Lemma 5.19. *The validity of tangent $l_{0i}^l(t_1)$ can be obtained from computing the validity of $L(t_1)$.*

Proof. The tangent L at time t_1 is exactly the tangent $l_{0i}^l(t_1)$. □

The validity of $L(T_0)$ is computed when executing Algorithm 10, where, L is a right-right tangent with length $|l_{0i}^l(t_1)|$. Suppose that L is valid at time T_0 . Then, the disc projects on L , which changes the validity of L at time t_q , is an obstacle disc on $l_{0i}^l(t_1)$, if $t_q < t_1$, according to Lemma 5.16. We call a disc C_J *supporting disc* for $l_{0i}^l(T_0)$, such that $DC(l_{0i}^l) \cap DC(l_{0J}^l) = q^*(0, J, i)$, and t_q is the minimum time among all $DC(l_{0i}^l) \cap DC(l_{0j}^l) \neq \emptyset$, $C_i, C_j \in \mathbb{D}$.

Lemma 5.20. *Considering the discs in $\mathbb{D}$ for $l_{0i}^l(t_1)$. Let the supporting tangent L of $l_{0i}^l(t_1)$ be valid at time T_0 . Let L and l_{0R}^l be co-linear at time t_q , where, C_R is the nearest-block for $(L, l_{0i}^l(t_1))$ (as defined in Section 5.2.2). Let $l_{0i}^l(T_0)$ and $l_{0J}^l(T_0)$ be co-linear at time t_w , where, C_J is the supporting disc of $l_{0i}^l(T_0)$. Tangent $l_{0i}^l(t_1)$ is valid if and only if $t_1 \leq \min\{t_q, t_w\}$.*

Proof. According to the definition of nearest-block, l_{0R}^l is right-block of L , C_R projects on L , C_R is a left-block of $l_{0i}^l(T_0)$, or an obstacle on $l_{0i}^l(T_0)$. $\Leftarrow$: By Lemma 5.15, the validity of $l_{0i}^l(t_1)$ is determined by a left-block disc and a right-block disc. By Lemma 5.16, C_R is an obstacle disc on L after time t_q . Also by the definition of $\mathbb{D}$, C_J is higher than C_0 along $l_{0i}^l(T_0)$, thus, C_J is an obstacle disc on l_{0i}^l after time t_w . Therefore, if $l_{0i}^l(t_1)$ is valid, then $t_1 \leq \min\{t_q, t_w\}$.

$\Rightarrow$: The right-blocks of L can be divided into three groups. Group A includes all the left-blocks of $l_{0i}^l(T_0)$, group B includes all the discs intersecting $l_{0i}^l(T_0)$, and group C includes all the right-blocks of $l_{0i}^l(T_0)$. We first claim that if $C_R \notin A$ and $C_R \notin B$, then $A = \emptyset$ and $B = \emptyset$, according to the definition of C_R for $(L, l_{0i}^l(t_1))$.

Thus, the discs in C change the validity of $L(t_1)$, therefore, in this case, the statement, if $t_1 \leq \min\{t_q, t_w\}$ then $l_{0i}^r(t_1)$ is valid, is true. Second we claim that if $C_R \in A$ or $C_R \in B$, then $t_q < t_1$. This is because that $\text{slope}(l_{Ri}^l) < \text{slope}(L) = \text{slope}(l_{0i}^r(t_1))$, and C_R is an obstacle when the robot arrives at C_i along $L(t_1)$. Since in this case $t_q < t_1$, it has no effect on the statement. Overall, the statement is true. $\square$

By Lemma 5.20, we know that either C_R or C_J determines the validity of $l_{0i}^r(t_1)$. How to compute C_R ? When the supporting tangent L of $l_{0i}^r(t_1)$ is valid, according to Lemma 5.7, we can obtain C_R for L . C_J , which determines $q^*(0, J, i)$, for l_{0i}^r can be obtained from the arrangement of the right-left departure curves around C_0 .

So far we have shown the discs determining the validity of $l_{0i}^r(t_1)$. Suppose that $l_{0i}^r(t_1)$ is valid, then we have to compute the time $t_q \in [t_1, t_2]$ when the validity of $l_{0i}^r(t_1)$ changes. This is the second task, which is carried out by computing the arrangement of the minimum segments in S associated with C_0 . Suppose that at time t_q , C_0 , C_j and C_i are co-tangent to $l_{0i}(t_q)$, $t_q \in [t_1, t_2]$. Then, according to Lemma 5.17, we can ignore the intersection points on the portion $[t_q, t_2]$, even if $l_{0i}(t_1)$ is invalid. This can be controlled during the execution of the Bentley-Ottmann sweepline algorithm for the intersection points. At each intersection point q , we test if a C_j exists which will become an obstacle after t_q . In that case, we will delete the remaining portion $[t_q, t_2]$ in the arrangement. If C_0 , C_i and C_j are co-tangent to $l_{0j}(t_q)$, then we can delete the part $t > t_q$ on the minimum segment in $DC(l_{0j})$, as $l_{0j}(t_q + \epsilon)$ becomes invalid because of disc C_i . Finally, we record the intersection points on each $[t_1, t_2]$ together with their corresponding time. The arrangement of the right-left departure curves can be computed in the same way in order to obtain C_J efficiently. But in this computation, we use the disc set $\mathbb{D}$ associated with $[t_1, t_2]$ instead of $\mathcal{D}$, as $\mathbb{D}$ eliminates the disc $C_J \in \mathcal{D}$ for $l_{0i}^r(T_0)$, but this $C_J \in \mathcal{D}$ does not project on $l_{0i}^r(t_1)$.

Summarizing, Algorithm 14 computes all valid buffers for same-speed discs.

Note, the buffer computation on $DC(l_{0i}) = DC(l_{0i}^{rr})$ is considerably easier, since there is at most one valid buffer on $DC(l_{0i}^{rr})$. By Lemma 5.18, follows that if $C_i \in \mathcal{D}$, and $l_{0i}^{rr}(T_0)$ is invalid, no valid buffer exists on $DC(l_{0i}^{rr})$. But for consistency of the

algorithm, we process them in the same way as the right-left departure curves.

Lemma 5.21. *The arrangement of the partial departure curves of the same-speed discs in $\mathcal{D}$ can be computed in $O(n^2 \log n)$ time. The total number of the intersection points in the arrangement is $O(n^2)$.*

Proof. Each disc in $\mathcal{D}$ takes the place of C_0 exactly once. The arrangement of the partial departure curves around C_0 can be computed in $O(n \log n + k)$ time by applying the Bentley-Ottmann sweepline paradigm. Since at each intersection point during the execution of the sweepline algorithm, one departure curve is deleted, in total $k = O(n)$ intersection points are obtained. The arrangement of the partial departure curves is computed twice. Once is applied to the discs in $\mathcal{D}$, and once is applied to each group of discs $\mathbb{D}$: the discs in $\mathcal{D}$ are divided into $\deg(V_{C_0})$ groups. Thus, there are $\sum_{C_i \in \mathcal{D}} (\deg(V_{C_i})) \cdot n = O(n^2)$ intersection points in total. The computation takes $\sum_{C_i \in \mathcal{D}} (\deg(V_{C_i})) \cdot n \log n = O(n^2 \log n)$ time. $\square$

Theorem 5.3. *By applying Algorithm 14, the valid buffers on all the departure curves of the same-speed discs in $\mathcal{D}$ can be computed in $O(n^2 \log n)$ time.*

Proof. First we establish the correctness. Each disc in $\mathcal{D}$ takes the place of C_0 exactly once. Step 4 computes the validity changing point q on each segment $[t_1, t_2]$ by computing the arrangement of the segments. Step 5 divides the segments into several groups. Each group has a corresponding set of discs in $\mathcal{D}$. The related discs around C_0 are deleted from $\mathcal{D}$, if they project on $l_{0i}(\bar{t})$, $\bar{t} < t_1$, but do not project on $l_{0i}(t_1)$. We deal with each group of discs separately. Step 7 computes the arrangement of the right-left departure curves for the discs in a group $\mathbb{D}$ in order to obtain C_J . For each minimum segment $[t_1, t_2]$ associated with $\mathbb{D}$, we take $[T_0, t_2]$ as the departure curve, and we take the parts of all other departure curves that are located inside V_{C_0} . This is because that for l_{0i}^l , $C_i \in \mathbb{D}$, $q^*(0, J, i)$ may happen at $t_q < t_1$, $C_J \in \mathbb{D}$. C_J is captured in Step 13. Steps 8-15 compute the validity of $l_{0i}^l(t_1)$, according to Lemma 5.19. According to Lemma 5.17, once $l_{0i}^l(t_1)$ is valid, a valid buffer exists on $[t_1, t_2]$. This is done in Steps 17-21.

Next is the time complexity analysis. The arrangement of the departure curves can be computed in $O(n^2 \log n)$ time by Lemma 5.21. There are $O(n^2)$ intersection points in total. By Lemma 5.18, there are $O(n^2)$ minimum segments, and by Lemma 5.17, there are $O(n^2)$ valid buffers. By Theorem 5.1, all valid tangents are detected in $O(n^2 \log n)$ time. By Lemma 5.7, C_R for all valid L s are computed in $O(n^2 \log n)$ time as there are $O(n^2)$ minimum segments. Steps 13 and 19 take $O(1)$ time. Thus, the $O(n^2 \log n)$ time complexity of Algorithm 14 is derived. $\square$

5.7 Detecting the validity of a spiral

As in [115], by computing the arrangement of the departure curves, and then building a trapezoidal map, we find the next departure curve that a spiral will meet on the same disc efficiently. However, when discs grow at the same speed, we compute the arrangement of the valid buffers on the departure curves instead. This is because a node (departure curve) is useful only when a valid tangent exists on this departure curve. For a spiral, it can continue around a disc after arriving at a point on a departure curve on the disc. When running Dijkstra's shortest path algorithm, the arriving point $p(t_1, \theta_0)$ on a departure curve on a disc, say C , is known. Point p is represented by (t_1, θ_0) in polar coordinates on C (suppose that C is a point at time T_0). The spiral function is given [115], where, the speed of disc C is 1, and the speed of the robot is V_R :

$$\theta(t) = \sqrt{V_R^2 - 1} \log t + \theta_0.$$

By performing point location, p is found in a cell of the trapezoidal map built upon the arrangement of the departure curves around C_0 , and the next departure curve of the spiral passing through p is known in the cell. The Euclidean coordinates of the spiral passing through point $p(t_1, \theta_0)$ is given by:

$$\begin{cases} x(t) = t \cos(\sqrt{V_R^2 - 1}) \log t - \sqrt{V_R^2 - 1} \log t_1 + \theta_0 \\ y(t) = t \sin(\sqrt{V_R^2 - 1}) \log t - \sqrt{V_R^2 - 1} \log t_1 + \theta_0 \end{cases}$$

Let the arrival time of the spiral to the next departure curve be t_2 . The functions of the Voronoi edges of C_0 also have x and y variables in Cartesian coordinates. If the spiral does not intersect the corresponding Voronoi edge of C_0 in $[t_1, t_2]$, then the spiral is valid. Otherwise, this spiral is invalid. Note, the above computation is applicable to any spiral from the arriving point $p(t_1, \theta_0)$ to the next departure curve on a disc, no matter p is on a departure curve or not. This step is the same for variant-speed discs as for same-speed discs.

5.8 Executing Dijkstra's shortest path

We now apply a variant of Dijkstra's shortest path algorithm on the graph G constructed in Algorithm 9. The nodes in the graph are the departure curves around each disc. For the same-speed case, the nodes are the valid departure curves. There are two kinds of edges: one is spirals around discs. Each spiral is the portion between two neighboring departure curves on a disc. The other edge is a tangent portion possibly followed by a spiral starting from the ending point of the tangent on a disc to the next departure curve on the same disc. The tangent portion e of an edge exists iff e has a life span.

