Mobile Backbone Architecture for Wireless Ad-Hoc Networks: Algorithms and Performance Analysis

by

Anand Srinivas

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in the field of Communications and Networks

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author .... [illegible]
Department of Aeronautics and Astronautics
June 19, 2007

Certified by .... [illegible]
Eytan H. Modiano
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .... [illegible]
Nicholas Roy
Assistant Professor of Aeronautics and Astronautics

Certified by .... [illegible]
Moe Z. Win
Associate Professor of Aeronautics and Astronautics

Accepted by .... [illegible]
Jaime Peraire
New Chair David Darmofal on behalf of Professor of Aeronautics and Astronautics
Chair, Commitee on Graduate Students

Mobile Backbone Architecture for Wireless Ad-Hoc Networks: Algorithms and Performance Analysis

by

Anand Srinivas

Submitted to the Department of Aeronautics and Astronautics
on June 19, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in the field of Communications and Networks

Abstract

In this thesis, we study a novel hierarchical wireless networking approach in which some of the nodes are more capable than others. In such networks, the more capable nodes can serve as Mobile Backbone Nodes and provide a backbone over which end-to-end communication can take place. The main design problem considered in this thesis is that of how to (i) Construct such Mobile Backbone Networks so as to optimize a network performance metric, and (ii) Maintain such networks under node mobility.

In the first part of the thesis, our approach consists of controlling the mobility of the Mobile Backbone Nodes (MBNs) in order to maintain network connectivity for the Regular Nodes (RNs). We formulate this problem subject to minimizing the number of MBNs and refer to it as the Connected Disk Cover (CDC) problem. We show that it can be decomposed into the Geometric Disk Cover (GDC) problem and the Steiner Tree Problem with Minimum Number of Steiner Points (STP-MSP). We prove that if these subproblems are solved separately by γ- and δ-approximation algorithms, the approximation ratio of the joint solution is $\gamma+\delta$. Then, we focus on the two subproblems and present a number of distributed approximation algorithms that maintain a solution to the GDC problem under mobility. A new approach to the solution of the STP-MSP is also described. We show that this approach can be extended in order to obtain a joint approximate solution to the CDC problem. Finally, we evaluate the performance of the algorithms via simulation and show that the proposed GDC algorithms perform very well under mobility and that the new approach for the joint solution can significantly reduce the number of Mobile Backbone Nodes.

In the second part of the thesis, we address the the joint problem of placing a fixed number K MBNs in the plane, and assigning each RN to exactly one MBN. In particular, we formulate and solve two problems under a general communications model. The first is the Maximum Fair Placement and Assignment (MFPA) problem in which the objective is to maximize the throughput of the minimum throughput RN. The second is the Maximum Throughput Placement and Assignment (MTPA) problem, in which the objective is to maximize the aggregate throughput of the RNs. Due to the change in model (e.g. fixed number of MBNs, general communications model) from the first part of the thesis, the problems of this part of the thesis require a significantly different approach and solution methodology. Our main result is a novel optimal polynomial time algorithm for the MFPA problem for fixed K. For a restricted version of the MTPA problem, we develop an optimal polynomial time algorithm for $K \leq 2$. We also develop two heuristic algorithms for both problems, including an approximation algorithm for which we bound the worst case performance loss. Finally, we present simulation results comparing the performance of the various algorithms developed in the paper.

In the third part of the thesis, we consider the problem of placing the Mobile Backbone Nodes over a finite time horizon. In particular, we assume complete a-priori knowledge of each of the RNs' trajectories over a finite time interval, and consider the problem of determining the optimal MBN *path* over that time interval. We consider the path planning of a single MBN and aim to maximize the time-average system throughput. We also assume that the velocity of the MBN factors into the performance objective (e.g. as a constraint/penalty). Our first approach is a discrete one, for which our main result is a dynamic programming based approximation algorithm for the path planning problem. We provide worst case analysis of the performance of the algorithm. Additionally, we develop an optimal algorithm for the 1-step velocity constrained path planning problem. Using this as a sub-routine, we develop a greedy heuristic algorithm for the overall path planning problem. Next, we approach the path-planning problem from a continuous perspective. We formulate the problem as an optimal control problem, and develop interesting insights into the structure of the optimal solution. Finally, we discuss extensions of the base discrete and continuous formulations and compare the various developed approaches via simulation.

Thesis Supervisor: Eytan H. Modiano
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

There are many people who have had a significant impact on my experience at MIT. First and foremost I would like to thank Prof. Eytan Modiano for being a terrific advisor. Looking back it seems like there was always more or less the correct balance between allowing me the space and time to formulate a problem and find a solution, getting me to move on from a dead end problem, and during the lulls, pushing me just hard enough to get me back in gear. I also have very fond memories of our meetings, the contents of which ranged anywhere from college basketball to American politics, and of course research, when time permitted.

Next I would like to acknowledge Dr. Gil Zussman, whose contributions towards chapter 2 of this Thesis, my technical writing skills, and my knowledge of random Yiddish words, have been invaluable. The support and constructive suggestions of my thesis committee of Profs. Nick Roy and Moe Win have also been extremely beneficial. I would like to thank Profs. Jon How and Emilio Frazolli for sharing some of their insights on optimal control with me. I would also like to acknowledge the helpful comments of my thesis reader, Dr. Phil Lin.

During my Ph.D. I have moved offices twice, and have been extremely lucky to have wonderful officemates each time. In my Stata center office I had great interactions with Damien Jourdan, Li-Wei Chen, Xin Huang, Parikshit Shah and Paul Njoroge. In particular I must thank "Le" Damien for humouring my terrible french, and for our many discussions on linear programming and general optimization theory. After Stata I had a bit of an office downgrade to building 31, but I got to spend more time with my fellow CNRG-mates, Jun Sun, Murtaza Zafer, Andrew Brezinski, Krishna Jagannathan, Guner Celik and Wajahat Khan. Indeed it's a good thing this move happened towards the end of most of our PhDs, since when all of us were simultaneously in the office very little work got done. In this sense, I should probably acknowledge the contributions of the Starbucks coffee shop chain for the fact that most of my productive research was done there, as opposed to my actual office. Yet, I did have several extremely fruitful technical discussions with my officemates:

optimal control theory with Murtaza, combinatorial optimization with Andrew and optimization and calculus with Jun.

Over my time at MIT I have made several very close friends, without whose friendship my graduate life would have been considerably worse off. Masha, I appreciate all of our tea and coffee times together. Nigel and Vidya, while I didn't appreciate you guys pushing me to go to the gym back then, looking back it was definitely a good thing. Dhananjay, it was great having a fellow South Indian from UofT around here. Shashi, it was good times to share in your anti-research promoting ways and Sonia, it was good to have you back in Boston. Finally, Suri, those were some trying times pulling those all-nighters for advanced algorithms, but we're better off for it - I think.

I would like to acknowledge that my Ph.D. research has been supported by a grant from Draper Laboratory, the National Science Foundation, DARPA/AFOSR through the University of Illinois grant number F49620-02-1-0325, and by NASA grant number NAG3-2835. I am extremely grateful for this support, without which I would not have been able to complete my studies.

Last but far from least, I would like to thank my wonderful wife Nammi for supporting me throughout my Ph.D. Moving to frigid Boston from sunny California was a big sacrifice and besides immediately increasing my standard of living, being married and finally living together and in the same city has been a great experience. I also would like to thank my Mom, whose advice when I first got to MIT and was going through a rough acadamic patch I will never forget: “Anand, you're not there to socialize and make friends - now go and focus on your work!”. Well Mom, despite my best efforts, I still made a few...

Contents

List of Figures

List of Tables

List of Acronyms

ALGO	Algorithm Solution (used in proofs)
CC	Cluster Cover (Algorithm)
CC-1C	Circular Constrained 1-Center (Algorithm)
C-MPP-MFPA	Continuous MPP-MFPA (Problem)
CDC	Connected Disk Cover (Problem)
CDMA	Code-Division Multiple Access
DCC-PMAS	Disk-Cover with Circular Separation P-MAS (Algorithm)
DCR-PMAS	Disk-Cover with Rectangular Separation P-MAS (Algorithm)
DPA	Dynamic-Programming based Approximation (Algorithm)
EDA	Extended Diameter Algorithm
EST	Euclidean Steiner Tree (Problem)
FPH	Farthest Point Heuristic (Algorithm)
GDC	Geometric Disk Cover (Problem)
MANET	Mobile Ad-hoc Networks
MAS	Merge-and-Separate (Algorithm)
MBN	Mobile Backbone Node

MFPA	Maximum Fair Placement and Assignment (Problem)
MIS	Maximal Independent Set
MDS	Minimum Dominating Set (Problem)
MOAC	MObile Area Cover (Algorithm)
MPP	MBN Path Planning (Problem)
MPP-MFPA	MPP with time average MFPA objective function (Problem)
MST	Minimum Spanning Tree (Problem)
MTB	Minimum Time Broadcast (Problem)
MTPA	Maximum Throughput Placement and Assignment (Problem)
NWST	Node-Weighted Steiner Tree (Problem)
O-EDA	Optimized Extended Diameter Algorithm
O-FPH	Optimized Farthest Point Heuristic (Algorithm)
OPT	Optimal Solution (used in proofs)
P-MAS	Planar MAS (Algorithm)
PTAS	Polynomial-Time Approximation Scheme
RN	Regular Node
SC-PMAS	Square-Cover with Rectangular Separation P-MAS (Algorithm)
SCD	Strip Cover with Disks (Algorithm)
SCR	Strip Cover with Rectangles (Algorithm)
STP-MSP	Steiner Tree Problem with Minimum Number of Steiner Points
TPBVP	Two-Point Boundary Value Problem
WSN	Wireless Sensor Networks

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) and Mobile Ad Hoc Networks (MANETs) can operate without any physical infrastructure (e.g. base stations). Moreover, they can operate under a flat architecture, i.e. one in which every node in the network takes the role of host and router. However for several reasons, including the simplification of network computational tasks (e.g. routing, consensus) and energy efficiency, it has been shown that it often desirable to introduce a hierarchical network architecture [5], [9], [10], [11], [25], [31], [34], [37], [40], [42], [46], [47], [54], [62],[63], [66], [75], [79], [82], [83], [85], [94]. In such an architecture, nodes are divided into two categories: Regular Nodes and Backbone Nodes[1]. The Backbone nodes are responsible for the bulk of the network computational tasks, and the regular nodes are therefore freed to perform the arbitrary tasks which they were assigned.

One pertinent example of such hierarchical network architecture is a WSN or MANET with a *virtual backbone* [25],[62]. If all nodes have similar communication capabilities and similar limited energy resources, the virtual backbone may pose several challenges. For example, bottleneck formation along the backbone may affect the available bandwidth and the lifetime of the backbone nodes. In addition, the virtual backbone cannot deal with network partitions resulting from the spatial distribution and mobility of the nodes.

[1]In general, nodes can be divided into an arbitrary number of categories/levels. In the literature Backbone Nodes are commonly referred to as Clusterheads, Base Station Nodes, Dominators, etc.

Alternatively, if some of the nodes are more capable than others, these nodes can be dedicated to providing a backbone over which reliable end-to-end communication can take place. A novel hierarchical approach for a *Mobile Backbone Network* operating in such a way was recently proposed and studied by Rubin et al. (see [79] and references therein) and by Gerla et al. (e.g. [40],[94]). In this thesis, we develop and analyze novel algorithms for the construction and maintenance (under node mobility) of a Mobile Backbone Network. Our general approach is somewhat different from the previous works, since we focus on *controlling the mobility* of the more capable nodes in order to optimize various properties of the communications network. In particular, we focus on connectivity and throughput optimization. However, it should be noted that the construction of a Mobile Backbone Network may improve other aspects of the network performance, including lifetime and Quality of Service as well as network reliability and survivability. Note that a Mobile Backbone Network can be tailored to support the operation of both MANETs and WSNs. For example, in a MANET, Backbone Nodes should be repositioned in response to the mobility of the Regular Nodes. On the other hand, in a static WSN, Backbone Nodes could be positioned near nodes with high requirements or limited energy resources.