When executing Dijkstra's shortest path algorithm, at some time t , for the tangent portion e of an edge, only t is in e 's life span that Dijkstra's algorithm can consider. Section 5.7 shows how the validity of a spiral is computed. If Dijkstra's algorithm encounters an edge with the life span of its tangent portion expired, we delete it from the heap. If a spiral is invalid, then we delete the corresponding edge from the heap. Also, for each node, we always keep the smallest time associated with the node.

We now summarize this chapter with the main Theorem:

Theorem 5.4. *The C-SPAGD problem can be solved in $O(n^2 \log n)$ time, and the*

V-SPAGD problem can be solved in $O(n^3 \log n)$ time, where, n is the number of discs.

Proof. By Theorem 5.3, with the same-speed discs in $\mathcal{D}$, there are $O(n^2)$ valid buffers on the departure curves, and the valid buffers can be computed in $O(n^2 \log n)$ time. The graph has $O(n^2)$ nodes corresponding to departure curves (valid buffers), and $O(n^2)$ edges corresponding to the spirals and relevant valid tangents. Thus, using Dijkstra's algorithm on the edges, i.e., tangents that have expiry time and spirals, as illustrated above, a shortest path can be determined in $O(n^2 \log n)$ time. By Theorem 5.2, with the various-speed discs, the valid buffers can be computed in $O(n^3 \log n)$ time. The graph has $O(n^3)$ nodes and edges. Thus, a shortest path can be found in $O(n^3 \log n)$ time. $\square$

5.9 Conclusions

The result of this chapter is an $O(n^2 \log n)$ time algorithm to compute a collision-free shortest path between two points among a set of n discs growing at the same speed, thus improved the original result by van den Berg and Overmars, and matches the known bound for the more restricted case when all discs are static. When the discs grow at different speeds, our algorithm runs in $O(n^3 \log n)$ time. Overall, we predict a possible future path for the robot between every two discs at the time when the robot starts. One benefit of our algorithms is, once we have the buffer (possible future path), then we can plan a shortest path for the robot starting at any time from any location s to any location g among a set of discs, without re-computing the possible future paths due to the change of the starting/ending location and starting time. It is interesting to see how to modify our algorithms to predict when there will not be a shortest path to reach g for the robot anymore. Such that it may help to solve the decision problem efficiently: does there exist a path from s to g . For example, a group of discs may connect to each other at time $\bar{t}$, and the robot happens to be located inside the connected region. Thus, if the connected region can be detected in advance together with the time that it will happen, then we can terminate the

computation. Another problem coming along is if there is a path from s to g , then efficiently compute the path by an approximation solution. A more realistic problem is that the obstacles are truly dynamic. The discs emerge and disappear in the plane at different times, or even shrink and expand dynamically. Then in this discs set, we can reconsider the problems on finding a path (shortest) from s to g .

Algorithm 8 Computing C_R for $(L, l_{0i}^l(t_1))$

Input: A list $\mathcal{L}$ of valid right-right and right-left tangents originating at C_0 for all discs in $\mathcal{D}$, stored in decreasing order of slope. A list $\mathbb{L}$ of pairs $(L, l_{0i}^l(t_1))$, stored in decreasing order of $\text{slope}(l_{0i}^l(t_1))$

Output: nearest-block C_R for each $(L, l_{0i}^l(t_1)) \in \mathbb{L}$

```

1: Let  $l_{01}^{rr} \in \mathcal{L}$  have the smallest length among all the valid right-right tangents.
   Push  $l_{01}^{rr}$  onto a stack  $S$ . Start from  $l_{01}^{rr}$ , in decreasing order of slope of the tangents
   in  $\mathcal{L}$ 
2: for each tangent  $l \in \mathcal{L}$  in order, let  $l$  be tangent to  $C_i$  do
3:   while  $S \neq \emptyset$  do
4:     Compute the projection point  $p$  of  $o_i$  onto the top element  $l_{0m}^{rr} \in S$ 
5:     if  $p \in l_{0m}^{rr}$  then
6:       pop off  $l_{0m}^{rr}$  from  $S$ 
7:     else if  $C_i$  is higher than  $C_m$  along  $l_{0m}^{rr}$  /*Eliminate the case where  $C_i$  is
       lower than  $C_0$  along  $l_{0m}^{rr}$  */ then
8:       if  $l = l_{0i}^{rr}$  then
9:         push  $l_{0i}^{rr}$  onto  $S$ 
10:      else if  $(L, l(t_1)) \in \mathbb{L}$  /* $l = l_{0i}^l$ */ then
11:        compute  $C_R$  for  $(L, l(t_1))$ 
12:      end if
13:    exit while loop
14:  else
15:    exit while loop
16:  end if
17: end while
18: end for

```

Algorithm 9 Computing a shortest path for a robot traveling among growing discs

Input: Disc set $\mathcal{D} \in \mathbb{R}^2$. Each $C_i \in \mathcal{D}$ has a speed V_i . The speed of the robot is V_R .
The discs are various-speed or same-speed. Start location s and goal location g .

Output: a shortest path from s to g

- 1: compute departure curves on each disc $C_i \in \mathcal{D}$ (see Section 5.2.1). The departure curves are the nodes of graph G (implicitly)
 - 2: compute valid buffers on the departure curves (see Section 5.5 and Section 5.6).
The valid buffers define the tangent portions of the edges in graph G
 - 3: /*run Dijkstra's shortest path algorithm on the implicit graph G on-the-fly
 $E_1 := \{ \text{spirals between two neighbouring departure curves} \}$
 $E_2 := \{ \text{tangents} + \text{spirals} \}$ */
 An edge of G is either in E_1 , a spiral between two neighboring departure curves on a disc, or in E_2 , a tangent determined by a valid buffer, possibly followed by a spiral from the ending point of the tangent on a disc to the next neighboring departure curve on the same disc. The edges are revealed at run time, processed by the following two steps
 - 4: during running Dijkstra, according to the arriving point p on disc C_i , check the validity of the spiral (CW or CCW, either in E_1 or E_2) from p to the next departure curve on C_i (see Section 5.7)
 - 5: **if** during running Dijkstra, the arriving point p is on a valid departure curve
then
 - 6: p is the starting point of a tangent portion in E_2
 - 7: **end if**
 - 8: find time-minimal shortest path from s to g in G .
-

Algorithm 10 Computing valid tangents around C_0

Input: A set of right-right and right-left tangents originating at C_0 in a quadrant (w.l.o.g assume that it is North-West quadrant). Radius of disc C_0 is 0 ($r_0 = 0$).

Output: A set L' of valid tangents.

- 1: Initialize a min-heap Q , which stores all tangents with length as key.
 - 2: Construct a balanced binary search tree $\mathbb{T}$, which stores all the tangents in increasing order of slope.
 - 3: **while** $Q \neq \emptyset$ **do**
 - 4: Let C_i be the corresponding disc having the tangent with the minimum length in Q
 - 5: locate the two tangents l_{0i}^{rr} and l_{0i}^{rl} in $\mathbb{T}$. Let $\mathbb{A}$ be the set of tangents in $\mathbb{T}$ whose slope is between $\text{slope}(l_{0i}^{rr})$ and $\text{slope}(l_{0i}^{rl})$.
 - 6: delete all tangents in $\mathbb{A}$.
 - 7: **if** l_{0i}^{rr} (resp. l_{0i}^{rl}) is not deleted before, and the tangent point on C_i is located in the Voronoi cell of C_i **then**
 - 8: $L' := L' \cup \{l_{0i}^{rr}\}$ (resp. $L' := L' \cup \{l_{0i}^{rl}\}$)
 - 9: **end if**
 - 10: delete l_{0i}^{rr} and l_{0i}^{rl} from the tree.
 - 11: **end while**
-

Algorithm 11 Computing valid buffers

Input: A set of discs in $\mathcal{D}$. All right-right and right-left departure curves around C_0 : $\forall C_i \in \mathcal{D}$, $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$

Output: Valid buffers on each $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$

- 1: compute the arrangement of the departure curves $DC(l_{0i})$ around C_0 , $\forall C_i \in \mathcal{D}$, where, $DC(l_{0i})$ represents $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$
 - 2: associate a list S_{0i} with $DC(l_{0i})$, where, S_{0i} represents S_{0i}^{rr} and S_{0i}^{rl} accordingly, and S_{0i} holds all the time sorted intersection points (with corresponding time) in the arrangement on $DC(l_{0i})$. The point corresponding to time T_0 on $DC(l_{0i})$ is also stored in S_{0i} .
 - 3: **for** every two consecutive intersection points $p, q \in S_{0i}$ in order **do**
 - 4: compute valid buffers $[t'_p, t'_q]$ on $DC(l_{0i})$, with $[t'_p, t'_q] \subseteq [t_p, t_q]$.
 - 5: **end for**
-

Algorithm 12 Computing valid buffers with various-speed discs

Input: The arrangement of $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$, $\forall C_i \in \mathcal{D}$. The set S_{0i}^{rr} (S_{0i}^{rl}) associated with $DC(l_{0i}^{rr})$ (resp. $DC(l_{0i}^{rl})$) stores the time-sorted intersection points on $DC(l_{0i}^{rr})$ (resp. $DC(l_{0i}^{rl})$)

Output: Valid buffers on each $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$

```

1: compute the obstacle discs in  $\mathcal{D}$  for  $l_{0i}^{rr}(T_0)$  (resp.  $l_{0i}^{rl}(T_0)$ ), put the obstacle discs
   into  $\mathbb{M}$ . Let  $t_p = T_0$ 
2: for each  $DC(l_{0i})$  do
3:   let  $Q' =$  each event point on  $DC(l_{0i})$  determined by Voronoi edge of  $C_0$ 
     and of  $C_i$ ,  $Q = S_{0i} \cup Q'$ 
4:   while  $Q \neq \emptyset$ , extract the point  $q \in Q$  with minimum time do
5:     if  $l_{0i}^{rr}(t_p)$  (resp.  $l_{0i}^{rl}(t_p)$ ) is valid, i.e.,  $|\mathbb{M}| = 0$  then
6:        $[t_p, t_q]$  is a valid buffer on  $DC(l_{0i}^{rr})$  (resp.  $DC(l_{0i}^{rl})$ )
7:       if  $q$  is an event point defined in Corollary 5.3, let  $C_j$  be the disc which
         determines  $q$  /*except the event points  $q : (0, i, j)$  in  $S_{0i}$ */ then
8:          $\mathbb{M} = \mathbb{M} \cup \{C_j\}$ . Set  $t_p = t_q$ , and set  $l_{0i}^{rr}(t_p)$  (resp.  $l_{0i}^{rl}(t_p)$ ) to be invalid
9:       end if
10:      else if  $q$  is an event point defined in Corollary 5.3, let  $C_j$  be the disc which
        determines  $q$  /* $l_{0i}(t_p)$  is invalid, i.e.,  $|\mathbb{M}| \neq 0$ */ then
11:        if  $C_j \in \mathbb{M}$  then
12:           $\mathbb{M} = \mathbb{M} - \{C_j\}$ , set  $t_p = t_q$ 
13:        else
14:           $\mathbb{M} = \mathbb{M} \cup \{C_j\}$ , set  $t_p = t_q$ 
15:        end if
16:      end if
17:    end while
18:  end for

```

Algorithm 13 multiplicatively additively weighted Voronoi diagram

Input: The set of cones corresponding to the set of discs in $\mathcal{D}$. (For each disc $C_i \in \mathcal{D}$, $cone_i$ is constructed with apex located at the center of C_i , and open angle $2 \arctan V_i$. At time $t = 0$, the cones are cut to the discs in $\mathcal{D}$.)