We elaborate further regarding our specific problem model and formulation as well as the main contributions of this thesis in the remainder of this section. Additionally, we provide a summary of the relevant related work and an outline for the overall thesis.

1.1 Problem Description and Contributions

A Mobile Backbone Network is composed of two types of nodes. The first type includes static or mobile nodes (e.g. sensors or MANET nodes) with limited capabilities. We refer to these nodes as *Regular Nodes* (RNs). The second type includes mobile nodes with superior communication, mobility, and computation capabilities as well as greater energy resources (e.g. Unmanned-Aerial-Vehicles and Rovers). We refer to them as *Mobile Backbone Nodes* (MBNs). The main purpose of the MBNs is to

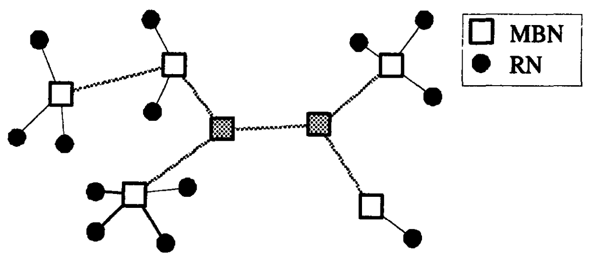

Figure 1-1: A Mobile Backbone Network in which every Regular Node (RN) can directly communicate with at least one Mobile Backbone Node (MBN). All communication is routed through a connected network formed by the MBNs.

provide a mobile infrastructure facilitating network-wide communication. Figure 1-1 illustrates an example of the architecture of a Mobile Backbone Network.

In the first part of the thesis, we focus on the problem of placing the *minimum number* of MBNs such that (i) every RN can directly communicate with at least one MBN, and (ii) the network formed by the MBNs is connected. We assume a *disk* connectivity model, whereby two nodes can communicate if and only if they are within a certain communication range. We also assume that the communication range of the MBNs is significantly larger than the communication range of the RNs. We term this overall problem the *Connected Disk Cover* (CDC) problem.

Our main contribution in this part starts with showing that the CDC problem can be decomposed into the *Geometric Disk Cover* (GDC) problem and the *Steiner Tree Problem with Minimum Number of Steiner Points* (STP-MSP). We prove that if these subproblems are solved separately by γ- and δ-approximation algorithms, the approximation ratio of the joint solution is $\gamma+\delta$. Then, we focus on the two subproblems and present a number of distributed approximation algorithms that maintain a solution to the GDC problem under mobility. A new approach to the solution of the STP-MSP is also described. We show that this approach can be extended in order to obtain a joint approximate solution to the CDC problem. Finally, we evaluate the performance of the algorithms via simulation and show that the proposed GDC algorithms perform very well under mobility and that the new approach for the joint solution can significantly reduce the number of Mobile Backbone Nodes.

An implicit assumption in the formulation of the CDC problem is that an arbitrary number of MBNs are available for deployment (i.e. with the goal being to minimize the number actually deployed). In many scenarios however, a more appropriate (and perhaps realistic) assumption would be that the number of available MBNs is *fixed* a-priori, and the objective is to do the "best we can" with these fixed resources. Note however, that the CDC-type formulation for MBN placement arises very naturally given the assumption of a discrete communications model (e.g. disk model) coupled the requirement for network-wide connectivity. Thus in the second part of the thesis, we attempt to address both of these issues.

Specifically, in the second part of the thesis we consider the joint problem of (i) Placing a fixed number K MBNs in the plane, and (ii) Assigning each RN to exactly one MBN. We formulate and solve two problems under a general communications model (e.g. as compared to a disk model). Specifically, we assume that the "throughput" achieved by an RN transmitting to its assigned MBN is a *decreasing* function of (i) The distance between the RN and MBN, and (ii) The total number of RNs assigned to that MBN. The idea is that the first factor models the loss due to wireless propagation, and the second models loss due to interference caused by multiple RNs trying to access a single MBN. We also assume that under this communications model, MBNs can always communicate with one another. This removes the need to explicitly consider the MBN connectivity issue, and allows us to focus on optimizing RN throughput.

The first problem we consider is the Maximum Fair Placement and Assignment (MFPA) problem in which the objective is to maximize the throughput of the *minimum* throughput RN. The second is the Maximum Throughput Placement and Assignment (MTPA) problem, in which the objective is to maximize the *aggregate* throughput of the RNs. It should be noted that due to the change in model (e.g. fixed number of MBNs, general communications model) from the first part of the thesis, the problems of this part of the thesis require a significantly different approach and solution methodology.

Our main contribution is a novel optimal polynomial time algorithm for the MFPA problem for fixed K. For a restricted version of the MTPA problem, we develop an optimal polynomial time algorithm for $K \leq 2$. We also develop two heuristic algorithms for both problems, including an approximation algorithm for which we bound the worst case performance loss. Finally, we present simulation results comparing the performance of the various algorithms developed.

To this point, we have the solved the Mobile Backbone Construction problem based on "current" location information of the RNs. Specifically, at any given time, the MBNs are placed *reactively* based on RNs' locations at that time. Yet, in many practical scenarios entire RN trajectories are known a-priori (e.g. as waypoints for particular missions). If this is the case, then placing the MBNs by solving an placement problem independently at each time step is, in general, suboptimal. Indeed, it would be desirable to solve for the entire optimal *sequence of placements* for the MBNs at once. In the third part of the thesis, we address this MBN path planning problem both from a discrete and continuous perspective. For our exposition, we consider planning the path of a single MBN given the trajectories of the RNs. Our goal is to maximize the time-average system throughput over the MBN path. For this, we assume an objective function that combines the MFPA throughput objective from the second part of the thesis, along with a penalty/constraint on the speed of the MBN. The reason for this is that it is undesirable to have the MBN moving large distances in response to small RN movements. Additionally, there can be scenarios in which it is undesirable to have large MBN movements even in response to large RN movements, e.g. limited MBN velocity, energy efficiency, MBN location predictability, etc.

Our first contribution involves a discrete formulation of the MBN Path Planning Problem (MPP). We develop a dynamic programming based approximation algorithm for the MPP problem. We provide worst case analysis of the performance of the algorithm. Additionally, we develop an optimal algorithm for the 1-step velocity constrained path planning problem. Using this as a sub-routine, we develop a greedy heuristic algorithm for the overall path planning problem. Next, we approach the path-planning problem from a continuous perspective. We formulate the problem as an optimal control problem, and develop interesting insights into the structure of the optimal solution. Finally, we discuss extensions of the base discrete and continuous formulations and compare the various developed approaches via simulation.

We present extensions to the GDC algorithms developed in Chapter 2 of the thesis in Appendix A. In particular, we develop a number of *distributed* planar-based algorithms for this problem, in contrast to the strip-based algorithms presented in Chapter 2. We analyze the worst case performance of the algorithms using a novel graph-based analysis technique, which we develop. Finally, we present simulation results to evaluate the performance of the algorithms.

1.2 Related Work

The idea of employing hierarchical network architectures for Wireless Networks is a well studied one in the literature [5], [9], [10], [11], [25], [31], [34], [37], [40], [42], [46], [47], [54], [62],[63], [66], [75], [79], [82], [83], [85], [94]. Indeed, several have been proposed for different types of wireless networks. Examples include single-hop clusterings [9],[37], virtual backbones [25],[62] and k-clusterings [5],[31]. However, a common feature of such architectures is that nodes are homogeneous (i.e. have identical capabilities) and that the mobility of the Backbone Nodes is not explicitly controlled. In particular, the network is assumed to be already connected, and the clustering and virtual backbone formation is overlayed on top of the connected network. The work of this thesis significantly differs since we assume a heterogeneous network consisting of Regular Nodes (RNs) and more capable Mobile Backbone Nodes (MBNs). We do not assume the network is connected a-priori, and the goal itself is to place and mobilize the MBNs such that connectivity as well as other network objectives are optimized.

The idea of deliberately controlling the motion of specific nodes in order to maintain some desirable network property (e.g. lifetime or connectivity) has been introduced only recently (e.g. [58],[66], [75]). The Mobile Backbone Architecture that is considered in this thesis was originally presented by Rubin et al. [79] and Gerla et al. [40],[94]. In their work, they assume that the RNs and MBNs are already placed, and a-priori form a connected network. Thus the focus of their work relates to developing system-level protocols for routing, scheduling, MBN election, etc. Our general approach differs in that we focus specifically on the fundamental problem of given a set of arbitrarily located RNs, how to place the MBNs such that various network objectives are optimized.

1.2.1 Work Related to the CDC Problem

The problem of placing and mobilizing MBNs for providing network connectivity is formulated in Chapter 2 as the Connected Disk Cover (CDC) problem. Several problems that are somewhat related have been studied in the past. For simplicity, when describing these problems we will use our terminology (RNs and MBNs). One such problem is the Connected Dominating Set problem [25]. Unlike the CDC problem, in this problem there is no distinction between the communication ranges of RNs and MBNs. Additionally, MBN locations are restricted to RN locations. Similarly, the Connected Facility Location problem [86], [43], also restricts potential MBN locations. Furthermore, this problem implies a cost structure (e.g. the assumption of weights satisfying the triangle inequality) that is not directly adaptable to that of the CDC problem. Finally, The Connected Sensor Cover problem [42] involves placing the minimum number of RNs such that they form a connected network, while still covering (i.e. sensing) a specified *area*. This is significantly different from the objective of the CDC problem, which places MBNs to cover a *discrete set* of RNs, while forming a connected network.

We note that Tang et al. [87] have recently independently formulated and studied the CDC problem (termed in [87] as the Connected Relay Node Single Cover). A centralized 4.5-approximation algorithm for this problem is presented in [87]. In chapter 2, we will show that our approach provides a centralized 3.5-approximation for the CDC problem.

We propose to solve the CDC problem by decomposing it into two NP-Complete subproblems: the Geometric Disk Cover (GDC) problem and the Steiner Tree Problem with Minimum number of Steiner Points (STP-MSP). Hochbaum and Maass [52] provided a Polynomial Time Approximation Scheme (PTAS)[2] for the GDC problem. However, their algorithm is impractical for our purposes, since it is centralized and has a high computational complexity for reasonable approximation ratios. Several other algorithms have been proposed for the GDC problem (see the review in [32]). For example, Gonzalez [38] presented an algorithm based on dividing the plane into strips. In [32] it is indicated that this is an 8-approximation algorithm. We will show that by a simple modification, the approximation ratio is reduced to 6.

Problems related to the GDC problem under node mobility are addressed in [34],[47], and [54],[46]. In [54], a 4-approximate centralized algorithm and a 7-approximate distributed algorithm are presented. Hershberger [47] presents a *centralized* 9-approximation algorithm for a slightly different problem: the mobile geometric *square* cover problem. In this thesis we build upon his approach in order to develop a *distributed* algorithm for the GDC problem.

The algorithm for the STP-MSP proposed in [64] places Relay MBNs along edges of the Minimum Spanning Tree (MST) which connects the Cover MBNs. It has been shown in [20] and [67] that its approximation ratio is 4. In addition, [20] proposed a modified MST-based algorithm that provides an approximation ratio of 3, and a randomized algorithm with approximation ratio 2.5. Finally, [16] studied the general k-connectivity version of the STP-MSP. For $k = 1$ (i.e. the original STP-MSP), the approximation ratios of the algorithms developed in [16] are higher than those in [20] and [64].

Finally, note that there has been a lot of work done on the original Euclidean Steiner Tree (EST) problem and its many network variants [6],[74], [80],[60]. However, the STP-MSP involves solving an EST problem with bounded edge lengths and node weights. Thus the solution methodologies for the STP-MSP differ significantly from those of the EST.

[2]Given a constant $\varepsilon > 0$, a PTAS always finds a solution with value at most $(1 + \varepsilon)$ times the optimal. The running time of a PTAS is polynomial for a *fixed* ε.

1.2.2 Work Related to the MFPA and MTPA Problems

The problem of jointly placing MBNs and assigning each RN to an MBN so as to maximize network throughput is formulated in Chapter 3. The problems are referred to as the (i) Maximum Fair Placement and Assignment (MFPA) and (ii) Maximum Throughput Placement and Assignment (MTPA) problems. To our knowledge, these problems have not been considered before in the literature. With respect to the underlying Mobile Network Architecture, much of the related work to the CDC problem presented in the previous section is also related to this work. However, the MFPA and MTPA problems require solving a joint problem, namely (i) placing the MBNs, and (ii) Assigning RNs to MBNs. Strictly speaking, the CDC problem does not have an explicit assignment component, since an arbitrary number of MBNs can be placed, and one can consider an RN to be assigned to any MBN within its (fixed) communications range. This joint aspect of placement and assignment, as well as some of the other modelling considerations causes the solution approach and methodology for the MFPA and MTPA problems to significantly differ from that of the CDC problem.

Given the more general communications model assumed for the MFPA and MTPA, the closest related work is actually in regards to base station selection/placement for cellular and indoor wireless systems, e.g. [4],[85],[89], [68],[45]. Yet, there are several aspects which differentiate our work from the work in this area. First, the major levers of optimization in our work are both the MBN (e.g. base station) placement *and* the RN to MBN assignments. By contrast, much of the cellular work uses trivial solutions to the assignment problem (e.g. assign each RN to the nearest MBN) and optimize via base station placement/selection and/or power control. Another key difference is that practical considerations for cellular base station placement usually a-priori restricts the set of possible locations to a discrete set of candidates. This restriction typically results in solution methodologies along the lines of simple heuristics, or large scale optimization tools (e.g. Mixed Integer-Linear Programming (MILP), Genetic Algorithms (GA), etc). In contrast, we develop optimal combinatorial algorithms for the joint node placement and assignment problems of this work.

1.2.3 Work Related to the MPP Problem

The problem of determining an optimal path for a single MBN that maximizes the time-average system throughput subject to a velocity constraint/penalty is formulated in Chapter 4, as the MBN Path Planning (MPP) problem. It is formulated from both a discrete and continuous perspective. The discrete version is related to several time-horizon network planning and facility location works considered in the past, e.g. [92],[44], [30],[78]. The work in [44] is especially pertinent since they consider the time-horizon 1-center and 1-median problems on graphs. They show that this problem can be optimally solved in polynomial time. However, a key difference between the MPP problem and the network planning works is that for the MPP problem, the set of potential locations for the MBNs is infinite (i.e. anywhere on the plane). By contrast, the network location work assumes that centers/medians (e.g. MBNs in our context) can only move along edges and vertices of the graph. Moreover, they restrict their objective functions to linear functions of the center/median metrics and movement[3]. We consider general non-linear objective functions as well has hard and soft constraints on the MBN movement, as will be further described in chapter 4.

Along the lines of hard constraints on MBN movement (e.g. velocity) is the work of [12], in which they consider what approximation ratios to the *unconstrained* 1-center/median metrics can be achieved when the MBN to RN velocity is upper bounded. By contrast, we enforce a velocity bound on the MBN, but leave the RN velocity unbounded. Our focus is on characterizing the performance with respect to the *MBN velocity constrained* MPP objective function. Moreover while they consider instantaneous placement problems, our focus is over the entire time horizon.

We formulate a continuous version of the MPP problem as an optimal control problem. The theory of optimal control has been very well studied in the past, e.g. [18], [7]. However, it turns out that the MPP problem maps to a specific class of optimal control problems known as singular control problems. This class of problems are somewhat harder to solve (as compared to regular optimal control problems) and

[3]The 1-center metric is the distance from the MBN to the farthest RN. The 1-median metric is the average distance from the MBN to the RNs.

are not as well studied. Numerical procedures for solving singular control problems have been proposed in the literature [8],[50],[81].

Finally, it should be noted that time horizon network planning and facility location problems have also been formulated in the continuous domain, e.g. [27], [71]. Yet, the heuristic solutions they employ are discrete methods over a discrete time horizon, as opposed to the continuous-time optimal control methods applied in our work.

1.3 Thesis Outline

This Thesis is organized as follows. In Chapter 2 we discuss the problem of placing the minimum number of MBNs to provide network connectivity. In Section 2.2 we formulate the Connected Disk Cover (CDC) problem. In Section 2.3 we present the decomposition approach in which we decompose the CDC problem into the Geometric Disk Cover (GDC) problem and Steiner Tree Problem with Minimum Number of Steiner Points (STP-MSP). Distributed approximation algorithms for placing the Cover MBNs (i.e. for the GDC problem) are presented in Section 2.4. A new approach to placing the Relay MBNs (i.e. for the STP-MSP) is described in Section 2.5. A joint solution to the CDC problem is discussed in Section 2.6. In Section 2.7 we evaluate and compare the performance of the different algorithms via simulation. We summarize the results and discuss future research directions in Section 2.8.

In chapter 3 we describe the joint problem of placing MBNs and assigning RNs to MBNs in order to optimize network throughput. In Sections 3.2 and 3.3 we formulate the MFPA and MTPA problems and give illustrative examples. Section 3.4 presents an optimal solution for the MFPA problem. In section 3.5, we discuss solutions for a restricted version of the MTPA problem. In section 3.6, we present approximation and heuristic algorithms for both problems. Finally, in section 3.7 we evaluate the performance of the algorithms via simulation.

In chapter 4 we describe the problem of computing the MBN path that maximizes the time-average throughput, given that the RN trajectories are known a-priori. In section 4.2 we provide our general discrete problem model and formulation. We next develop a dynamic-programming based approximation algorithm in section 4.4. This is followed our development of a greedy algorithm in section 4.5. We discuss relaxing the hard velocity constraint in the base discrete formulation in section 4.6. Next, in section 4.7 we formulate an MBN path planning problem as a continuous time optimal control problem, and in section 4.8 discuss extensions. Finally, in section 4.9 we present simulation results comparing the various approaches developed in this chapter.

In appendix A we present extensions to the GDC algorithms developed in Chapter 2. In Section A.2 we formulate the problem. The new distributed planar algorithms are presented and analyzed in Sections A.3 and A.4. In Section A.5 we evaluate and the performance of the algorithms via simulation. We summarize the results in Section A.6. Finally, in appendix B we briefly formulate and discuss a satellite broadcast problem, whose solution methodologies were inspired from those used throughout the Thesis.

Chapter 2

Minimizing the Number of Backbone Nodes for Connectivity

2.1 Introduction

As described in chapter 1, a Mobile Backbone Network is composed of two types of nodes. The first type includes static or mobile nodes with limited capabilities. We refer to these nodes as *Regular Nodes* (RNs). The second type includes mobile nodes with superior communication, mobility, and computation capabilities as well as greater energy resources. We refer to them as *Mobile Backbone Nodes* (MBNs). The main purpose of the MBNs is to provide a mobile infrastructure facilitating network-wide communication. In this chapter, we focus on minimizing the number of MBNs needed for connectivity. We develop and analyze novel algorithms that place and mobilize these MBNs in order to maintain network connectivity and to provide a backbone for reliable communication.

Fig. 2-1 illustrates an example of the architecture of a Mobile Backbone Network. The set of MBNs has to be placed such that (i) every RN can directly communicate with at least one MBN, and (ii) the network formed by the MBNs is connected. We assume a *disk* connectivity model, whereby two nodes can communicate if and only if they are within a certain communication range. We also assume that the communication range of the MBNs is significantly larger than the communication

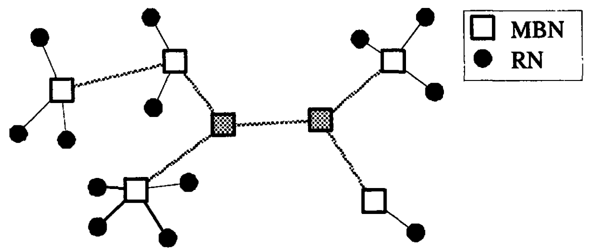

Figure 2-1: A Mobile Backbone Network in which every Regular Node (RN) can directly communicate with at least one Mobile Backbone Node (MBN). All communication is routed through a connected network formed by the MBNs.

range of the RNs.

We term the problem of placing the *minimum* number of MBNs such that both of the above conditions are satisfied as the *Connected Disk Cover (CDC)* problem. While related problems have been studied in the past [20],[25],[47],[52],[86] (see Section 1.2.1 for more details), this chapter is one of the first attempts to deal with the CDC problem.

Our first approach is based on *decomposing* the CDC problem into two subproblems. This approach enables us to develop efficient *distributed algorithms* that have good average performance as well as bounded worst case performance. We view the problem as a two-tiered problem. In the first phase, the minimum number of MBNs such that all RNs are *covered* (i.e. all RNs can communicate with at least one MBN) is placed. We refer to these MBNs as *Cover MBNs* and denote them in Figure 2-1 by white squares. In the second phase, the minimum number of MBNs such that the MBNs' network is connected is placed. We refer to them as *Relay MBNs* and denote them in the figure by gray squares.

In the first phase, the Geometric Disk Cover (GDC) problem [52] has to be solved, while in the second phase, a Steiner Tree Problem with Minimum Number of Steiner Points (STP-MSP) [64] has to be solved. We show that if these subproblem are solved separately by γ and δ approximation algorithms[1], the approximation ratio of the joint solution is $\gamma+\delta$.

[1]A γ-approximation algorithm always finds a solution with value at most γ times the value of the optimal solution.

We then focus on the Geometric Disk Cover (GDC) problem. In the context of static points (i.e. RNs), this problem has been extensively studied in the past (see [2] and references therein). However, much of the previous work is either (i) centralized in nature, (ii) too impractical to implement (in terms of running time), or (iii) has poor average or worst-case performance. Recently, a few attempts to deal with related problems under node mobility have been made [34],[47],[54].

We attempt to develop algorithms that do not fall in any of the above categories. Thus, we develop a number of practically implementable *distributed algorithms* for covering mobile RNs by MBNs. We assume that all nodes can detect their position via GPS or a localization mechanism. This assumption allows us to take advantage of location information in designing distributed algorithms. We obtain the worst case approximation ratios of the developed algorithms and the average case approximation ratios for two of the algorithms. We note that using our analysis methodology, we show that the approximation ratios of algorithms presented in [38] and [47] are lower than the ratios obtained in the past. Finally, we evaluate the performance of the algorithms via simulation and discuss the tradeoffs between the time and communication complexity, and the approximation ratio. We show that on average some of algorithms obtain results that are close to optimal.

Regarding the STP-MSP, [64] and [20] propose 3 and 4-approximation algorithms which are based on finding a Minimum Spanning Tree (MST). However, when applied to the STP-MSP, such *MST-based* algorithms may overlook relatively efficient solutions. We present a *Discretization Approach* that can potentially provide improved solutions. In certain practical instances the approach can yield a 2 approximate solution for the STP-MSP.

We extend the Discretization Approach and show that it can obtain a solution to the *joint CDC* problem in a centralized manner. Even for the CDC problem, using this approach enables obtaining a *2-approximation* for specific instances. Due to the continuous nature of the CDC problem, methods such as integer programming cannot yield an optimal solution. Thus, for specific instances this approach provides the lowest known approximation ratio. It is shown via simulation that this is also the case in practical scenarios.

To summarize, our first main contribution is a decomposition result regarding the CDC problem. Additional major contributions are the development and analysis of distributed algorithms for the GDC problem in a mobile environment, as well as the design of a novel Discretization Approach for the solution of the STP-MSP and the CDC problems.