Output: The Voronoi diagram of the set of discs

```

1: for each  $cone_i$ ,  $1 \leq i \leq n$  do
2:   for each  $cone_j$ ,  $j \neq i$ ,  $j \in [1, n]$  do
3:     compute  $proj_{ij} = proj(cone_i \cap cone_j)$ 
4:   end for
5:   compute  $V_i$  as the intersections of all regions  $proj_{ij}$  /* $V_i$  is the Voronoi cell of
   the corresponding disc  $C_i$ */
6: end for

```

Algorithm 14 Computing valid buffers of the same-speed discs

Input: A set of same-speed discs in $\mathcal{D}$. All right-right and right-left departure curves around C_0 : for all $C_i \in \mathcal{D}$, $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$

Output: Valid buffers on each $DC(l_{0i}^{rr})$ and $DC(l_{0i}^{rl})$

```

1: for each disc in  $\mathcal{D}$ , let the disc be  $C_0$  do
2:   compute the validity of the tangents around  $C_0$  at time  $T_0$ 
3:   compute the minimum segments on all departure curves around  $C_0$ . Put the
     minimum segments in  $S$ .
4:   efficiently compute the arrangement of the minimum segments /*obtaining the
     point which determines the maximum time value on a valid buffer*/
5:   divide the discs in  $\mathcal{D}$  around  $C_0$  into  $\deg(V_{C_0})$  groups, each minimum segment
      $[t_1, t_2]$  in  $S$  has a group of discs associated with,  $t_1 > T_0$ 
6:   for each group of discs, say  $\mathbb{D}$  do
7:     efficiently compute the arrangement of the right-left departure curves
8:     for each minimum segment  $[t_1, t_2] \in S$  associated with  $\mathbb{D}$  do
9:       compute the validity of  $L$ ,  $L \parallel l_{0i}^{rl}(t_1)$ ,  $|L| = |l_{0i}^{rl}(t_1)|$ ,  $C_i \in \mathbb{D}$ , by applying
         Algorithm 10 /*building  $L$  for  $l_{0i}^{rl}(t_1)$ */
10:    end for
11:    compute  $C_R$  for all valid  $L$ , by applying Algorithm 8 /*obtaining the disc
         $C_R$ */
12:    for each valid  $L$  do
13:      find the intersection point  $q^*(0, J, i)$  on  $[T_0, t_2] \in DC(l_{0i}^{rl})$  computed in
        Step 7 /*obtaining the disc  $C_J$ */
14:      compute the validity of  $l_{0i}^{rl}(t_1)$  according to  $C_R$  and  $C_J$  by Lemma 5.19
15:    end for
16:  end for
17:  for each  $C_j \in \mathcal{D}$  do
18:    if  $l_{0j}(t_1)$  is valid,  $t_1 \geq T_0$  then
19:      compute the valid buffer on  $[t_1, t_2] \in DC(l_{0j})$ 
20:    end if
21:  end for
22: end for

```

Chapter 6

Summary and Future Work

This thesis focuses on building the architecture and framework of the ubiquitous IMA system, and designing efficient algorithms related to two services in the IMA system. The work in the IMA system is a fundamental step towards building a realistic user-centric mobile system which provides services related to geometric data. The work in the algorithms design would yield a host of related interesting questions likely to stimulate other researchers.

The IMA framework, in which mobile users can get access to a wide and ever extending range of services, is implemented in a distributed environment. Each user is represented by some personal agents that can communicate between themselves, allowing them to help users who need to cooperate for a purpose or another. By experiment, we have shown that the architecture is able to support a wide range of services to connected mobile users, including services that facilitate collaboration between users.

The IMA architecture is geared towards mobile users and addresses a growing need for mobile infrastructure. It differentiates itself from other existing systems in two aspects: collaboration (co-working), and customization and personalization. The collaboration and co-working, is aimed at providing an architecture where users can interact with each other in order to accomplish a task. The most important aspect of the architecture (related to this point) is that it enables users to be mobile and able to participate in the task irrespective of their location. This aspect has been solved here.

So, the next step for the IMA system focuses on customization and personalization, which is aimed at different mobile users who require spatial data to accomplish their tasks. The main idea is not to develop a map or a spatial visualization according to some specification, but rather to determine which specification should be used and when, in order to obtain the appropriate output that is required by the user. However, these users do not need to continuously interact with the system and the application. An application should have knowledge about the user needs, habits, abilities and limitations. This information which is private to the user is accessed and used by the user's agent (the intelligent map agent) to accomplish the task with little or no interaction with the user. We continue to research and develop these core areas.

To produce a service inside the IMA system requires: (1) a theoretical solution to the problem, (2) an implementation of the solution, (3) conversion of the implementation into an integrated IMA service. If a theoretical solution to a problem underlying the service is not yet available, research must be carried out to find one. In our theoretical work, we just provide this. In particular, for the spanning growing disc problem, we have provided algorithms improving currently known bounds. Implementation of these algorithms are beyond the scope of this thesis, as they pose numerous challenges resulting from the fact that no multiplicatively additively weighted Voronoi diagram is implemented and that the computational geometry implementations would require the numerically stable handling of non-linear geometric primitives (such as collinearity detection) and computations involved. The previous algorithms [115], were only implemented as a heuristics. Similarly, the meeting scheduler implementation would require an implementation of the additively weighted furthest Voronoi diagram, which is not available among others. We do however provide an implementation of a particular meeting scheduler as described in [43] and have successfully provided it as a service.

From the implementation view, for the future work, we can consider improving the complexity of the multiplicatively additively weighted Voronoi diagram in Section 5.5.2.2, and then implementing a numerical stable algorithm for the growing disc

problem.

In our theoretical research, the meeting scheduler problem is finding a suitable place and time for a group of people to meet, given each of the participants space and time constraints. We offered several techniques to tackle the problem which include an approach in solving the proposed problem by viewing it entirely as a geometrical problem, or linear programming or graph problems.

The following is a list of future research problems on the meeting scheduler

1. Each participant has a list of meeting schedules

Suppose that each participant has k meetings scheduled. By plane sweep, the longest new meeting can be computed in $O(kn^2 \log n)$ time as follows: first compute all valid common free time intervals, i.e, no meeting happens at any time in those time intervals and they are maximal. Second, for each valid time interval, apply the algorithm in Chapter 4 to find the longest meeting location and meeting duration. Third, from all new meeting durations, select the new meeting location with the longest meeting duration. Since there are $O(kn)$ events, i.e., the locations of the sweep planes, the total $O(kn^2 \log n)$ time complexity is obtained. Can one reduce this time complexity?

2. Suppose that we obtain a longest meeting that lasts L hours. A new meeting lasting L'' hours is however needed, with the number of participants as large as possible, $L'' > L$.

The movement of a participant can be represented by a cone-like model, which is composed of three parts joined together: each of the upper and lower part of the model is a cone whose base is an ellipse, and the middle part is a cylinder. The base of the cylinder is the same ellipse as the base of the cones. The ellipse is obtained as follows: the previous and the subsequent meeting locations, l^{pre} and l^{sub} , are the two focuses of the ellipse. Let the longest time meeting location which lasts L hours be p . The possible meeting locations lasting L hours must satisfy: $\|x - l^{pre}\| + \|x - l^{sub}\| = \|pl^{pre}\| + \|pl^{sub}\|$, which is the equation of

an ellipse. For a new meeting lasting L'' hours, the meeting location x should satisfy: $\|x - l^{pre}\| + \|x - l^{sub}\| \leq \|pl^{pre}\| + \|pl^{sub}\| - (L'' - L)$, which means each participant should travel less than scheduling an L hours meeting, as $L'' > L$.

The problem can be solved by finding a meeting location with maximal number of participants in the intersection area of n cone-like models that representing n participants. The question is how to efficiently compute the new meeting location with duration L'' .

3. Find a new meeting location which minimizes the average travel distance for the participants to travel.

This problem is similar to the typical Fermat-Weber problem [25]. Currently, no exact solution to the Fermat-Weber problem is known as stated in [23].

The shortest path problem we tackled focuses on finding a shortest path for a point robot among discs growing at the various speeds or at the same speed. We make a significant improvement over the original results, where, the growing disks are modeled as cones in 3-space and the shortest path travels along a combination of tangents between cones and spiral curves along each cone. Our main contribution for this problem is to show that when the discs grow at the same speed, it is possible to produce a graph having $O(n^2)$ edges, thus yields an $O(n^2 \log n)$ time complexity. When the discs grow at various speed, we are able to show that the graph has $O(n^3)$ edges.

In the future, we may consider a case like this: obstacles have different shapes, grow at a constant/different speeds, and also a steady/varying speed on the path is given. We may also consider computing a shortest path among the obstacles under different metrics.

Appendix A

Computing the function of a departure curve

A.1 Compute time range for a departure curve

A.1.1 The time range for a right-right departure curve

Refer to Figure A.1. This section computes a time range for a right-right departure curve when discs grow at different speeds.

$C_1 : (o_1, r_1, 1)$

$C_2 : (o_2, r_2, V_2), V_2 \neq 1$

T : travel time of C_1

t : travel time of the robot, with speed V

$a = |o_1 o_2|$

$a^2 = (r_2 + V_2(T + t) - r_1 - T)^2 + V^2 t^2$

$(V_2^2 + V^2)t^2 + 2[(V_2 - 1)T + (r_2 - r_1)]V_2 t + [(V_2 - 1)T + (r_2 - r_1)]^2 - a^2 = 0$

$t = \frac{-V_2[(V_2 - 1)T + (r_2 - r_1)] \pm \sqrt{-V^2[(V_2 - 1)T + (r_2 - r_1)]^2 + a^2 V_2^2 + a^2 V^2}}{V_2^2 + V^2}$

When the right-right departure curve exists, $t \geq 0$. We have:

$-V_2[(V_2 - 1)T + (r_2 - r_1)] \pm \sqrt{-V^2[(V_2 - 1)T + (r_2 - r_1)]^2 + a^2 V_2^2 + a^2 V^2} \geq 0$
which is:

$$\pm \sqrt{-V^2[(V_2 - 1)T + (r_2 - r_1)]^2 + a^2 V_2^2 + a^2 V^2} \geq V_2[(V_2 - 1)T + (r_2 - r_1)] \quad (\text{A.1})$$

In order to satisfy inequality A.1,

$$-V^2[(V_2 - 1)T + (r_2 - r_1)]^2 + a^2 V_2^2 + a^2 V^2 \geq 0$$

Figure A.1: (a) for the right-right function computation. (b) for the right-left function computation.

$$[(V_2 - 1)T + (r_2 - r_1)]^2 \leq \frac{a^2(V_2^2 + V^2)}{V^2}$$

$$-\frac{a}{V}\sqrt{V^2 + V_2^2} \leq (V_2 - 1)T + (r_2 - r_1) \leq \frac{a}{V}\sqrt{V^2 + V_2^2}$$

There are two cases, one is $V_2 > 1$, the other one is $V_2 < 1$.

$$\begin{cases} V_2 > 1 \\ -\frac{a\sqrt{V^2 + V_2^2}}{V(V_2 - 1)} - \frac{r_2 - r_1}{V_2 - 1} \leq T \leq \frac{a\sqrt{V^2 + V_2^2} - V(r_2 - r_1)}{V(V_2 - 1)} \end{cases} \quad (\text{A.2})$$

and

$$\begin{cases} V_2 < 1 \\ \frac{a\sqrt{V^2 + V_2^2} - V(r_2 - r_1)}{V(V_2 - 1)} \leq T \leq \frac{-a\sqrt{V^2 + V_2^2} - V(r_2 - r_1)}{V(V_2 - 1)} \end{cases} \quad (\text{A.3})$$

Now, we go back to inequality A.1. There are two cases: “+” and “-”.

$$+ : \sqrt{-V^2[(V_2 - 1)T + (r_2 - r_1)]^2 + a^2V_2^2 + a^2V^2} \geq V_2[(V_2 - 1)T + (r_2 - r_1)]$$

$$-V^2(V_2 - 1)^2T^2 - 2V^2(V_2 - 1)(r_2 - r_1)T - V^2(r_2 - r_1)^2 + a^2V_2^2 + a^2V^2 \geq V_2^2(V_2 - 1)^2T^2 + 2V_2^2(V_2 - 1)(r_2 - r_1)T + V_2^2(r_2 - r_1)^2$$

$$(V_2 - 1)^2T^2 + 2(V_2 - 1)(r_2 - r_1)T(r_2 - r_1)^2 - a^2 \leq 0$$

$$[(V_2 - 1)T + (r_2 - r_1)]^2 - a^2 \leq 0$$

Solving this inequality, we first obtain the following:

$$\begin{cases} V_2 - 1 > 0 \\ T \geq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \leq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.4})$$

Because $(r_2 - r_1) + a > (r_2 - r_1) - a$.