This rest of this chapter is organized as follows. In Section 2.2 we formulate the problem. In Section 2.3 we present the decomposition approach. Distributed approximation algorithms for placing the Cover MBNs (i.e. for the GDC problem) are presented in Section 2.4. A new approach to placing the Relay MBNs (i.e. for the STP-MSP) is described in Section 2.5. A joint solution to the CDC problem is discussed in Section 2.6. In Section 2.7 we evaluate and compare the performance of the different algorithms via simulation. We summarize the results and discuss future research directions in Section 2.8.

2.2 Problem Formulation

We consider a set of *Regular Nodes* (RNs) distributed in the plane and assume that a set of *Mobile Backbone Nodes* (MBNs) has to be deployed in the plane. We denote by $N = \{1, 2, \ldots, n\}$ the collection of *Regular Nodes*, by $M = \{d_1, d_2, \ldots, d_m\}$ the collection of MBNs, and by d_{ij} the distance between nodes i and j. The locations of the RNs are denoted by the $x-y$ tuples (i_x, i_y) $\forall i$.

We assume that the RNs and MBNs have both a communication channel (e.g. for data) and a low-rate control channel. For the communication channel, we assume the disk connectivity model. Namely, an RN i can communicate bi-directionally with another node j (i.e. an MBN) if the distance between i and j, $d_{ij} \leq r$. We denote by $D = 2r$ the diameter of the disk covered by an MBN communicating with RNs. Regarding the MBNs, we assume that MBN i can communicate with MBN j if $d_{ij} \leq R$ $(R > r)$. For the control channel, we assume that both RNs and MBNs can communicate over a much longer range than their respective data channels. Since given a fixed transmission power, the communication range is inversely related to data rate, this is a valid assumption.

At this stage, we assume that the number of available MBNs is not bounded (e.g. if necessary, MBNs can dispatched from a depot). Yet in our analysis, we will try to minimize the number of MBNs that are actually deployed. Finally, we assume that all nodes can detect their position, either via GPS or by a localization mechanism.We shall refer to the problem of Mobile Backbone Placement as the Connected Disk Cover (CDC) problem and define it as follows.

Problem CDC: Given a set of RNs (N) distributed in the plane, place the smallest set of MBNs (M) such that:

1. For every RN $i \in N$, there exists at least one MBN $j \in M$ such that $d_{ij} \leq r$.
2. The undirected graph $G = (M, E)$ imposed on M (i.e. $\forall k, l \in M$, define an edge $(k, l) \in E$ if $d_{kl} \leq R$) is connected.

The first property is closely related to geometric coverage of the RNs by the MBNs and the second property relates to the connectivity of the MBNs network. We will present solutions both for the case in which the nodes are static and for the case in which the RNs are mobile and some of the MBNs move around in order to maintain a solution the CDC problem. We assume there exists some sort of MBN routing algorithm, which routes specific MBNs from their old locations to their new ones. The actual development of such an algorithm is beyond the scope of this work.

Before proceeding, we introduce additional notation required for the presentation and analysis of the proposed solutions. A few of the proposed algorithms operate by dividing the plane into strips. When discussing such algorithms, we assume that the RNs in a strip are ordered left to right by their x-coordinate and that ties are broken by the RNs' identities (e.g. MAC addresses). Namely, $i < j$, if $i_x < j_x$ or $i_x = j_x$ and the id of i is lower than id of j. We note that in property (1) of the CDC problem it is required that every RN is connected to at least one MBN. We assume that even if an RN can connect to multiple MBNs, it is actually assigned to exactly one MBN. Thus, we denote by P_{d_i} the set of RNs connected to MBN d_i. We denote by d_i^L and

d_i^R the leftmost and rightmost RNs connected to MBN d_i (their x-coordinates will be denoted by $(d_i^L)_x$ and $(d_i^R)_x$). Similar to the assumption regarding the RNs, we assume that the MBNs in a strip are ordered left to right by the x-coordinate of their leftmost RN ($(d_i^L)_x$).

2.3 Decomposition Approach

In this section we obtain an upper bound on the performance of an algorithm that solves the CDC problem by decomposing it and solving each of the two subproblem separately. The first subproblem is the problem of placing the minimum number of *Cover MBNs* such that all the RNs are connected to at least one MBN. In other words, all the RNs have to satisfy only property (1) in the CDC problem definition. This problem is the Geometric Disk Cover (GDC) problem [52] which is formulated as follows:

Problem GDC: Given a set N of RNs (points) distributed in the plane, place the smallest set M of Cover MBNs (disks) such that for every RN $i \in N$, there exists at least one MBN $j \in M$ such that $d_{ij} \leq r$.

The second subproblem deals with a situation in which a set of *Cover MBNs* is given and there is a need to place the minimum number of *Relay MBNs* such that the formed network is connected (i.e. satisfying only property (2) in the CDC problem definition). This subproblem is equivalent to the Steiner Tree Problem with Minimum Number of Steiner Points (STP-MSP) [64] and can be formulated as follows:

Problem STP-MSP: Given a set of Cover MBNs (M_{cover}) distributed in the plane, place the smallest set of Relay MBNs (M_{relay}) such that the undirected graph $G = (M, E)$ imposed on $M = M_{cover} \cup M_{relay}$ (i.e. $\forall k, l \in M$, define an edge (k, l) if $d_{kl} \leq R$) is connected.

We now define a *Decomposition Based CDC Algorithm* and bound the worst case performance of such an algorithm.

Definition 2.3.1. *A* Decomposition Based CDC Algorithm *solves the CDC problem* *by using a γ-approximation algorithm for solving the GDC problem, followed by using a δ-approximation algorithm for solving the STP-MSP.*

Theorem 2.3.1. *For $R \geq 2r$, the Decomposition Based CDC Algorithm is a $(\gamma + \delta)$-approximation algorithm for the CDC problem.*

Proof. Define $ALGO$ as the solution found by solving the CDC problem by the Decomposition Based CDC Algorithm. Also, define $ALGO_{cov}$ and $ALGO_{rel}$ as the set of *Cover* and *Relay* MBNs placed by ALGO. Specifically, an MBN a_i is a *cover* MBN if it covers at least 1 RN (i.e. $P_{a_i} \neq \emptyset$). Otherwise, a_i is a relay MBN. Next, define OPT_{CDC} as the overall optimal solution similarly broken up into OPT_{CDC}^{cov} and OPT_{CDC}^{rel}. Thus we have that,

$$\begin{aligned} |ALGO| &= |ALGO_{cov}| + |ALGO_{rel}| \\ &\leq \gamma \cdot |OPT_{cov}| + \delta \cdot |OPT_{ALGO-cov-rel}| \end{aligned} \tag{2.1}$$

where OPT_{cov} represents the optimal GDC of the RNs, and $OPT_{ALGO-cov-rel}$ represents the optimal STP-MSP solution connecting the Cover MBNs placed by the γ approximate GDC algorithm, $ALGO_{cov}$.

Next, we make use of the fact that a candidate STP-MSP solution given $ALGO_{cov}$ as the input Cover MBNs can be constructed by placing MBNs in the positions defined by those in OPT_{CDC}. The reason this represents a valid STP-MSP solution is that since $ALGO_{cov}$ is a valid GDC for the RNs, it follows that every MBN in $ALGO_{cov}$ is at most a distance r away from *some* RN. Since OPT_{CDC}^{cov} is also a valid GDC, it follows that every MBN in $ALGO_{cov}$ is at most a distance $2r$ from *some* MBN in OPT_{CDC}^{cov}. Therefore, as long as $R \geq 2r$, the MBNs in $ALGO_{cov} \cup OPT_{CDC}$ form a connected network. Finally, since $OPT_{ALGO-cov-rel}$ represents an STP-MSP solution that must be of lower cost than this candidate solution, we have that,

$$\begin{aligned} |ALGO| &\leq \gamma \cdot |OPT_{cov}| + \delta \cdot (|OPT_{CDC}^{cov}| + |OPT_{CDC}^{rel}|) \\ &\leq (\gamma + \delta) \cdot |OPT_{CDC}^{cov}| + \delta \cdot |OPT_{CDC}^{rel}| \\ &\leq (\gamma + \delta) \cdot |OPT_{CDC}| \end{aligned} \tag{2.2}$$

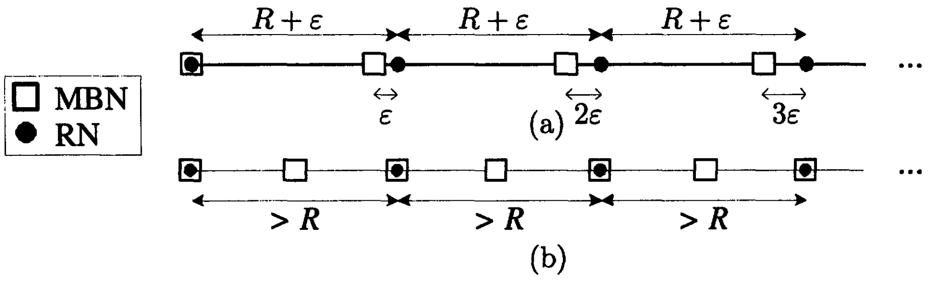

Figure 2-2: Tight example of the approximation ratio of the decomposition approach: (a) optimal solution and (b) decomposition approach solution.

where the second line followed from the fact that the optimal GDC for the RNs is of lower cost than OPT_{CDC}^{cov}. □

According to Theorem 2.3.1, even if the two subproblems (GDC and STP-MSP) are solved optimally (i.e. with $\gamma = \delta = 1$), this yields a 2-approximation to the CDC problem. A tight example of this fact is illustrated in Figure 2-2. Figure 2-2-a shows an n node instance of the CDC problem where ε refers to a sufficiently small constant. Also shown is the optimal solution with cost n MBNs. Figure 2-2-b shows a solution using the decomposition approach (with $\gamma = \delta = 1$), composed of an optimal disk cover and an optimal STP-MSP solution. The cost is $n + n - 1 = 2n - 1$.

We note that if a centralized solution can be tolerated, the approximation ratio of the GDC problem can be very close to 1 (e.g. using a PTAS [52]). The lowest known approximation ratio of the STP-MSP solution is 2.5 [21]. Therefore, by Theorem 2.3.1, the *framework immediately yields a 3.5-approximation algorithm for the solution of the CDC problem.* This improves upon the 4.5-approximation algorithm, recently presented in [87]. It should be noted that since both algorithms make use of a PTAS, their respective complexities are quite high. The key point with respect to our Decomposition Framework is that *any future improvement to the approximation ratio of the STP-MSP will directly reduce the CDC approximation ratio.*

2.4 Placing the Cover MBNs

In this section, we present and analyze distributed algorithms for placing and mobilizing (under RNs mobility) the Cover MBNs.

2.4.1 Strip Cover Algorithms

Hochbaum and Maass [52] introduced a method for approaching the GDC problem by (i) dividing the plane into equal width strips, (ii) solving the problem locally on the points within each strip, and (iii) taking the overall solution as the union of all local solutions. Below we present algorithms that are based on this method. These algorithms are actually two different versions of a single generic algorithm. The first version locally covers the strip with rectangles encapsulated in disks while the second version locally covers the strip directly with disks. We then generalize (to arbitrary strip widths) the effects of solving the problem locally in strips. We use this extension to provide approximation guarantees for the two algorithms in the worst case and in the average case. Finally, we discuss the distributed implementation of the algorithms.

Centralized Algorithms

For simplicity of the presentation, we start by describing the centralized algorithms. The two versions of the Strip Cover algorithm (*Strip Cover with Rectangles* - SCR and *Strip Cover with Disks* - SCD) appear below. In line 6, the first version (SCR) calls the *Rectangles* procedure and the second one (SCD) calls the *Disks* procedure. The input is a set of points (RNs) $N = \{1, 2, \ldots, n\}$ and their (x, y) coordinates, $(i_x, i_y)\,\forall i$. The output includes a set of disks (MBNs) $M = \{d_1, d_2, \ldots, d_m\}$ and their locations such that all points are covered. The first step of the algorithm is to divide the plane into K strips of width $q_{SC} = \alpha D$ (recall that $D = 2r$). The values of q_{SC} that guarantees certain approximation ratios will be derived below. We denote the strips by S_j and let M_{S_j} represent the set of MBNs for strip S_j.

An example of the SCR algorithm and in particular of step 9 in which disks are placed such that they compactly cover all points in the rectangular area with

Algorithm 1 Strip Cover with Rectangles/Disks (SCR/SCD)

```
1: divide the plane into $K$ strips of width $q_{SC} = \alpha D$
2: $M_{S_j} \leftarrow \emptyset, \forall j = 1, \dots, K$
3: for all strips $S_j, j = 1, \dots, K$ do
4:    while there exist uncovered RNs in $S_j$ do
5:       let $i$ be the leftmost uncovered RN in $S_j$
6:       call Rectangles($i$) or call Disks($i$)
7:       $M_{S_j} \leftarrow M_{S_j} \cup d_k$
8: return $\bigcup_j M_{S_j}$
Procedure Rectangles($i$)
9: place an MBN $d_k$ such that it covers all RNs in the rectangular area with $x$-
   coordinates $[i_x, i_x + \sqrt{1-\alpha^2}D]$
10: return $d_k$
Procedure Disks($i$)
11: $P_{d_k} \leftarrow \emptyset$ {set of RNs covered by the current MBN $d_k$}
12: while $P_{d_k} \cup i$ coverable by a single MBN (disk) do
13:    $P_{d_k} \leftarrow P_{d_k} \cup i$
14:    if there are no more RNs in the strip then
15:       break
16:    let $i$ be the next leftmost uncovered RN in $S_j$ not currently in $P_{d_k}$
17: place MBN (disk) $d_k$ such that it covers the RNs $P_{d_k}$
18: return $d_k$
```

x-coordinate range i_x to $i_x + \sqrt{1-\alpha^2}D$ is shown in Figure 2-3.

As mentioned in Section 1.2.1, Gonzalez [38] presented an algorithm for covering points with unit-squares. It is based on dividing the plane into equal width strips and covering the points in each of the strips separately. In [32] it was indicated that when the same algorithm is applied to covering points with unit disks, the approximation ratio is 8. The Strip Cover with Rectangles (SCR) algorithm, described above, is actually a slight modification to the algorithm of [38]. Unlike in [38], in our algorithm we allow the selection of the strip width. This will enable us to prove that the approximation ratio for covering points with unit disks is 6 and to bound the average case approximation ratio by 3.

The Strip Cover with Disks (SCD) algorithm requires to answer the following question (in Step 12): can a set of points $P_{d_k} \cup i$ be covered by a single disk of radius r? This is actually the decision version of the 1-center problem. Many algorithms for solving this problem exist, an example being an $O(n \log n)$ algorithm due to [51]. We

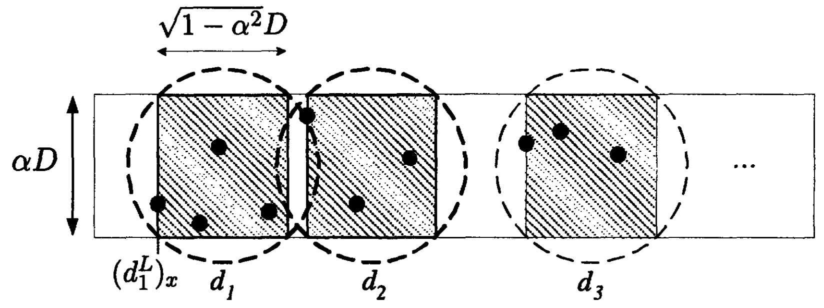

Figure 2-3: An example illustrating step 9 of the SCR algorithm.

will show that solving the *1-center problem* instead of compactly covering rectangles (as done in the SCR algorithm) provides a lower approximation ratio.

The computation complexity of the SCR algorithm is $O(n \log n)$, resulting from sorting the points by ascending x-coordinate. In the SCD algorithm the 1-center subroutine may potentially need to be executed as many as $O(n)$ times for each of the $O(n)$ disks placed. Therefore, the computation complexity is $O(C(n)n^2)$, where $C(n)$ is the running time of the 1-center subroutine used in steps 12 and 17. By using a binary search technique to find the maximal P_{d_k}, we can lower the running time to $O(C(n)n \log n)$.

Approximation Ratios

Let algorithm A denote the local algorithm within a strip, and let $|A_{S_j}|$ denote the cardinality of the GDC solution found by algorithm A covering only the points in strip S_j. Let algorithm B represent the overall algorithm, which works by running algorithm A locally within each strip and taking the union of the local solutions as the overall solution. In our case algorithm B is either the SCR or the SCD algorithm and algorithm A is composed of the steps 4-7 within the for loop.

Let $|OPT|$ represent the cardinality of an optimal solution of the GDC problem in the plane and $|OPT_{S_j}|$ the cardinality of an optimal solution for points exclusively within strip S_j. Note that $OPT \neq \bigcup_{S_j} OPT_{S_j}$, since OPT can utilize disks covering points across multiple strips. Finally, let Z_A denote the worst case approximation ratio of algorithm A. Namely, Z_A is the maximum of $|A_{S_j}|/|OPT_{S_j}|$ over all possible point-set configurations in a strip S_j. Similarly, let Z_B denote the worst case approximation ratio of algorithm B.

We characterize Z_B as a function of Z_A. Namely, if $q \leq D$, the cardinality of the solution found by algorithm B is at most $(\lceil \frac{D}{q} \rceil + 1)Z_A$ times that of the optimal solution, $|OPT|$.

Observation 2.4.1. *If the strip width is $q \leq D$, a single disk can cover points from at most $(\lceil \frac{D}{q} \rceil + 1)$ strips.*

Lemma 2.4.1. *If the strip width is $q \leq D$, $Z_B = (\lceil \frac{D}{q} \rceil + 1)Z_A$.*

Proof. Consider the set of disks in an optimal solution to the GDC problem in the plane, $OPT = d_1, \ldots, d_{|OPT|}$. From OPT, we can create an "algorithm B type" solution (i.e. made up of disks covering points only from single strips) in the following way. Assume OPT disk d_k covers points from c_k different strips (e.g. $S_j, S_{j+1}, \ldots, S_{j+c_k-1}$). For each such d_k, create c_k new disks $d'_1, d'_2, \ldots, d'_{c_k}$ and assign to each d'_j the points covered by d_k that lie exclusively within strip S_j.

Upon doing this for all $d_k \in OPT$, let OPT' denote the resulting set of disks. Clearly, OPT' can be expressed as $\bigcup_{S_j} OPT'_{S_j}$, where OPT'_{S_j} represents the subset of disks in OPT' that cover points exclusively within strip S_j. Therefore, we have that,

$$|OPT'| = \sum_{S_j} |OPT'_{S_j}| = \sum_{k=1}^{|OPT|} c_k \leq \left(\left\lceil \frac{D}{q} \right\rceil + 1 \right) \cdot |OPT| \tag{2.3}$$

where the second equality, i.e. converting a sum over strips into a sum over disks, follows from the construction of OPT', and the inequality follows from Observation 2.4.1.

Next, we note that by definition $|A_{S_j}| \leq Z_A \cdot |OPT_{S_j}|$. Combining this with the fact that $OPT_{S_j} \leq OPT'_{S_j}$ for all strips S_j, we have that,

$$\begin{aligned} |B| &= \sum_{S_j} |A_{S_j}| \leq Z_A \cdot \sum_{S_j} |OPT_{S_j}| \leq Z_A \cdot |OPT'| \\ &\leq Z_A \cdot \left(\left\lceil \frac{D}{q} \right\rceil + 1 \right) \cdot |OPT| \end{aligned} \tag{2.4}$$

where the last inequality followed from (2.3). □

We now show that in the SCR algorithm, $Z_A = 2$. This approximation ratio is tight, as illustrated in Figure 2-4-a.

Lemma 2.4.2. *If the strip width $q_{SC} \leq \frac{\sqrt{3}D}{2}$, steps 4-7 of the SCR algorithm provide a 2-approximation algorithm for the GDC problem within a strip.*

Proof. Consider some strip S. Let $OPT_S = \{d_1, d_2, \ldots, d_{|OPT_S|}\}$ and $ALGO_S = \{a_1, a_2, \ldots, a_{|ALGO_S|}\}$ denote an optimal in-strip solution and SCR in-strip subroutine (steps 4-7) solution, respectively. Recall that we assume that the MBNs of both OPT_S and $ALGO_S$ are ordered from left to right by x-coordinate of the leftmost covered point (i.e. $i < j$ if $(d_i^L)_x \leq (d_j^L)_x$). Finally, define a_{b_m} as the b_m^{th} algorithm disk (from the left) corresponding to the disk that covers the rightmost point covered by the m^{th} OPT_S disk d_m.

Let $q_{SC} = \alpha D, \alpha < 1$. We now prove by induction that if $\alpha \leq \sqrt{3}/2$, the in-strip subroutine has approximation ratio of 2, i.e. $|ALGO_S| = b_{|OPT_S|} \leq 2|OPT_S|$.

Base Case: The *area* covered by d_1 (the leftmost optimal disk) is bounded by a rectangle with x-coordinate range $(d_1^L)_x$ (the x-coordinate of the leftmost point) to $(d_1^L)_x + D$. The *minimum* area covered by two SCR algorithm disks whose leftmost point is $(d_1^L)_x$ is a rectangle with x-coordinate range $(d_1^L)_x$ to $(d_1^L)_x + 2\sqrt{1-\alpha^2}D$. Therefore, as long as $2\sqrt{1-\alpha^2}D \geq D$, it is the case that $b_1 \leq 2$. This condition is met if $q_{SC} \leq \sqrt{3}D/2$.

Inductive Step: Assume that the in-strip algorithm uses no more than $2m$ disks to cover all the points covered by $d_1, \ldots, d_m$ (i.e. $b_m \leq 2m$). Now consider the number of additional disks it takes for the algorithm to cover the points covered by $d_1, \ldots, d_m, d_{m+1}$. Clearly, since all of the points up to the rightmost point of d_m are already covered, by the same argument as the base case, the algorithm will use at most 2 extra disks to cover the points covered by d_{m+1}. It thus follows that if $q \leq \sqrt{3}D/2$, $b_{m+1} \leq b_m + 2 \leq 2m + 2 = 2(m+1)$. □

By combining the results of lemmas 2.4.1 and 2.4.2, we obtain the approximation ratio of the SCR algorithm.

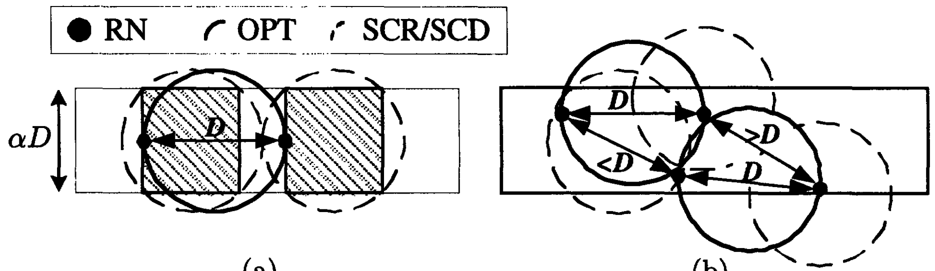

Figure 2-4: Tight examples of the 2 and 1.5 approximation ratios obtained by the in-strip subroutines of the (a) SCR and (b) SCD algorithms.

Theorem 2.4.1. *If $\frac{D}{2} \leq q_{SC} \leq \frac{\sqrt{3}D}{2}$, the SCR algorithm is a 6-approximation algorithm for the GDC problem.*

Proof. Define algorithm A as the in-strip subroutine of the SCR algorithm (steps 4-7) and algorithm B as the SCR algorithm. From Lemma 2.4.2, for $q \leq \sqrt{3}D/2$, $Z_A = 2$. From Lemma 2.4.1, $Z_B \leq Z_A(\lceil D/q \rceil + 1)$, the minimum value of which (for $q < D$) is $3Z_A$. This is attained when $q \geq D/2$. □

Below it is shown that in the SCD algorithm $Z_A = 1.5$. Combining this result with Lemma 2.4.1, we obtain the approximation ratio of the SCD algorithm. Notice that the approximation ratio of 1.5 for the in-strip subroutine of the SCD algorithm is tight, as illustrated in Figure 2-4-b.