So, $-\frac{V(r_2 - r_1) + aV}{V(V_2 - 1)} \leq T \leq -\frac{V(r_2 - r_1) - aV}{V(V_2 - 1)}$

Combining this with inequality A.2, we have:

$$-\frac{(r_2 - r_1) + a}{V_2 - 1} \leq T \leq -\frac{(r_2 - r_1) - a}{(V_2 - 1)} \quad (\text{A.5})$$

From inequality A.1 with prefix “+”, we obtain the second inequalities:

$$\begin{cases} V_2 - 1 > 0 \\ T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \geq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.6})$$

There is no solution in this case. We also obtain the third inequalities from A.1 with prefix “+”:

$$\begin{cases} V_2 - 1 < 0 \\ T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \geq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.7})$$

$$-\frac{(r_2 - r_1) - a}{V_2 - 1} \leq T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1}$$

Combining this inequality A.3, we have:

$$-\frac{(r_2 - r_1) - a}{V_2 - 1} \leq T \leq -\frac{(r_2 - r_1) + a}{(V_2 - 1)} \quad (\text{A.8})$$

We obtain the fourth inequalities from A.1 with prefix “+”:

$$\begin{cases} V_2 - 1 < 0 \\ T \geq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \leq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.9})$$

There is no solution in this case.

When inequality A.1 has prefix “-”: $-\sqrt{-V^2[(V_2 - 1)T + (r_2 - r_1)]^2 + a^2V_2^2 + a^2V^2} \geq V_2[(V_2 - 1)T + (r_2 - r_1)]$

$$\begin{aligned}
 & -V^2(V_2 - 1)^2T^2 - 2V^2(V_2 - 1)(r_2 - r_1)T - V^2(r_2 - r_1)^2 + a^2V_2^2 + a^2V^2 \leq V_2^2(V_2 - 1)^2T^2 + 2V_2^2(V_2 - 1)(r_2 - r_1)T + V_2^2(r_2 - r_1)^2 \\
 & [(V_2 - 1)T + (r_2 - r_1)]^2 - a^2 \geq 0
 \end{aligned}$$

Similar to inequality A.1 with prefix “+”, we obtain the following fourlists of inequalities, according to $V_2 - 1 > 0$ or $V_2 - 1 < 0$:

$$\begin{cases} V_2 - 1 > 0 \\ T \geq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \geq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.10})$$

$$T \geq -\frac{(r_2 - r_1) - a}{V_2 - 1}$$

Combining this with inequality A.2, we have:

$$-\frac{(r_2 - r_1) - a}{V_2 - 1} \leq T \leq \frac{a\sqrt{V^2 + V_2^2} - (r_2 - r_1)V}{V(V_2 - 1)} \quad (\text{A.11})$$

$$\begin{cases} V_2 - 1 > 0 \\ T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \leq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.12})$$

$$T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1}$$

Combining this with inequality A.2, we have:

$$-\frac{a\sqrt{V^2 + V_2^2} + (r_2 - r_1)V}{V(V_2 - 1)} \leq T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1} \quad (\text{A.13})$$

$$\begin{cases} V_2 - 1 < 0 \\ T \leq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \leq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.14})$$

$$T \leq -\frac{(r_2 - r_1) - a}{V_2 - 1}$$

Combining this with inequality A.3, we have:

$$-\frac{(r_2 - r_1) + a}{V_2 - 1} \leq T \leq -\frac{a\sqrt{V^2 + V_2^2} + (r_2 - r_1)V}{V(V_2 - 1)} \quad (\text{A.15})$$

$$\begin{cases} V_2 - 1 < 0 \\ T \geq -\frac{(r_2 - r_1) + a}{V_2 - 1} \\ T \geq -\frac{(r_2 - r_1) - a}{V_2 - 1} \end{cases} \quad (\text{A.16})$$

$$T \geq -\frac{(r_2-r_1)+a}{V_2-1}$$

Combining this with inequality A.3, we have:

$$\frac{a\sqrt{V^2+V_2^2}-(r_2-r_1)V}{V(V_2-1)} \leq T \leq -\frac{(r_2-r_1)-a}{V_2-1} \quad (\text{A.17})$$

Combining A.5, A.11 and A.13, we have:

$$\begin{cases} V_2 > 1 \\ 0 \leq T \leq \frac{a\sqrt{V^2+V_2^2}-(r_2-r_1)V}{V(V_2-1)} \end{cases} \quad (\text{A.18})$$

Combining A.8, A.15 and A.17, we have:

$$\begin{cases} V_2 < 1 \\ 0 \leq T \leq -\frac{a\sqrt{V^2+V_2^2}-(r_2-r_1)V}{V(V_2-1)} \end{cases} \quad (\text{A.19})$$

A.1.2 The time range for a right-left departure curve

Refer to Figure A.1 (b).

$C_1 : (o_1, r_1, 1)$

$C_2 : (o_2, r_2, V_2), V_2 \neq 1$

T : travel time of C_1

t : travel time of the robot, with speed V

$a = |o_1 o_2|$

$a^2 = (r_2 + V_2(T+t) + r_1 + T)^2 + V^2 t^2$

$(V_2^2 + V^2)t^2 + 2[(V_2 + 1)T + (r_2 + r_1)]V_2 t + [(V_2 + 1)T + (r_2 + r_1)]^2 - a^2 = 0$

$t = \frac{-V_2[(V_2+1)T+(r_2+r_1)] \pm \sqrt{-V^2[(V_2+1)T+(r_2+r_1)]^2 + a^2 V_2^2 + a^2 V^2}}{V_2^2 + V^2}$

When the right-left departure curve exists, $t \geq 0$, we have:

$$\begin{cases} -V^2[(V_2 + 1)T + (r_2 + r_1)]^2 + a^2 V_2^2 + a^2 V^2 \geq 0 & (a) \\ -V_2[(V_2 + 1)T + (r_2 + r_1)] \pm \sqrt{-V^2[(V_2 + 1)T + (r_2 + r_1)]^2 + a^2 V_2^2 + a^2 V^2} \geq 0 & (b) \end{cases} \quad (\text{A.20})$$

Solving inequality (a), $a^2 V_2^2 + a^2 V^2 \geq V^2[(V_2 + 1)T + (r_2 + r_1)]^2$

$$-\frac{a\sqrt{V_2^2 + V^2} + V(r_2 + r_1)}{V(V_2 + 1)} \leq T \leq \frac{a\sqrt{V_2^2 + V^2} - V(r_2 + r_1)}{V(V_2 + 1)} \quad (\text{A.21})$$

Solving inequality (b),

$$\pm \sqrt{-V^2[(V_2 + 1)T + (r_2 + r_1)]^2 + a^2 V_2^2 + a^2 V^2} \geq V_2[(V_2 + 1)T + (r_2 + r_1)]$$

$$\begin{aligned}
 + : -V^2[(V_2 + 1)T + (r_2 + r_1)]^2 + a^2V_2^2 + a^2V^2 &\geq V_2^2[(V_2 + 1)T + (r_2 + r_1)]^2 \\
 (V_2 + 1)^2T^2 + 2(V_2 + 1)(r_2 + r_1)T + (r_2 + r_1)^2 &\leq a^2 \\
 [(V_2 + 1)T + (r_2 + r_1)]^2 &\leq a^2 \\
 -\frac{a+(r_2+r_1)}{V_2+1} \leq T \leq \frac{a-(r_2+r_1)}{V_2+1}
 \end{aligned}$$

Combining this with inequality A.21, we have:

$$0 \leq T \leq \frac{a - (r_2 + r_1)}{V_2 + 1} \quad (\text{A.22})$$

$$\begin{aligned}
 - : -V^2[(V_2 + 1)T + (r_2 + r_1)]^2 + a^2V_2^2 + a^2V^2 &\leq V_2^2[(V_2 + 1)T + (r_2 + r_1)]^2 \\
 [(V_2 + 1)T + (r_2 + r_1)]^2 &\geq a^2 \\
 T \geq \frac{a-(r_2+r_1)}{V_2+1} \text{ or } T \leq -\frac{a+(r_2+r_1)}{V_2+1} &\text{ (no solution)}
 \end{aligned}$$

Combining this with inequality A.21 and A.23, we have:

$$0 \leq T \leq \frac{a\sqrt{V_2^2 + V^2} - V(r_2 + r_1)}{V(V_2 + 1)} \quad (\text{A.23})$$

A.2 Compute the formula for a departure curve in various-speed case

This computation follows van den Berg and Overmars.

$C_0 : (o_0, r_0, 1)$ with $o_0 : (0, 0)$, $r_0 = 0$, when $t = 0$

$C_i : (o_i, r_i, V_i)$, $V_i \neq 1$, and $o_i : (a, b)$

T : travel time of C_0

t : time

V : travel speed of the robot

The equation of spiral is represented in the polar coordinates $(r(t), \Theta(t))$, with parameter t , refer to Figure 5.6 for an example. $r(t) = t$. For the $\Theta(t)$, we have:

$$\sqrt{x'(t)^2 + y'(t)^2} = V \quad (\text{A.24})$$

$$\begin{cases} x(t) = r(t) \cos \Theta(t) \\ y(t) = r(t) \sin \Theta(t) \end{cases} \quad (\text{A.25})$$

$$\begin{cases} x'(t) = r'(t) \cos \Theta(t) - r(t) \Theta'(t) \sin \Theta(t) \\ y'(t) = r'(t) \sin \Theta(t) + r(t) \Theta'(t) \cos \Theta(t) \end{cases} \quad (\text{A.26})$$

$$x'(t)^2 + y'(t)^2 = r'(t)^2 + r(t)^2 \Theta'(t)^2 = 1 + t^2 \Theta'(t)^2 \quad (\text{A.27})$$

Combining Equation A.24 and A.27, we have:

$$\sqrt{1 + t^2 \Theta'(t)^2} = V, 1 + t^2 \Theta'(t)^2 = V^2, \Theta'(t) = \pm \frac{\sqrt{V^2 - 1}}{t}$$

$$\begin{cases} \Theta(t) = \pm \sqrt{V^2 - 1} \log t + \Theta_0 \\ r(t) = t \end{cases} \quad (\text{A.28})$$

The $\pm$ in Equation A.28 indicates the spiral is counterclockwise (+), or clockwise (−) around the growing disc. Θ_0 is the starting angle of the spiral. The growing disc C_0 can be represented by two parameters time T and Θ :

$$C_0(T, \Theta) \longrightarrow \{T \cos \Theta, T \sin \Theta, T\}$$

Considering the CCW spirals around C_0 . Each of the spirals is uniquely defined by the initial angle Θ_0 . Each point (T, Θ) on the growing disc C_0 has a unique spiral that goes through that point. This spiral can be found by solving $\Theta(T) = \Theta$ for Θ_0 .

$$\Theta(t) = \sqrt{V^2 - 1} \log t + \Theta_0$$

$$\Theta_0 = -\sqrt{V^2 - 1} \log T + \Theta$$

Hence, the spiral through (T, Θ) is described in Euclidean coordinate

$$\begin{cases} x(t) = r(t) \cos \Theta(t) = t \cos \Theta(t) = t \cos(\sqrt{V^2 - 1} \log t + \Theta_0) \\ y(t) = r(t) \sin \Theta(t) = t \sin \Theta(t) = t \sin(\sqrt{V^2 - 1} \log t + \Theta_0) \end{cases} \quad (\text{A.29})$$

$$\begin{cases} x(t) = t \cos(\sqrt{V^2 - 1} \log t - \sqrt{V^2 - 1} \log T + \Theta) \\ y(t) = t \sin(\sqrt{V^2 - 1} \log t - \sqrt{V^2 - 1} \log T + \Theta) \end{cases} \quad (\text{A.30})$$