Lemma 2.4.3. *If $q_{SC} \leq \frac{\sqrt{5}D}{3}$, steps 4-7 of the SCD algorithm provide a 1.5-approximation algorithm for the GDC problem within a strip.*

Proof. Similar to the proof of Lemma 2.4.2, we use induction to prove the result. We utilize the same definitions as from that proof.

Base Case: There are 2 "sub" base-cases to consider. First, assume $(d_1^R)_x < (d_2^L)_x$, as shown in Fig. 2-5-a. If this this is the case, it is easy to see that $b_1 = 1$, as by definition all of the points from d_1^L to d_1^R are coverable by a single disk; this fact would have been exploited in step 12 of the SC algorithm. Second, assume $(d_1^R)_x \geq (d_2^L)_x$, as shown in Fig. 2-5-b. In this scenario, we consider a base case of $m = 2$, and show that $b_2 \leq 3$. To see this, first note that all points immediately left of d_2^L could

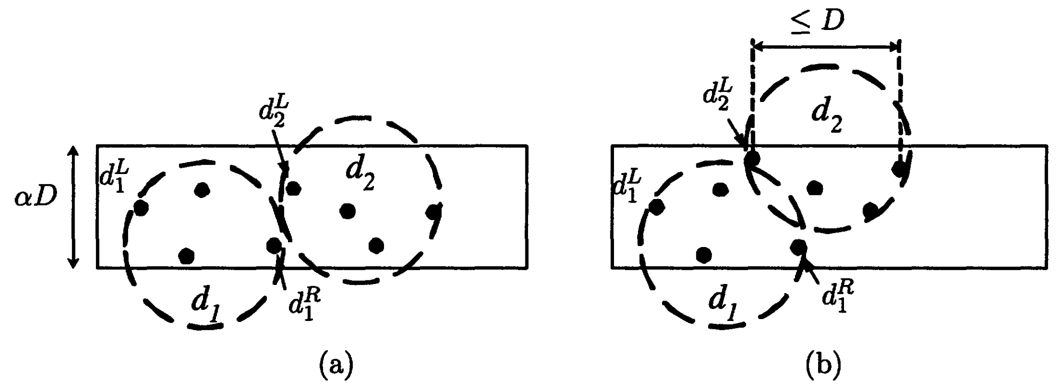

Figure 2-5: Illustration of the SCD induction proof. (a) "Sub" Base-Case 1: $(d_1^R)_x < (d_2^L)_x$. (b) "Sub" Base-Case 2: $(d_1^R)_x \geq (d_2^L)_x$

have and therefore would have been covered by a single SCD disk as per line 17 of the algorithm. Next, as shown in Fig. 2-5-b, we note that the remaining uncovered points all lie within a rectangular area of at most D along the strip. Since a lower bound on the *area* each SCD disk must cover is $\sqrt{1-\alpha^2}D$ along the strip (i.e. this area is always compactly coverable), we have that these points will be covered by at most 2 disks as long as $2\sqrt{1-\alpha^2}D \geq D$. This condition is met if $\alpha \leq \frac{\sqrt{3}}{2}$.

Inductive Step: Assume the in-strip algorithm uses no more than $\frac{3}{2}m$ disks to cover all the points covered by $d_1, \ldots, d_m$, i.e. $b_m \leq \frac{3}{2}m$. Assume also that m is even[2]. Now consider the number of additional disks it takes for the algorithm to cover the points covered by $d_1, \ldots, d_m, d_{m+1}$. Define d^{R*} as the rightmost point covered by $d_1, \ldots, d_m, d_{m+1}$, e.g. $d^{R*} = max_{1 \leq l \leq m+1}[d_l^R]$.

Again we have two cases: First, assume that $(d^{R*})_x < (d_{m+2}^L)_x$. This case is identical to the first base-case in that the algorithm uses exactly one extra disk to cover the points from d_{m+1}^L to d_{m+1}^R, i.e.,

$$b_{m+1} = b_m + 1 \leq \frac{3}{2}m + 1 \leq \frac{3}{2}(m+1). \tag{2.5}$$

The second case assumes $(d^{R*})_x \geq (d_{m+2}^L)_x$. This case is identical to the second base case, whereby we can conclude that the algorithm uses at most 3 extra disks in

[2]The second base-case and second inductive-case ensure the lemma is true for all m.

order to cover the points covered by $d_1, \ldots, d_{m+2}$, i.e.,

$$b_{m+2} \leq b_m + 3 = \frac{3}{2}m + 3 = \frac{3}{2}(m+2). \tag{2.6}$$

□

Theorem 2.4.2. *If $\frac{D}{2} \leq q_{SC} \leq \frac{\sqrt{5}D}{3}$, the SCD algorithm is a 4.5-approximation algorithm for the GDC problem.*

Proof. Exactly the same as the proof of Theorem 2.4.1, except that as per Lemma 2.4.3, we use $Z_A = 1.5$ instead of 2. □

Up to now we have discussed the *worst case* performance. We now wish to bound the approximation ratios of the SCR and the SCD algorithms in the *average case.* We assume that the RNs are randomly distributed according to a two dimensional Poisson process[3]. Due to the random locations of the RNs, $|OPT|$ is a random variable. Similarly, we define $|SCR|$ and $|SCD|$ as random variables corresponding to the number of disks placed by the SCR and the SCD algorithms. We define the *average approximation ratios* β_{SCR} and β_{SCD} as,

$$\beta_{SCR} = \frac{E[|SCR|]}{E[|OPT|]}, \; \beta_{SCD} = \frac{E[|SCD|]}{E[|OPT|]}. \tag{2.7}$$

It should be noted that β_{SCR} differs from the expected value of the approximation ratio $(E[|SCR|/|OPT|])$. Yet, it provides a good measure of the average performance.

The following Theorem and Corollary bound the average approximation ratio of the SCR algorithm, thereby bounding the ratio of the SCD algorithm (since SCD always outperforms SCR). It can be seen that although the worst case approximation ratios are 6 and 4.5 (respectively), selecting a specific strip width results in an average approximation ratio which is bounded by 3. In Section 2.7 we will show by simulation that in practice the approximation ratios are actually much lower.

[3]When the number of RNs is given, their positions are independent and each is *uniformly distributed* in the plane.

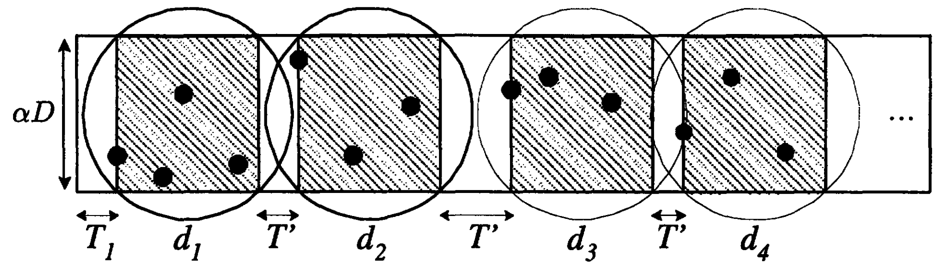

Figure 2-6: Probabilistic analysis of the performance of the SCR algorithm within a strip.

Theorem 2.4.3. *Given RNs distributed in the plane according to a two dimensional Poisson process with density λ,*

$$\beta_{SCD} \leq \beta_{SCR} \leq \frac{D^2\lambda + 2D\sqrt{\lambda} + 1}{\alpha\sqrt{1-\alpha^2}D^2\lambda + 1}. \tag{2.8}$$

Proof. To prove the theorem, we start by upper bounding $E[|SCR|]$. To this end, consider a single strip S and recall that the SCR algorithm iteratively places disks by identifying the leftmost uncovered point i and fully covering the x-range between i_x and $i_x + \sqrt{1-\alpha^2}D$ along the strip. The points are distributed in the plane according to a two dimensional Poisson process with density λ. Therefore, the horizontal (x-coordinate) distance between points is exponentially distributed with average $\frac{1}{\lambda\alpha D}$. Thus, the expected distance to the location of the first disk is $E[T_1] = \frac{1}{\lambda\alpha D}$ (see Figure 2-6.). Furthermore, once a disk is placed, the expected distance between the end of its coverage and the start of the next disk is $E[T']$. Due to the memoryless property of the exponential random variable, we can conclude $E[T'] = \frac{1}{\lambda\alpha D}$.

It therefore follows that the expected number of disks used by the SCR algorithm within a strip is the total length of the strip (less the initial space) divided by the expected distance between the start of one disk and the start of another. Namely,

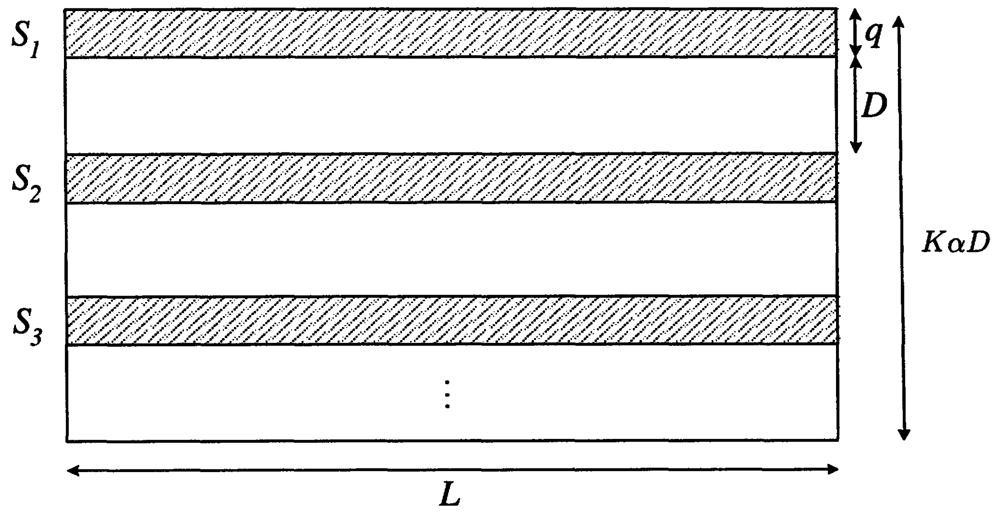

Figure 2-7: Dividing the plane into strips in order to lower bound $E[|OPT|]$

$$\begin{aligned}
E[|SCR|_S] &= \frac{L - \frac{1}{\lambda\alpha D}}{\sqrt{1-\alpha^2}D + \frac{1}{\lambda\alpha D}} \\
&\approx \frac{L}{\sqrt{1-\alpha^2}D + \frac{1}{\lambda\alpha D}} \\
&= \frac{\lambda\alpha D L}{\lambda\alpha\sqrt{1-\alpha^2}D^2 + 1}
\end{aligned} \tag{2.9}$$

where $E[|SCR|_S]$ is the expected number of disks used by the SCR algorithm within strip S. Note that in the second line of (2.9) we assume that $L >> \frac{1}{\lambda\alpha D}$; we technically don't need this assumption, but it makes the analysis cleaner. The expected total number of disks used by the algorithm over the entire plane is therefore this number multiplied by the total number of strips in the plane, i.e.,

$$E[|SCR|] = \frac{\lambda\alpha D L K}{\lambda\alpha\sqrt{1-\alpha^2}D^2 + 1}. \tag{2.10}$$

We next aim to lower bound $E[|OPT|]$. To this end, we divide the plane into D-spaced horizontal strips of width q as shown in Figure 2-7.