If we walk along this spiral, we can depart for another disc C_i , if the straight line segment tangent to this spiral is tangent to C_i as well. The straight line segment l tangent to the spiral at point (T, Θ) is represented by:

$$l(t) = \{x(T) + (t - T)x'(T), y(T) + (t - T)y'(T)\}$$

$$\begin{aligned} x'(t) &= \cos(\sqrt{V^2 - 1} \log t - \sqrt{V^2 - 1} \log T + \Theta) \\ &\quad - t \frac{\sqrt{V^2 - 1}}{t} \sin(\sqrt{V^2 - 1} \log t - \sqrt{V^2 - 1} \log T + \Theta) \\ x'(T) &= \cos \Theta - \sqrt{V^2 - 1} \sin \Theta \end{aligned}$$

$$\begin{aligned}
x(T) + (t - T)x'(T) &= T \cos(\sqrt{V^2 - 1} \log T - \sqrt{V^2 - 1} \log T + \Theta) + (t - T) \cos \Theta \\
&\quad - (t - T) \sqrt{V^2 - 1} \sin \Theta \\
&= T \cos \Theta + t \cos \Theta - T \cos \Theta - (t - T) \sqrt{V^2 - 1} \sin \Theta \\
&= T \cos \Theta - (t - T) \sqrt{V^2 - 1} \sin \Theta
\end{aligned}$$

$$\begin{aligned}
y'(t) &= \sin(\sqrt{V^2 - 1} \log t - \sqrt{V^2 - 1} \log T + \Theta) \\
&\quad + t \frac{\sqrt{V^2 - 1}}{t} \cos(\sqrt{V^2 - 1} \log t - \sqrt{V^2 - 1} \log T + \Theta)
\end{aligned}$$

$$y'(T) = \sin \Theta + \sqrt{V^2 - 1} \cos \Theta$$

$$\begin{aligned}
y(T) + (t - T)y'(T) &= T \sin(\sqrt{V^2 - 1} \log T - \sqrt{V^2 - 1} \log T + \Theta) + (t - T) \sin \Theta \\
&\quad + (t - T) \sqrt{V^2 - 1} \cos \Theta \\
&= T \sin \Theta + t \sin \Theta - T \sin \Theta + (t - T) \sqrt{V^2 - 1} \cos \Theta \\
&= T \sin \Theta + (t - T) \sqrt{V^2 - 1} \cos \Theta
\end{aligned}$$

So,

$$l(t) = \{T \cos \Theta - (t - T) \sqrt{V^2 - 1} \sin \Theta, T \sin \Theta + (t - T) \sqrt{V^2 - 1} \cos \Theta\} \quad (\text{A.31})$$

The tangent $l(t)$ must be tangent to disc $C_i = (o_i, r_i, V_i)$, in order for point (T, Θ) to be on a departure curve $DC(l_{o_i})$. The function of C_i is:

$$C_i : (x - a)^2 + (y - b)^2 = (V_i t + r_i)^2, \text{ where, } o_i = (a, b).$$

We fill the above equation by substituting $x = l_x(t)$ and $y = l_y(t)$ in Equation A.31, and solve for t . We get a solution of the following form:

$$t_{1,2} = A(T, \Theta) \pm \sqrt{D(T, \Theta)}$$

When $D(T, \Theta) = 0$, l is tangent to C_i . By solving $D(T, \Theta) = 0$ for T , we obtain the functions of the departure curves. (The arrival time on C_i is given by $A(T, \Theta)$.) The details are as follows.

$$\begin{aligned}
C_1 : (x - a)^2 + (y - b)^2 &= (V_1 t + r_1)^2, \text{ put } l_x(t) \text{ and } l_y(t) \text{ in Equation A.31.} \\
(t \cos \Theta - (t - T) \sqrt{V^2 - 1} \sin \Theta - a)^2 &+ (t \sin \Theta + (t - T) \sqrt{V^2 - 1} \cos \Theta - b)^2 = \\
(V_1 t + r_1)^2 & \\
(\cos \Theta - \sqrt{V^2 - 1} \sin \Theta)^2 t^2 &+ 2(\sqrt{V^2 - 1} \sin \Theta - a)(\cos \Theta - \sqrt{(V^2 - 1) \sin \Theta})t + \\
(T \sqrt{V^2 - 1} \sin \Theta - a)^2 &+ (\sin \Theta + \sqrt{V^2 - 1} \cos \Theta)^2 t^2 - 2(\sqrt{V^2 - 1} \cos \Theta + b)(\sin \Theta + \\
\sqrt{(V^2 - 1) \cos \Theta})t &+ (T \sqrt{V^2 - 1} \cos \Theta + b)^2 = V_1^2 t^2 + 2V_1 r_1 t + r_1^2
\end{aligned}$$

$$\begin{aligned}
 & (V^2 - V_1^2)t^2 + 2[(\sqrt{V^2 - 1} \sin \Theta T - a)(\cos \Theta - \sqrt{V^2 - 1} \sin \Theta) - (\sqrt{V^2 - 1} \cos \Theta T + \\
 & b)(\sin \Theta + \sqrt{V^2 - 1} \cos \Theta) - V_1 r_1]t + (T\sqrt{V^2 - 1} \sin \Theta - a)^2 + (T\sqrt{V^2 - 1} \cos \Theta + b)^2 - \\
 & r_1^2 = 0 \\
 & (V^2 - V_1^2)t^2 + 2[-(V^2 - 1)T - (a \cos \Theta + b \sin \Theta) + \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - \\
 & V_1 r_1]t + (V^2 - 1)T^2 + 2(b\sqrt{V^2 - 1} \cos \Theta - a\sqrt{V^2 - 1} \sin \Theta)T + a^2 + b^2 - r_1^2 = 0 \\
 & \text{Let } A = V^2 - V_1^2, \\
 & B = -(V^2 - 1)T - (a \cos \Theta + b \sin \Theta) + \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - V_1 r_1, \\
 & C = (V^2 - 1)T^2 + 2(b\sqrt{V^2 - 1} \cos \Theta - a\sqrt{V^2 - 1} \sin \Theta)T + a^2 + b^2 - r_1^2
 \end{aligned}$$

$$t = \frac{-B \pm \sqrt{B^2 - AC}}{V^2 - V_1^2} = \frac{A(T, \Theta) \pm \sqrt{D(T, \Theta)}}{V^2 - V_1^2} \quad (\text{A.32})$$

Note: there should be $B^2 - 4AC$ inside. But we define B without coefficient 2 before it. Same as $B'^2 - A'C'$ later.

$$\begin{aligned}
 B^2 &= (V^2 - 1)^2 T^2 + 2(V^2 - 1)[(a \cos \Theta + b \sin \Theta) - \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) + \\
 & V_1 r_1]T + [(a \cos \Theta + b \sin \Theta) - \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) + V_1 r_1]^2 \\
 AC &= (V^2 - 1)[(V^2 - 1)T^2 - 2\sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta)T + a^2 + b^2 - r_1^2] \\
 B^2 - AC &= (V^2 - 1)(V_1^2 - 1)T^2 + 2[(V^2 - 1)(a \cos \Theta + b \sin \Theta) + (V^2 - 1)V_1 r_1 + \\
 & \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - V_1^2 \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta)]T + (a \cos \Theta + b \sin \Theta + \\
 & V_1 r_1)^2 + (V^2 - 1)(a \sin \Theta - b \cos \Theta)^2 - 2(a \cos \Theta + b \sin \Theta + V_1 r_1)\sqrt{V^2 - 1}(a \sin \Theta - \\
 & b \cos \Theta) - (V^2 - V_1^2)(a^2 + b^2 - r_1^2)
 \end{aligned}$$

In Equation A.32, when $D(T, \Theta) = 0$, segment l is tangent to C_i . $D(T, \Theta)$ is an implicit equation for the right-right and right-left departure curve ($DC(l_{oi})$).

Solving $D(T, \Theta) = 0$, we have:

$$T = \frac{-B' \pm \sqrt{B'^2 - A'C'}}{(V^2 - 1)(V_1^2 - 1)} \quad (\text{A.33})$$

where,

$$\begin{aligned}
 A' &= (V^2 - 1)(V_1^2 - 1) \\
 B' &= (V^2 - 1)(a \cos \Theta + b \sin \Theta) - (V_1^2 - 1)\sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) + (V^2 - 1)V_1 r_1 \\
 C' &= (a \cos \Theta + b \sin \Theta + V_1 r_1)^2 + (V^2 - 1)(a \sin \Theta - b \cos \Theta)^2 - 2(a \cos \Theta + b \sin \Theta + \\
 & V_1 r_1)\sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - (V^2 - V_1^2)(a^2 + b^2 - r_1^2) \\
 B'^2 &= (V^2 - 1)^2(a \cos \Theta + b \sin \Theta + V_1 r_1)^2 + (V_1^2 - 1)^2(V^2 - 1)(a \sin \Theta - b \cos \Theta)^2 - \\
 & 2(V^2 - 1)(V_1^2 - 1)\sqrt{V^2 - 1}(a \cos \Theta + b \sin \Theta + V_1 r_1)(a \sin \Theta - b \cos \Theta) \\
 A'C' &= (V_1^2 - 1)(V^2 - 1)(a \cos \Theta + b \sin \Theta + V_1 r_1)^2 + (V_1^2 - 1)^2(V^2 - 1)(a \sin \Theta - \\
 & b \cos \Theta)^2 - 2(V^2 - 1)(V_1^2 - 1)\sqrt{V^2 - 1}(a \cos \Theta + b \sin \Theta + V_1 r_1)(a \sin \Theta - b \cos \Theta) - \\
 & (V_1^2 - 1)(V^2 - 1)(V^2 - V_1^2)(a^2 + b^2 - r_1^2) \\
 B'^2 - A'C' &= (V^2 - V_1^2)(V^2 - 1)[(a \cos \Theta + b \sin \Theta + V_1 r_1)^2 - (V_1^2 - 1)(a \sin \Theta - \\
 & b \cos \Theta)^2 + (V_1^2 - 1)(a^2 + b^2 - r_1^2)] \\
 \text{Let } A &= \sqrt{a^2 + b^2}, \sin \alpha = \frac{a}{\sqrt{a^2 + b^2}}, \cos \alpha = \frac{b}{\sqrt{a^2 + b^2}}
 \end{aligned}$$

$$\begin{aligned}
 a \cos \Theta + b \sin \Theta &= \mathbb{A} \left(\frac{a}{\sqrt{a^2+b^2}} \cos \Theta + \frac{b}{\sqrt{a^2+b^2}} \sin \Theta \right) = \mathbb{A} \sin(\Theta + \alpha) \\
 a \sin \Theta - b \cos \Theta &= \mathbb{A} \left(\frac{a}{\sqrt{a^2+b^2}} \sin \Theta - \frac{b}{\sqrt{a^2+b^2}} \cos \Theta \right) = \mathbb{A} \cos(\Theta + \alpha) \\
 B'^2 - A'C' &= (V_1^2 - 1)(V^2 - 1)[(\mathbb{A} \sin(\Theta + \alpha) + V_1 r_1)^2 - (V_1^2 - 1)\mathbb{A}^2 \cos^2(\Theta + \alpha) + \\
 &\quad (V_1^2 - 1)(a^2 + b^2 - r_1^2)] = (V_1^2 - 1)(V^2 - 1)[\mathbb{A}^2 \sin^2(\Theta + \alpha) + V_1^2 r_1^2 + 2\mathbb{A} V_1 r_1 \sin(\Theta + \alpha) - \\
 &\quad V_1^2 \mathbb{A}^2 \cos^2(\Theta + \alpha) + \mathbb{A}^2 \cos^2(\Theta + \alpha) + V_1^2(a^2 + b^2 - r_1^2) - a^2 - b^2 + r_1^2] = (V_1^2 - 1)(V^2 - 1)[a^2 + b^2 + \\
 &\quad V_1^2 r_1^2 + 2\mathbb{A} V_1 r_1 \sin(\Theta + \alpha) - V_1^2 \mathbb{A}^2 + V_1^2 \mathbb{A}^2 \sin^2(\Theta + \alpha) + V_1^2(a^2 + b^2 - r_1^2) - a^2 - b^2 + r_1^2] \\
 &= (V_1^2 - 1)(V^2 - 1)[(\mathbb{A} V_1 \sin(\Theta + \alpha) + r_1)^2 - V_1^2 \mathbb{A}^2 + V_1^2(a^2 + b^2)] = \\
 &= (V_1^2 - 1)(V^2 - 1)[\mathbb{A} V_1 \sin(\Theta + \alpha) + r_1]^2 = (V_1^2 - 1)(V^2 - 1)[V_1(a \cos \Theta + b \sin \Theta) + r_1]^2
 \end{aligned}$$

Equation A.33 can be rewritten as:

$$\begin{aligned}
 T &= \frac{-B' \pm \sqrt{B'^2 - A'C'}}{(V^2 - 1)(V_1^2 - 1)} \\
 &= -\frac{(V^2 - 1)(a \cos \Theta + b \sin \Theta) - (V_1^2 - 1)\sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) + (V^2 - 1)V_1 r_1 \pm \sqrt{V^2 - 1}\sqrt{V^2 - V_1^2}|V_1(a \cos \Theta + b \sin \Theta) + r_1|}{(V_1^2 - 1)(V^2 - 1)} \\
 &= \frac{-[\sqrt{V^2 - 1}(a \cos \Theta + b \sin \Theta) - (V_1^2 - 1)(a \sin \Theta - b \cos \Theta) + \sqrt{V^2 - 1}V_1 r_1] \pm \sqrt{V^2 - V_1^2}|V_1(a \cos \Theta + b \sin \Theta) + r_1|}{\sqrt{V^2 - 1}(V_1^2 - 1)}
 \end{aligned}$$

Let

$$\begin{aligned}
 A'(T, \Theta) &= \frac{-[\sqrt{V^2 - 1}(a \cos \Theta + b \sin \Theta) - (V_1^2 - 1)(a \sin \Theta - b \cos \Theta) + \sqrt{V^2 - 1}V_1 r_1]}{\sqrt{V^2 - 1}(V_1^2 - 1)} \\
 &= -\frac{a \cos \Theta + b \sin \Theta}{V_1^2 - 1} + \frac{a \sin \Theta - b \cos \Theta}{\sqrt{V^2 - 1}} - \frac{V_1 r_1}{V_1^2 - 1} = -\frac{a \cos \Theta + b \sin \Theta + V_1 r_1}{V_1^2 - 1} + \frac{a \sin \Theta - b \cos \Theta}{\sqrt{V^2 - 1}} \\
 D'(T, \Theta) &= \frac{\sqrt{V^2 - V_1^2}|V_1(a \cos \Theta + b \sin \Theta) + r_1|}{\sqrt{V^2 - 1}(V_1^2 - 1)}
 \end{aligned}$$

$$\begin{cases} T = A'(T, \Theta) + D'(T, \Theta) \\ T = A'(T, \Theta) - D'(T, \Theta) \end{cases} \quad (\text{A.34})$$

$$\begin{cases} A'(T, \Theta) + D'(T, \Theta) \geq 0 \\ V_1(a \cos \Theta + b \sin \Theta) + r_1 \geq 0 \\ V_1 > 1, \text{ or } V_1 < 1 \end{cases} \quad (\text{A.35})$$

$$\begin{cases} A'(T, \Theta) + D'(T, \Theta) \geq 0 \\ V_1(a \cos \Theta + b \sin \Theta) + r_1 \leq 0 \\ V_1 < 1, \text{ or } V_1 > 1 \end{cases} \quad (\text{A.36})$$

$$\begin{cases} A'(T, \Theta) - D'(T, \Theta) \geq 0 \\ V_1(a \cos \Theta + b \sin \Theta) + r_1 \geq 0 \\ V_1 > 1, \text{ or } V_1 < 1 \end{cases} \quad (\text{A.37})$$

$$\begin{cases} A'(T, \Theta) - D'(T, \Theta) \geq 0 \\ V_1(a \cos \Theta + b \sin \Theta) + r_1 \leq 0 \\ V_1 < 1, \text{ or } V_1 > 1 \end{cases} \quad (\text{A.38})$$

Solving Equality A.35, for T ,

$$\begin{cases} \frac{-\sqrt{V^2-1}(a \cos \Theta + b \sin \Theta + V_1 r_1) + (V_1^2 - 1)(a \sin \Theta - b \cos \Theta) + \sqrt{V^2 - V_1^2}[V_1(a \cos \Theta + b \sin \Theta) + r_1]}{\sqrt{V^2-1}(V_1^2-1)} \geq 0 \\ V_1(a \cos \Theta + b \sin \Theta) + r_1 \geq 0 \implies V_1 \Delta \sin(\Theta + \alpha) + r_1 \geq 0 \implies \sin(\Theta + \alpha) \geq \frac{-r_1}{V_1 \Delta} \\ V_1 < 1, \text{ or } V_1 > 1 \end{cases} \quad (\text{A.39})$$

$$\begin{aligned} A'(T, \Theta) + D'(T, \Theta) &= -\frac{a \cos \Theta + b \sin \Theta}{V_1^2 - 1} + \frac{a \sin \Theta - b \cos \Theta}{\sqrt{V^2 - 1}} + \frac{\sqrt{V^2 - V_1^2} V_1 (a \cos \Theta + b \sin \Theta)}{\sqrt{V^2 - 1}(V_1^2 - 1)} + \frac{\sqrt{V^2 - V_1^2} r_1}{\sqrt{V^2 - 1}(V_1^2 - 1)} - \frac{V_1 r_1}{V_1^2 - 1} \\ &= -\frac{\Delta}{V_1^2 - 1} \sin(\Theta + \alpha) + \frac{\Delta}{V_1^2 - 1} \cos(\Theta + \alpha) + \frac{\sqrt{V^2 - V_1^2} V_1 \sin(\Theta + \alpha)}{\sqrt{V^2 - 1}(V_1^2 - 1)} + \frac{\sqrt{V^2 - V_1^2} r_1}{\sqrt{V^2 - 1}(V_1^2 - 1)} - \frac{V_1 r_1}{V_1^2 - 1} \\ &= \frac{\sqrt{V^2 - V_1^2} V_1 \Delta - \sqrt{V^2 - 1} \Delta}{\sqrt{V^2 - 1}(V_1^2 - 1)} \sin(\Theta + \alpha) + \frac{\Delta}{\sqrt{V^2 - 1}} \cos(\Theta + \alpha) + \frac{\sqrt{V^2 - V_1^2} r_1}{\sqrt{V^2 - 1}(V_1^2 - 1)} - \frac{V_1 r_1}{V_1^2 - 1} \end{aligned}$$

Let:

$$C = \frac{\sqrt{V^2 - V_1^2} V_1 - \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)},$$

$$D = \frac{1}{\sqrt{V^2 - 1}},$$

$$E = \sqrt{C^2 + D^2} = \frac{-\sqrt{V^2 - V_1^2} + V_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)},$$

$$\cos \beta = \frac{C}{E}, \text{ and } \sin \beta = \frac{D}{E}$$

$$T = \Delta E \sin(\Theta + \alpha + \beta) + \frac{\sqrt{V^2 - V_1^2} - \sqrt{V^2 V_1^2 - V_1^2}}{\sqrt{V^2 - 1}(V_1^2 - 1)} r_1$$

Thus, the function of the departure curve is as the following:

$$\begin{cases} \bar{t} = \Delta E \sin(\theta + \alpha + \beta) + \frac{\sqrt{V^2 - V_1^2} - V_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)} r_1 \\ \sin(\Theta + \alpha) \geq -\frac{r_1}{V_1 \Delta} \\ A = \sqrt{a^2 + b^2}, \sin \alpha = \frac{a}{A}, \cos \alpha = \frac{b}{A} \\ E = \left| \frac{\sqrt{V^2 - V_1^2} - V_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)} \right|, \cos \beta = \frac{V_1 \sqrt{V^2 - V_1^2} - \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)}, \sin \beta = \frac{1}{\sqrt{V^2 - 1}} \end{cases} \quad (\text{A.40})$$

Solving Equality A.36,

$$\begin{aligned} A'(T, \Theta) + D'(T, \Theta) &= -\frac{a \cos \Theta + b \sin \Theta}{V_1^2 - 1} + \frac{a \sin \Theta - b \cos \Theta}{\sqrt{V^2 - 1}} - \frac{\sqrt{V^2 - V_1^2} V_1 (a \cos \Theta + b \sin \Theta)}{\sqrt{V^2 - 1}(V_1^2 - 1)} - \frac{\sqrt{V^2 - V_1^2} r_1 + V_1 r_1 \sqrt{V^2 - 1}}{(V_1^2 - 1)\sqrt{V^2 - 1}} \\ &= -\frac{\sqrt{V^2 - V_1^2} V_1 \Delta + \sqrt{V^2 - 1} \Delta}{\sqrt{V^2 - 1}(V_1^2 - 1)} \sin(\Theta + \alpha) + \frac{\Delta}{\sqrt{V^2 - 1}} \cos(\Theta + \alpha) - \frac{\sqrt{V^2 - V_1^2} r_1 + V_1 r_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)} \end{aligned}$$

Let:

$$C = -\frac{\sqrt{V^2 - V_1^2} V_1 + \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)},$$

$$D = \frac{1}{\sqrt{V^2 - 1}},$$

$$E = \sqrt{C^2 + D^2} = \pm \frac{\sqrt{V^2 - V_1^2} + V_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)}, \text{ where, } V_1 > 1 \text{ " + ", and } V_1 < 1 \text{ " - "},$$

$$\begin{aligned}
 \cos \beta &= \frac{C}{E}, \text{ and } \sin \beta = \frac{D}{E} \\
 T &= AE \sin(\Theta + \alpha + \beta) + \frac{\sqrt{V^2 - V_1^2} - \sqrt{V^2 V_1^2 - V_1^2}}{\sqrt{V^2 - 1}(V_1^2 - 1)} r_1 \\
 \left\{ \begin{array}{l}
 t = AE \sin(\theta + \alpha + \beta) - \frac{\sqrt{V^2 - V_1^2} + V_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)} r_1 \\
 \sin(\Theta + \alpha) \leq -\frac{r_1}{V_1 A} \\
 A = \sqrt{a^2 + b^2}, \sin \alpha = \frac{a}{A}, \cos \alpha = \frac{b}{A} \\
 E = \pm \frac{\sqrt{V^2 - V_1^2} + V_1 \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)} \\
 C = -\frac{\sqrt{V^2 - V_1^2} V_1 + \sqrt{V^2 - 1}}{\sqrt{V^2 - 1}(V_1^2 - 1)}, D = \frac{1}{\sqrt{V^2 - 1}}, \\
 \cos \beta = \frac{C}{E}, \sin \beta = \frac{D}{E}
 \end{array} \right. \quad (\text{A.41})
 \end{aligned}$$

Solving Equation A.37, which is exactly A.36 for T .

Solving Equation A.38, which is exactly A.35 for T .