We can lower bound the expected number of disks used to cover points in a single strip S by an optimal algorithm by noting that the area coverable by each OPT disk is no more than a rectangle of size $q \times D$. Thus, using a similar argument to when we upper bounded the number of SCR disks required to cover a strip, we have that,

$$E[|OPT|_S] \geq \frac{L}{D + \frac{1}{\lambda q}} \tag{2.11}$$

where $E[|OPT|_S]$ is the expected number of disks used by the optimal solution within strip S. Next we note that an upper bound on the expected number of OPT disks used to cover points in the whole plane can be achieved by summing over the disks used to cover each of the individual strips. The reason we can do this is that since there is a distance D between strips, it is impossible for a single OPT disk to simultaneously cover points from two different strips. We therefore have that,

$$\begin{aligned} E[|OPT|] &\geq \left(\frac{L}{D + \frac{1}{\lambda q}}\right) \cdot \left(\frac{K\alpha D}{D + q}\right) \\ &= \frac{KL\alpha D}{D^2 + \frac{1}{\lambda} + \left(Dq + \frac{D}{\lambda q}\right)}. \end{aligned} \tag{2.12}$$

Next, since we have control over the strip size q, and want to find the tightest possible lower bound, we can select q so as to maximize $E[|OPT|]$, i.e. minimize the bracketed quantity in the denominator of (2.12). It turns out that setting $q = \sqrt{\frac{1}{\lambda}}$ achieves this. Substituting this into (2.12), we have that,

$$E[|OPT|] \geq \frac{KL\alpha D}{D^2 + \frac{1}{\lambda} + \frac{2D}{\sqrt{\lambda}}}. \tag{2.13}$$

Finally, combining (2.13), (2.10) and (2.7) gives us our desired upper bound on β_{SCR}, i.e.,

$$\begin{aligned} \beta_{SCR} &\leq \left(\frac{\lambda\alpha DLK}{\lambda\alpha\sqrt{1-\alpha^2}D^2 + 1}\right) \cdot \left(\frac{\lambda D^2 + 2\sqrt{\lambda}D + 1}{\lambda\alpha DLK}\right) \\ &= \frac{D^2\lambda + 2D\sqrt{\lambda} + 1}{\alpha\sqrt{1-\alpha^2}D^2\lambda + 1}. \end{aligned} \tag{2.14}$$

□

Corollary 2.4.1. *If $q_{SC} = \frac{D}{\sqrt{2}}$, then $\beta_{SCD} \leq \beta_{SCR} \leq 3$.*

Proof. We derive the maximum value of (2.14) by differentiating with respect to λ. Upon doing so and plugging this value of λ into (2.14) gives us,

$$\beta_{SCR} \,|_{\lambda=\lambda_{max}} \leq \frac{\alpha\sqrt{1-\alpha^2}+1}{\alpha\sqrt{1-\alpha^2}} \tag{2.15}$$

which is interestingly independent of D. Finally, we note that for $\frac{1}{2} \leq \alpha < 1$, (2.15) is minimized when $\alpha = \frac{1}{\sqrt{2}}$, at which point it achieves a value of exactly 3. □

Distributed Implementation

By construction, the SCR and SCD algorithms can be easily implemented in a distributed manner. The algorithms are executed at the RNs and operate within the strips. Thus, we assume that the strips are fixed and that their boundaries are known to all nodes. The SCR algorithm executed at an RN i, consisting of rules regarding initial construction and maintenance under RN mobility is described below. RN mobility affects the design of the algorithms, since it can cause an RN to disconnect from its MBN or to move to a neighboring strip in which it is not covered by an MBN. Recall that we denote the RNs within a strip according to their order from the left (i.e. $i < j$ if $i_x \leq j_x$). Ties are broken by node ID.

It can be seen that every RN that has no left neighbors within distance D initiates the disk placement procedure that propagates along the strip. The propagation stops once there is a gap between nodes of at least D. If an RN arrives from a neighboring strip or leaves the MBN's coverage area, it initiates the disk placement procedure that may trigger an update of the MBN's locations within the strip. Notice that MBNs only move when a recalculation is required. Although the responsibility to place and move MBNs is with the RNs, simple enhancements would allow the MBNs to re-place themselves during the maintenance phase.

The computation complexity is $O(1)$ to determine what message to send out (if any). The communication complexity is potentially $O(n)$, since *MBN Placed* messages may potentially have to propagate the entire length of the strip. Information has to

Algorithm 2 Distributed SCR (at RN i)

```
Initialization
 1: let $G_i$ be the set of RNs $j$ such that $j < i$ and $i_x - j_x \leq D$
 2: if $G_i = \emptyset$ then
 3:   call Place MBN
Construction and Maintenance
 4: if *MBN Placed* message received then
 5:   call Place MBN
 6: if $i$ is disconnected from its MBN or enters from a neighboring strip then
 7:   if there is at least one MBN within distance $r$ then
 8:     join one of these MBNs
 9:   else
10:     call Place MBN
Procedure Place MBN
11: let $i^R$ be the rightmost RN s.t. $(i^R)_x \leq i_x + \sqrt{1-\alpha^2}D$
12: place MBN $d_k$ covering RNs $j$, where $j_x \in [i_x, (i^R)_x]$
13: if $(i^R+1)_x - (i^R)_x \leq D$ then
14:   send an *MBN Placed* message to $i^R + 1$
```

be transmitted between RNs over a distance $D = 2r$. Recall that in Section 2.2 we assumed that there is a long range control channel. Therefore, once RNs decide to place an MBN, we assume that there is a way to communicate this to one of the MBNs.

The distributed SCD algorithm is similar to the distributed SCR algorithm. The main difference is that in Step 11 of *Place MBN*, i^R is defined as the rightmost coverable point (by a single disk of radius r), given that i is the leftmost point. As mentioned earlier, finding this point requires solving 1-center problems. Then, in Step 12 a disk that covers all the points between i and i^R should be placed. The computation complexity of the distributed SCD algorithm is a periodic $O(C(n)\log n)$ to calculate the value of i^R, where $C(n)$ is the running time of the 1-center subroutine used. The communication complexity is $O(n)$.

2.4.2 MObile Area Cover (MOAC) Algorithm

In the SCR and SCD algorithms, an RN movement may change the allocation of RNs to MBNs along the whole strip. Thus, although they may operate well in a relatively static environment, it is desirable to develop algorithms that are more tailored to frequent node movements. In this section we present such an algorithm which builds upon ideas presented in [47]. As mentioned in Section 1.2.1, Hershberger [47] studied the problem of covering moving points (e.g. RNs) with mobile unit-squares (e.g. MBNs). Since the d-dimensional *smooth maintenance scheme* proposed in [47] does not easily lend itself to distributed implementation, we focus on the *simple 1-D algorithm* proposed there.

Applied to our context, the Simple 1-D algorithm covers mobile RNs along the strip with length D rectangles (MBNs). The key feature is that point transfers between MBNs are *localized.* Namely, changes do not propagate along the strip. According to [47], the algorithm has a worst case performance ratio of 3.[4]

Extending the *Simple 1-D algorithm* of [47] to diameter D disks is not straightforward. We will first show that an attempt to simply use rectangles encapsulated in disks without any additional modifications results in a 4-approximation to the GDC problem within a strip. Then, we will present the MObile Area Cover (MOAC) algorithm which reduces the approximation ratio to 3.

We define the strip width as $q_{MOAC} = \alpha D$. We reduce disks to the rectangles encapsulated in them and use these rectangles to cover points within the strip, as was depicted in Figure 2-3. The rectangles cover the strip width (αD) and their length is *at most* $\sqrt{1-\alpha^2}D$. We set $D = 1$ and $\alpha = \sqrt{5}/3$ (resulting in $\sqrt{1-\alpha^2}D = 2/3$). These are arbitrary values selected for the ease of presentation. Yet, the algorithm and the analysis are applicable to any $1/2 \leq \alpha \leq \sqrt{5}/3$. We restate the set of rules from [47] using our terminology and assuming (unlike [47]) that the rectangles' lengths are at most 2/3.

The following lemma provides the performance guarantee of this algorithm. Notice that since the changes are kept local, the approximation ratio holds at all time (i.e. there is no need to wait until the changes propagate).

[4]We note that using the same inductive proof methodology, used for Lemma 2.4.2, one can show that the simple 1-D algorithm actually maintains a 2-approximation at all times.

Algorithm 3 Simple 1-D [13]

```
0 initialize the cover greedily {using the SCR algorithm}
1 maintain the leftmost RN and rightmost RN of each MBN rectangle
2 if two adjacent MBN rectangles come into contact then
    exchange their outermost RNs
3 If a set of RNs covered by an MBN becomes too long {the separation between its
  leftmost and rightmost RNs becomes greater than 2/3} then
    split off its rightmost RN into a singleton MBN
    check whether rule 4 applies
4 if two adjacent MBN rectangles fit in a 2/3 rectangle then
    merge the two MBNs
```

Lemma 2.4.4. *The* Simple 1-D algorithm [47] *with* $\sqrt{1-\alpha^2} = 2/3$ *is* at all times *a 4-approximation algorithm for the GDC problem within a strip.*

Proof. To begin, we assume the same definitions of OPT_S, $ALGO_S$, and b_m from Lemma 2.4.2. We now proceed to prove the theorem by induction.

Base Case: The length (along the strip) covered by d_1 (the leftmost optimal disk) is at most 1 (recall that we pre-set $D = 1$ for this section). Next, we show by a packing argument that at most 4 $ALGO_S$ disks can simultaneously cover points from such a unit-length interval where by assumption, no uncovered points exist to the left of d_1^L. To see why, assume 5 such $ALGO_S$ disks existed. However, this would mean that the member points of four of the disks all lay within a unit interval. We define the "combined length" of adjacent $ALGO_S$ disks m_j and m_{j+1} as $Q_j = (d_{j+1}^R)_x - (d_j^L)_x$. We thus have that $Q_1+Q_3 \leq 1$. However, from rule 4 of the Simple 1-D algorithm two adjacent disks m_j and m_{j+1} are merged if their combined length $Q_j \leq \frac{2}{3}$. Therefore, assuming $Q_1 > \frac{2}{3}$ and $Q_3 > \frac{2}{3}$, we have that $Q_1+Q_3 > \frac{4}{3} > 1$, which is a contradiction.

Inductive Step: Assume the Simple 1-D algorithm uses no more than $4m$ disks to cover all the points covered by $d_1, \ldots, d_m$, i.e. $b_m \leq 4m$. Now consider the number of additional disks it takes for the algorithm to cover the points covered by $d_1, \ldots, d_m, d_{m+1}$. Since all of the points up to the rightmost point of d_m are already covered, by the same argument as the base case the algorithm will use at most 4 extra

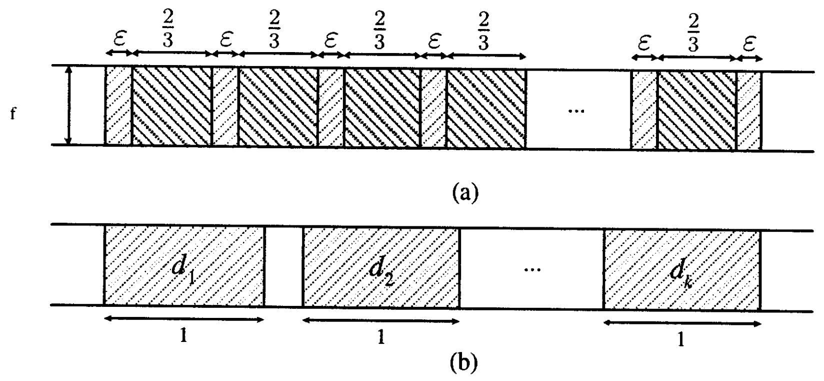

Figure 2-8: Worst case example for the performance of the Simple 1-D algorithm: (a) algorithmic solution and (b) optimal solution. The number of optimal MBNs is denoted by k.

disks to cover the points covered by d_{m+1}. Thus, we have that,

$$b_{m+1} \leq b_m + 4 \leq 4m + 4 = 4(m+1). \tag{2.16}$$

□

From lemmas 2.4.1 and 2.4.4 it follows that if implemented simultaneously in every strip, the algorithm provides a 12-approximation for the GDC problem in the plane, which is relatively high. We now focus on enhancements that reduce the approximation ratio while maintaining the desired locality property.