A.3 Compute the formula for a departure curve with the same-speed discs

$C_0 : (o_0, r_0, 1)$, $o_0 : (0, 0)$, $r_0 = 0$ when $t = 0$

$C_1 : (o_1, r_1, V_1)$, $V_1 \neq 1$, $o_1 : (a, b)$

T : travel time of C_0

t : time

V : travel speed of the robot

The function of growing disc C_1 is:

$$(x - a)^2 + (y - b)^2 = (V_1 t + r_1)^2 = (t + r_1)^2$$

Put the function of tangent l ,

$$l_x(t) = t \cos \Theta - (t - T) \sqrt{V^2 - 1} \sin \Theta,$$

$$l_y(t) = t \sin \Theta + (t - T) \sqrt{V^2 - 1} \cos \Theta \text{ into } C_1, \text{ we have:}$$

$$\begin{aligned}
 & (V^2 - 1)t^2 + 2[(\sqrt{V^2 - 1} \sin \Theta T - a)(\cos \Theta - \sqrt{V^2 - 1} \sin \Theta) - (\sqrt{V^2 - 1} \cos \Theta T + \\
 & b)(\sin \Theta + \sqrt{V^2 - 1} \cos \Theta) - r_1]t + (T \sqrt{V^2 - 1} \sin \Theta - a)^2 + (T \sqrt{V^2 - 1} \cos \Theta + b)^2 - r_1^2 = \\
 & 0
 \end{aligned}$$

$$\begin{aligned}
 & (V^2 - 1)t^2 + 2[-(V^2 - 1)T - (a \cos \Theta + b \sin \Theta) + \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - \\
 & r_1]t + (V^2 - 1)T^2 + 2\sqrt{V^2 - 1}(b \cos \Theta - a \sin \Theta)T + a^2 + b^2 - r_1^2 = 0
 \end{aligned}$$

$$t = \frac{(V^2 - 1)T + (a \cos \Theta + b \sin \Theta) - \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) + r_1 \pm \sqrt{B^2 - AC}}{V^2 - 1}$$

where,

$$A = V^2 - 1$$

$$B = -(V^2 - 1)T - (a \cos \Theta + b \sin \Theta) + \sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - r_1$$

$$\begin{aligned}
 C &= (V^2 - 1)T^2 + 2\sqrt{V^2 - 1}(b \cos \Theta - a \sin \Theta)T + a^2 + b^2 - r_1^2 \\
 B^2 - AC &= 2(V^2 - 1)[a \cos \Theta + b \sin \Theta + r_1]T + [(a \cos \Theta + b \sin \Theta + r_1) + \sqrt{V^2 - 1}(b \cos \Theta - a \sin \Theta)]^2 - (V^2 - 1)(a^2 + b^2 - r_1^2) \\
 &= 2(V^2 - 1)[a \cos \Theta + b \sin \Theta + r_1]T + (a \cos \Theta + b \sin \Theta + r_1)^2 + 2\sqrt{V^2 - 1}(a \cos \Theta + b \sin \Theta + r_1)(b \cos \Theta - a \sin \Theta) + (V^2 - 1)(b^2 - b^2 \sin^2 \Theta + a^2 - a^2 \cos^2 \Theta - 2ab \sin \Theta \cos \Theta - a^2 - b^2 + r_1^2) \\
 &= 2(V^2 - 1)[a \cos \Theta + b \sin \Theta + r_1]T + (a \cos \Theta + b \sin \Theta + r_1)^2 + 2\sqrt{V^2 - 1}(a \cos \Theta + b \sin \Theta + r_1)(b \cos \Theta - a \sin \Theta) - (V^2 - 1)(a \cos \Theta + b \sin \Theta + r_1)(a \cos \Theta + b \sin \Theta - r_1) = 0
 \end{aligned}$$

We have:

$$a \cos \Theta + b \sin \Theta + r_1 = 0, \quad (\text{A.42})$$

and, when $a \cos \Theta + b \sin \Theta + r_1 \neq 0$,

$$2(V^2 - 1)T + (a \cos \Theta + b \sin \Theta + r_1) - 2\sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - (V^2 - 1)(a \cos \Theta + b \sin \Theta - r_1) = 0 \quad (\text{A.43})$$

From Equation A.43, we have:

$$\begin{aligned}
 T &= \frac{(a \cos \Theta + b \sin \Theta) - 2\sqrt{V^2 - 1}(a \sin \Theta - b \cos \Theta) - (V^2 - 1)(a \cos \Theta + b \sin \Theta) + (V^2 - 1)r_1 + r_1}{2(V^2 - 1)} \\
 &= \frac{[2b\sqrt{V^2 - 1} - a(V^2 - 1)] \cos \Theta - [b(V^2 - 1) + 2a\sqrt{V^2 - 1}] \sin \Theta + V^2 r_1}{2(V^2 - 1)} \\
 &= \frac{2b - a\sqrt{V^2 - 1}}{2\sqrt{V^2 - 1}} \cos \Theta - \frac{b\sqrt{V^2 - 1} + 2a}{2\sqrt{V^2 - 1}} \sin \Theta + \frac{V^2 r_1}{2(V^2 - 1)} \\
 \text{Let: } A &= \frac{2b - a\sqrt{V^2 - 1}}{2\sqrt{V^2 - 1}}, \quad B = \frac{b\sqrt{V^2 - 1} + 2a}{2\sqrt{V^2 - 1}} \sin \Theta \text{ and } C = \frac{V^2 r_1}{2(V^2 - 1)} \\
 T &= A \cos \Theta + B \sin \Theta + C
 \end{aligned}$$

Bibliography

- [1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristan. The farthest color voronoi diagram and related problems. *In Proceedings of 17th European Workshop on Computational Geometry, Berlin, Germany, March 26-28*, pages 113–116, 2001.
- [2] P. K. Agarwal and M. Sharir. Davenport–Schinzel sequences and their geometric applications. Technical Report Technical Report DUKE–TR–1995–21, 1995.
- [3] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k points with minimum diameter and related problems. *Journal of Algorithms*, 12(1):38–56, 1991.
- [4] H-K. Ahn, H. Alt, T. Asano, S. W. Bae, P. Brass, O. Cheong, C. Knauer, H-S. Na, C-S. Shin, and A. Wolff. Constructing optimal highways. *In ACM Proceedings of the 13th Australasian Symposium on Theory of Computing*, 65:7–14, 2007.
- [5] J. L. Ambite, G. Barish, C.A. Knoblock, M. Muslea, and J. Oh. Getting from here to there: Interactive planning and agent execution for optimized travel. *Proceedings 18th National Conference on Artificial Intelligence*, pages 862–869, 2002.
- [6] F. Anton, D. Mioc, and C. Gold. Dynamic additively weighted Voronoi diagrams made easy. *Proceedings 10th CCCG, Montréal, Canada*, 1998.
- [7] H. Aonuma, H. Imai, K. Imai, and T. Tokuyama. Maximin location of convex objects in a polygon and related dynamic Voronoi diagrams. *Proceedings of the 6th Annual Symposium on Computational Geometry*, pages 225–234, 1990.
- [8] What Is Service-Oriented Architecture. Last access date 2008-07-24. <http://www.xml.com/pub/a/ws/2003/09/30/soa.html>.
- [9] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008.

- [10] B. Aronov, M. V. Kreveld, R.v. Oostrum, and K. Varadarajan. Facility location on terrains. *Proceedings of 9th International Symposium on Algorithms and Computation, LNCS*, 1533:19–28, 1998.
- [11] Software as a service. Last access date 2008-03-18. http://en.wikipedia.org/wiki/Software_as_a_Service.
- [12] P. F. Ash and E. D. Bolker. Generalized Dirichlet tessellations. *Geometriae Dedicata*, 20(2):209–243, 1986.
- [13] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987.
- [14] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.
- [15] F. Aurenhammer and H. Edelsbrunner. An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recognition*, 17(2):251–257, 1984.
- [16] F. Aurenhammer and R. Klein. Voronoi diagrams. *Handbook of Computational Geometry, eds., Jörg-Rüdiger Sack and Jorge Urrutia, Elsevier*, pages 201–290, 2000.
- [17] A. BenHassine and T. Ho. An agent-based approach to solve dynamic meeting scheduling problems with preferences. *Engineering Applications of Artificial Intelligence*, 20(6):857–873, 2007.
- [18] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi. A meeting scheduling problem respecting time and space. *the 4th International Conference on Algorithmic Aspects in Information and Management, AAIM'08, June 23-25, Shanghai, China, LNCS*, 5034:50–59, 2008.
- [19] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi. A meeting scheduling problem respecting time and space. *GeoInformatica*, 2008.
- [20] O. Berkman, B. Schieber, and U. Vishkin. Some doubly logarithmic optimal parallel algorithm based on finding all nearest smaller values. *Technical Report, University of Maryland, UMIACS-TR-88-79*, 1988.
- [21] J-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in Computational Geometry. *Discrete and Computational Geometry*, 8:51–71, 1992.

- [22] B. N. Boots. Weighting Thiessen polygons. *Economic Geography*, 56(3):248–259, 1980.
- [23] P. Bose, A. Maheshwari, and P. Morin. Fast approximations for sums of distances, clustering and the Fermat – Weber problem. *Computational Geometry: Theory and Applications*, 24(3):135–146, 2002.
- [24] J. Cardinal and S. Langerman. Min-max-min geometric facility location problems. In *Proceedings of the European Workshop on Computational Geometry EWCG06*, pages 149–152, 2006.
- [25] R. Chandrasekaran and A. Tamir. Algebraic optimization: the Fermat – Weber location problem. *Mathematical Programming: Series A and B*, 46(2):219–224, 1990.
- [26] E. C. Chang, S. W. Choi, D. Y. Kwon, H. Park, and C. K. Yap. Shortest path amidst disc obstacles is computable. *Proceedings of the 21st annual Symposium on Computational Geometry (SoCG)*, pages 116–125, 2005.
- [27] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM (JACM)*, 39(1):1–54, 1992.
- [28] A. Chun, H. Wai, and R.Y.M. Wong. Optimizing agent-based meeting scheduling through preference estimation. *Engineering Applications of Artificial Intelligence*, 16(7-8):727–743, 2003.
- [29] World Wide Web Consortium. Last access date 2008-03-18. <http://www.w3.org/>.
- [30] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational geometry: algorithms and applications, second edition. *Springer-Verlag*, 2000.
- [31] R. Descartes. Geometry / translated from the French and Latin by David Eugene Smith and Marcia L. Latham; with a facsimile of the first edition, 1637. 1954.
- [32] O. Devillers and M. J. Golin. Incremental algorithms for finding the convex hulls of circles and the lower envelopes of parabolas. *Information Processing Letters*, 56(3):157–164, 1995.
- [33] M. d’Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dmars. In *M. P. Singh, A. S. Rao, and M. Wooldridge, editors: Intelligent Agents IV Springer-Verlag Lecture Notes in AI*, 1365, 1998.

- [34] R. A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, 6(4):343–367, 1991.
- [35] L. Ehrler, M. A. Fleurke, and B.T.R. Savarimuthu. Agent-based workflow management systems(wfmss) : JBees- a distributed and adaptive WFMS with monitoring and controlling capabilities. *Journal of Information Systems and e-Business*, 4(1):5–23, 2006.
- [36] P. Epstein, J. Kavanagh, A. Knight, J. May, T. Nguyen, and J. R. Sack. A workbench for computational geometry. *Algorithmica*, 11(4):404–428, 1994.
- [37] L. De Floriani, E. Puppo, and P. Magillo. Applications of computational geometry to geographic information system, 2000.
- [38] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.
- [39] JADE Java Agent DEvelopment Framework. Last access date 2008-07-03. 2006. <http://jade.tilab.com>.
- [40] Anka Gajentaan and Mark H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5(3):165–185, 1995.
- [41] M. Gavrilova. Proximity and applications in general metrics. *Ph. D. Thesis, University of Calgary, Dept. of Computer Science, Canada*, 1998.
- [42] E. Gervais. Dynamap, a mobile and context-aware pedestrian navigation application. *Master Thesis, Carleton University, School of Computer Science*, 2005.
- [43] E. Gervais, H. Liu, D. Nussbaum, J.-R. Sack, Y.-S. Roh, and J. Yi. Intelligent map agents - a ubiquitous personalized GIS. *ISPRS Journal of Photogrammetry and Remote Sensing, special issue on Distributed Geoinformatics*, 62(5):347–365, 2007.
- [44] E. Gervais, D. Nussbaum, and J.-R. Sack. Dynamap: a context aware dynamic map application. *GISPlanet*, May 2005.
- [45] J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th ACM Symposium on Advances in GIS*, pages 35–42, 2006.
- [46] E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jber. Deutsch. Math. Verein.*, 32:175–176, 1923.

- [47] K. Hornsby and M. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):177–194, 2002.
- [48] Md. N. Huda, E. Kamioka, and S. Yamada. An efficient and privacy-aware meeting scheduling scheme using common computational space. *IEICE Transactions*, 90-D(3):656–667, 2007.
- [49] F. Hurtado, R. Klein, E. Langetepe, and V. Sacristán. The weighted farthest color Voronoi diagram on trees and graphs. *Computational Geometry: Theory and Applications*, 27:13–26, 2004.
- [50] C. D. Hyson and W. P. Hyson. The economic law of market areas. *Quarterly Journal of Economics*, 64(2):319–327, 1950.
- [51] K. Imai and H. Imai. Dynamic weighted Voronoi diagrams and weighted min-max matching of two corresponding point sets. *Optimization Methods and Software*, 10(2):261–274, 1998.
- [52] M. Inaba, N. Katoh, and H. Imai. Variance-based k -clustering algorithms by Voronoi diagrams and randomization. *IEICE Transactions on Information and Systems*, E83-D(6):1199–1206, 2000.
- [53] Geographic information system. Last access date 2008-03-18. http://en.wikipedia.org/wiki/Geographic_information_system.
- [54] N. R. Jennings and A. J. Jackson. Agent-based meeting scheduling: A design and implementation. *IEE Electronics Letters*, 31(5):350–352, 1995.
- [55] J.L.Bentley and T.Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, pages 643–647, 1979.
- [56] D. Johansen, R. Van Renesse, and F. B. Schneider. Operating system support for mobile agents. *Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems, Orcas Island, WA, USA.*, 1995.
- [57] M. I. Karavelas and I. Z. Emiris. Dynamic additively weighted Voronoi diagrams in 2d. *Proceedings of the 10th Annual European Symposium on Algorithms, LNCS*, 2461:586–598, 2002.
- [58] D. Kim, D. S. Kim, and K. Sugihara. Euclidean Voronoi diagram for circles in a circle. *International Journal of Computational Geometry*, 15(2):209–228, 2005.

- [59] D. S. Kim, K. Yu, Y. Cho, D. Kim, and C. Yap. Shortest path for disc obstacles. *Proceedings International Conference on Computational Science and its Applications, Assisi, Italy, May 14-17, 2004, LNCS*, 3045:62–70, 2004.
- [60] R. Klein. Concrete and abstract Voronoi diagrams. *LNCS* 400, 1987.
- [61] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Computational Geometry: Theory and Applications*, 3(3):157–184, 1993.
- [62] R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi. How to fit in another meeting. *IEEE The 2nd International Conference on Collaborative Computing: Networking, Applications and Worksharing, Atlanta, U.S.A, November 17-20, 2006*.
- [63] S. Lamy, A. Ruas, Y. Demazeau, C. Baeijs, M. Jackson, W. Mackaness, and R. Weibel. Agent project: Automated generalisation new technology. In *Proceedings of the 5th EC-GIS Workshop*, 1999.
- [64] D. Lange and M. Oshima. Programming and deploying mobile agents with Java aglets. *Addison-Wesley*, 1998.
- [65] M. Laukkanen, H. Helin, and H. Laamanen. Tourists on the move. In *Cooperative Information Agents VI, 6th International Workshop, LNCS 2446*, pages 36–50, Madrid, Spain, 2002.
- [66] S. M. LaValle. Planning algorithms. *Chapter 2, Cambridge University Press*, pages 27–76, 2006.
- [67] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Transactions on Computers*, C-31(6):478–487, 1982.
- [68] C. J. Liau. A modal logic framework for multi-agent belief fusion. *ACM Transactions on Computational Logic (TOCL)*, 6(1):124–174, 2005.
- [69] J. Lieberman. OpenGIS web services architecture, Open Geospatial Consortium specification 03-025. Jan. 2003. http://portal.opengeospatial.org/files/?artifact_id=1320.
- [70] A. Maheshwari, D. Nussbaum, J.-R. Sack, and J. Yi. An $O(n^2 \log n)$ time algorithm for computing shortest paths amidst growing discs in the plane. *the 18th International Symposium on Algorithms and Computation (ISAAC), LNCS*, 4835:668–680, 2007.

- [71] A. Maheshwari, D. Nussbaum, J.-R. Sack, and J. Yi. Shortest paths amidst growing discs in the plane. *Technical Report, School of Computer Science, Carleton University, Canada*, 2008.
- [72] R. Malaka and A. Zipf. Deep Map - challenging IT research in the framework of a tourist information system. *Information and Communication Technologies in Tourism 2000*, pages 15–27, 2000.
- [73] J. Matoušek. On geometric optimization with few violated constraints. *Proceedings 10th Annual Symposium on Computational Geometry (SOCG)*, pages 312–321, 1994.
- [74] N. Megiddo. The weighted Euclidean 1-center problem. *Mathematics of Operations Research*, 8:498–504, 1983.
- [75] N. Megiddo. On the ball spanned by balls. *Discrete & Computational Geometry*, 4:605–610, 1989.
- [76] K. Mehlhorn, S. Meiser, and R. Rasch. Furthest site abstract Voronoi diagrams. *International Journal of Computational Geometry & Applications*, 11(6):583–616, 2001.
- [77] E. Melachrinoudis and Z. Xanthopoulos. Semi-obnoxious single facility location in Euclidean space. *Computers and Operations Research*, 30(14):2191–2209, 2003.
- [78] P. Misikangas, M. Makela, and K. Raatikainen. Predicting quality-of-service for nomadic applications using intelligent agents. *Agent Technology for Communication Infrastructures*, Chapter 15:197–208, 2001.
- [79] Joseph S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, eds., Jörg-Rüdiger Sack and Jorge Urrutia, Elsevier, pages 633–701, 2000.
- [80] J.S.B Mitchell. Geometric shortest paths and network optimization. *Handbook of Computational Geometry*, eds., Jörg-Rüdiger Sack and Jorge Urrutia, Elsevier, pages 633–701, 2000.
- [81] P. Moraitis, E. Petraki, and N. Spanoudakis. Engineering JADE agents with the Gaia methodology. In R. Kowalszyk, J. Miller, H. Tianfield, and R. Unland, editors, *Agent Technologies, Infrastructures, Tools, and Applications for e-Services, LNCS 2592*, pages 77–91, 2003.

- [82] L. Mu. Polygon characterization with the multiplicatively weighted Voronoi diagram. *Professional Geographer*, 56(2):223–239, 2004.
- [83] D. Nussbaum, J.-R. Sack, and H. Ye. Concurrent parallel shortest path computation. *Parallel Computing, Malaga, Spain, September 2005, edited as Parallel Computing: Current&Future Issues of High-End Computing by G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, R. Tirado, and E. Zapata, NIC*, 33:277–284, 2006.
- [84] H. S. Nwana. Software agents: an overview. *Knowledge Engineering Review*, 11(3):1–40, 1996.
- [85] Y. Ohsawa. A geometrical solution for quadratic bicriteria location models. *European Journal of Operational Research*, 114(2):380–388, 1999.
- [86] Y. Ohsawa. Bicriteria Euclidean location associated with maximin and minimax criteria. *Naval Research Logistics*, 47(7):581–592, 2000.
- [87] Y. Ohsawa and A. Imai. Degree of locational freedom in a single facility Euclidean minimax location model. *Location Science*, 5(1):29–45, 1997.
- [88] Y. Ohsawa, F. Plastria, and K. Tamura. Euclidean push-pull partial covering problems. *Computers and Operations Research*, 33(12):3566–3582, 2006.
- [89] T. Ohshima. Some Voronoi diagrams that consider consumer behavior analysis. In *Japan Journal of Industrial and Applied Mathematics*, volume 22, pages 279–290, 2005.
- [90] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. Spatial tessellations: concepts and applications of Voronoi diagrams, 2nd edition. 2000.
- [91] Y. Ostrovsky-Berman. The transportation metric and related problems. *Information Processing Letters*, 95(4):461–465, 2005.
- [92] E. Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40:63–82, 2004.
- [93] E. Papadopoulou and D. T. Lee. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica*, 20(4):319–352, 1998.
- [94] G. A. Pinto and P. J. Rezende. Additively weighted Voronoi diagram on the oriented projective plane. *Proceedings 12th CCCG, August 16-19, New Brunswick, Canada*, pages 119–126, 2000.

- [95] F. Plastria and E. Carrizosa. Minimax geometric fitting of two corresponding sets of points and dynamic furthest Voronoi diagrams. *IEICE TRANSACTIONS on Information and Systems*, E81-D(11):1162–1171, 1998.
- [96] F. Plastria and E. Carrizosa. Undesirable facility location with minimal covering objectives. *European Journal of Operational Research*, 119:158–180, 1999.
- [97] S. Poslad, H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf. Creation of user-friendly mobile tourism services personalised for tourism. In *Proceedings 2nd International Conference on 3G Mobile Communication Technologies*, pages 28–32, London, UK, 2001.
- [98] D. Rappaport. Computing the furthest Voronoi diagram for a set of discs. *Workshop on Algorithms and Data Structures (WADS), LNCS 382*, pages 57–66, 1989.
- [99] D. Rappaport. A convex hull algorithm for discs, and applications. *Computational Geometry: Theory and Applications*, 1:171–187, 1992.
- [100] R. V. Rasmussen and M. A. Trick. The timetable constrained distance minimization problem. In *CPAIOR, Cork, Ireland, May 31-June 2, LNCS 3990*, pages 167–181, 2006.
- [101] F. Rinaldi and P. Serafini. Scheduling school meetings. In *PATAT, Brno, Czech Republic, Aug. 30- Sept. 1, LNCS 3867*, pages 280–293, 2006.
- [102] H. Rosenberger. Order-k Voronoi diagrams of sites with additive weights in the plane. *Algorithmica*, 6:490–521, 1991.
- [103] P. Santos and H. Vaughn. Where shall we meet? Proposing optimal locations for meetings. presented at MapISNet’07.
- [104] B. F. Schaudt. Multiplicatively weighted crystal growth Voronoi diagram. *Ph.D. Thesis, Dartmouth College*, 1992.
- [105] B. Schmidt-Belz, S. Poslad, A. Nick, and A. Zipf. Personalized and location-based mobile tourism services. In *Workshop on Mobile Tourism Support Systems*, Pisa, Italy, 17 September, 2002.
- [106] D. Schmitt and J-C. Spehner. Order-k Voronoi diagrams, k-sections, and k-sets. *Discrete and Computational Geometry, Japanese Conference, JCDCG’98, December 9-12, LNCS*, 1763:290–304, 1998.
- [107] S. Sen. Developing an automated distributed meeting scheduler. *IEEE Expert: Intelligent Systems and Their Applications*, 12(4):41–45, 1997.

- [108] Web Mapping Service. Last access date July 3rd, 2008. <http://www.opengeospatial.org/standards/wms>.
- [109] E. Shakshuki, H.-H. Koo, D. Benoit, and D. Silver. A distributed multi-agent meeting scheduler. *Journal of Computer and System Sciences*, 74(2):279–296, 2008.
- [110] M. Shamos and D. Hoey. Closest-point problems. *Proceedings 16th IEEE Symposium on Foundations of Computer Science, October 13-15, The University of California, Berkeley, CA, USA*, pages 151–162, 1975.
- [111] M. Shamos and D. Hoey. Geometric intersection problems. In *SFCS '76: Proceedings of the 17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 208–215, Washington, DC, USA, 1976. IEEE Computer Society.
- [112] M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM Journal of Computing*, 14(2):448–468, 1985.
- [113] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proceedings 1992 Symposium on Theoretical Aspects of Computer Science. LNCS*, 577:569–579, 1992.
- [114] J. van den Berg. Path planning in dynamic environments. *Ph.D. Thesis, Utrecht University, Utrecht, The Netherlands*, 2007.
- [115] J. van den Berg and M. Overmars. Planning the shortest safe path amidst unpredictably moving obstacles. *Proceedings Workshop on Algorithmic Foundations of Robotics, New York City, U.S.A., July 16-18*, 2006.
- [116] J. Wang, C. Niu, and R. Shen. Scheduling meetings in distance learning. In *APPT, Guangzhou, China, November 22-23, LNCS 4847*, pages 580–589, 2007.
- [117] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1991.
- [118] K. R. Weringh. Algorithms for the Voronoi diagram of a set of disks. *M.S. thesis, Department of Computing and Information Science, Queen's University, Kingston, Canada*, 1990.
- [119] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

- [120] A. Zipf. User-adaptive maps for location-based services (LBS) for tourism. In *Proceedings of the 9th International Conference for Information and Communication Technologies in Tourism*, volume 1, pages 183–197, Innsbruck, Austria, 2002.