Figure 2-8 presents an example which shows that the approximation ratio described in Lemma 2.4.4 is tight. It can be seen that the performance ratio is $(4k-1)/k$, where k is the number of disks used by the optimal solution. One of the sources of inefficiency is the potential presence of ε-length MBNs (e.g. covering a single RN) that cannot merge with their 2/3-length neighbor MBNs. Thus, up to 5 MBNs deployed by the Simple 1-D algorithm may cover points which are covered by a single optimal MBN. As long as such narrow MBNs can be avoided, a better approximation can be achieved. We now modify the Simple 1-D algorithm to yield the MOAC algorithm in which ε-length MBNs cannot exist.

Before describing the algorithm, we make the following definitions. For MBN d_i, in addition to its leftmost and rightmost RNs, defined earlier, as d_i^L and d_i^R, we also define L_i and R_i as the x-coordinates of its left and right *domain* boundaries. The interpretation of MBN d_i's domain is that any point in the x-range of $[L_i, R_i]$ will automatically become a member point of MBN d_i. Recall that by definition MBN d_i is to the left of MBN d_j if $(d_i^L)_x < (d_j^L)_x$.

The MOAC algorithm operates within strips and maintains the following *invariants* in each strip (in order of priority) at all times, for every MBN d_i:

1. *Domain definition*: $L_i \leq (d_i^L)_x \leq (d_i^R)_x \leq R_i$.

2. *Domain length*: $\frac{1}{3} \leq |R_i - L_i| \leq \frac{2}{3}$.

3. *Domain disjointness*: $[L_i, R_i] \bigcap [L_j, R_j] = \emptyset, \forall d_j \in M$.

4. *Domain influence*: $\forall p \in N$, $L_i \leq p_x \leq R_i \leftrightarrow p_x \in P_{d_i}$.

We describe the MOAC algorithm below. It consists of rules regarding construction and maintenance of the MBN cover. This algorithm can be implemented in distributed manner by applying some of the rules at the MBNs and some of them at disconnected (i.e. uncovered) RNs (it is clear from the context where each rule should be applied). For brevity, we only state the maintenance rules for the case in which an RN moves outside its MBN's domain boundary to the *right* (analogous rules apply to a leftward movement).

It should be noted that the operations in lines 22-26 can always be accomplished without violating invariant (2). This is due to the fact that an MBN d_j is created for point p only if $|p_x - L_{j-1}| > 2/3$ (otherwise MBN d_{j-1} would have been stretched), which implies there is enough space for two MBNs of size greater or equal to $1/3$ to coexist. Following the merge in line 28, the MBN should update its L_i and R_i such that the domain will include all RNs and will satisfy invariant (2). This is always possible, since the two merged MBNs satisfy the invariants prior to their merger.

The following lemma provides the performance guarantee of the MOAC algorithm within the strip. From Lemma 2.4.1 it follows that if MOAC is simultaneously executed in all strips, it is a 9-approximation algorithm.

Algorithm 4 MObile Area Cover (MOAC)

```
Initialization
1: cover the RNs with MBNs using the SCR algorithm
2: for all MBNs $i$ do
3:     $L_i \leftarrow d_i^L$ ; $R_i \leftarrow d_i^L + \frac{2}{3}$
4:     $P_{d_i} \leftarrow$ all RNs within $[L_i, R_i]$
Maintenance (analogous rules apply for leftward movement)
5: if an RN $p \in P_{d_i}$ moves right such that $p_x > R_i$ then
6:     if $L_j \leq p_x \leq R_j$, $j \neq i$ {$p$ in $d_j$'s domain} then
7:         remove $p$ from $P_{d_i}$
8:     else if $|p_x - L_i| \leq \frac{2}{3}$ then
9:         stretch $L_i$ and $R_i$ to maintain invariant (1) by setting $R_i \leftarrow p_x$ and $L_i \leftarrow$
           $max(L_i, p_x - \frac{2}{3})$
10:    else {$p$ not in the immediate domain of any MBN}
11:        remove $p$ from $P_{d_i}$
Disconnection
12: if at any time there exists an uncovered RN $p$ then
13:     if for some MBN $d_j$, $L_j \leq p_x \leq R_j$ then
14:         $P_{d_j} \leftarrow p$
15:     else if for some MBN $d_j$, $L_j$ and $R_j$ can be stretched (see line 9) to include $p$
        while maintaining invariant (2) then
16:         $P_{d_j} \leftarrow p$
17:         strech $L_j$ and $R_j$ to maintain invariants (1),(2)
18:     else {$p$ cannot be covered by an existing MBN}
19:         let $d_{j-1}$ and $d_{j+1}$ represent the MBNs to the left and right of $p$
20:         if $|L_{j+1} - R_{j-1}| \geq \frac{1}{3}$ {i.e. enough "open space" to maintain invariant (2)}
            then
21:             create MBN $d_j$ with $P_{d_j} = p$ and $|R_j - L_j| \geq \frac{1}{3}$ while maintaining invariant
                (3)
22:         else {$< \frac{1}{3}$ space around $p$}
23:             shrink MBN $d_{j-1}$ such that $R_{j-1} = p_x - \frac{1}{3}$
24:             create MBN $d_j$ with $L_j = p_x - \frac{1}{3}$ and $R_j = p_x$
25:             $P_{d_{j-1}} \leftarrow$ all points in $[L_{j-1}, R_{j-1}]$
26:             $P_{d_j} \leftarrow$ all points in $[L_j, R_j]$
Merge
27: if there exists MBN $d_j$ such that $|(d_j^R)_x - (d_i^L)_x| \leq \frac{2}{3}$ or $|(d_i^R)_x - (d_j^L)_x| \leq \frac{2}{3}$ then
28:     merge $d_j$ into $d_i$
```

Lemma 2.4.5. *The MOAC algorithm is a 3-approximation algorithm* at all times *for the GDC problem within a strip.*

Proof. The proof is almost identical to that of Lemma 2.4.4, except now we define the "domain length" of each $ALGO_S$ MBN m_j separately, as $Q_j = |R_j - L_j|$.

Base Case: Again, recall that the length (along the strip) covered by d_1 (the leftmost optimal disk) is at most 1. This time we show that at most 3 $ALGO_S$ disks can simultaneously cover points from this interval where by assumption, no uncovered points exist to the left of d_1^L. Too see why, assume 4 such $ALGO_S$ disks existed. As before, this means that the member points of 3 of these $ALGO_S$ disks must all lay within this interval, requiring that $\sum_{j=1}^{3} Q_j \leq 1$. However, the merging rule of the algorithm implies that the sum of domain lengths of two adjacent disks must be $> \frac{2}{3}$. Furthermore, invariant (ii) of the algorithm states that the domain length of any disk must be $\geq \frac{1}{3}$. We therefore have that $Q_1 + Q_2 > \frac{2}{3}$ and $Q_3 \geq \frac{1}{3}$, which together imply $\sum_{j=1}^{3} Q_j > 1$, which is a contradiction.

Inductive Step: Assume the MOAC algorithm uses no more than $3m$ disks to cover all the points covered by $d_1, \ldots, d_m$, i.e. $b_m \leq 3m$. Now consider the number of additional disks it takes for the algorithm to cover the points covered by $d_1, \ldots, d_m, d_{m+1}$. Since all of the points up to the rightmost point of d_m are already covered, by the same argument as the base case the algorithm will use at most 3 extra disks to cover the points covered by d_{m+1}. Thus, we have that,

$$b_{m+1} \leq b_m + 3 \leq 3m + 3 \leq 3(m+1). \tag{2.17}$$

□

Both the computation complexity and communication complexity of the MOAC algorithm are always $O(1)$ per single node-movement. The only assumption required is that MBNs and disconnected RNs have access to information regarding L_j, d_j^L, d_j^R and R_j of their immediate neighbors to the right and left (as long as they are less than $2D$ away). Thus, in terms of complexity, the MOAC algorithm is by far the best of the distributed algorithms.

2.4.3 Merge-and-Separate (MAS) Algorithm

The relatively high approximation ratio of the MOAC algorithm results from the fact that it reduces disks into rectangles, thereby not taking advantage of about 35% of disk coverage area. The difficulty in dealing with disks is that there are no clear *borders* and that even confined to a single strip, many disks can overlap even though they cover disjoint nodes.

On average any algorithm with a merge rule should perform well. However, just having a merge rule is not sufficient in the rare but possible case where many mutually pairwise *non-mergeable* MBNs move into the same area. Based on this premise, we present the Merge-And-Separate (MAS) algorithm, as an algorithm which merges pairwise disks where possible (similar to the MOAC algorithm) and separates disks, if too many mutually non-mergeable disks concentrate in a small area. As will be shown, the MAS algorithm retains some of the localized features of the MOAC and obtains better performance ratio. However, this come at a cost of increased complexity.

We define the strip-widths as $q_{MAS} = \alpha D$ and set $D = 1$, $\alpha = \sqrt{5}/3$, $\sqrt{1-\alpha^2} = 2/3$. These are arbitrary values selected for the ease of presentation, the algorithm and the analysis are applicable to any $0.5 \leq \alpha < \sqrt{3}/2$. Let $x_{R_{\{i,j,k\}}}$ and $x_{L_{\{i,j,k\}}}$ be the x-coordinates of the rightmost and leftmost points of $\{P_{d_i} \cup P_{d_j} \cup P_{d_k}\}$. The algorithm is initialized by covering the nodes within a strip with MBNs by using the SCR algorithm. The algorithm that then operates at an MBN d_i is described below. Notice that Figure 2-9 demonstrates the separation done at lines 8-11.

Notice that rearrangement of MBNs is done to the *right* (see for example Figure 2-9). Clearly, this means that if a Separation event occurs at the far left of a crowded strip, this could initiate other Separation events along the remainder of the strip. Simple heuristic modification can be designed to deal with such a situation.

Define *steady state* as any point in time in which there are no merge or separate actions immediately pending. Below we describe the performance of the MAS algorithm.

Lemma 2.4.6. *In steady state, the MAS algorithm is a 2-approximation algorithm for the GDC problem within a strip.*

Proof. We prove the lemma by induction in a similar way to the proof of Lemma 2.4.2. We assume the same definitions of OPT_S, $ALGO_S$, and b_m as in that proof.

Algorithm 5 Merge-and-Seperate (MAS)

```
Initialization
 1: cover the RNs with MBNs using the SCR algorithm
 2: $P_{d_i} \leftarrow$ all RNs within $[L_i, R_i]$
Merge
 3: for all MBNs $d_k$ within $2D$ of $d_i$ do
 4:    if $\{P_{d_i} \bigcup P_{d_k}\}$ can be covered by a single MBN then
 5:       merge $d_i$ and $d_k$
Separation
 6: for all MBN pairs $d_j$, $d_k$ within $2D$ of $d_i$ do
 7:    if $|x_{R_{\{i,j,k\}}} - x_{L_{\{i,j,k\}}}| \leq 2D$ then
 8:       separate and reassign MBNs and RNs such that
 9:          $P_{d_i} \leftarrow$ all RNs in $[x_{L_{\{i,j,k\}}}, x_{L_{\{i,j,k\}}} + \frac{2}{3}]$
10:          $P_{d_j} \leftarrow$ all RNs in $[x_{L_{\{i,j,k\}}} + \frac{2}{3}, x_{L_{\{i,j,k\}}} + \frac{4}{3}]$
11:          $P_{d_k} \leftarrow$ all RNs in $[x_{L_{\{i,j,k\}}} + \frac{4}{3}, x_{R_{\{i,j,k\}}}]$
Creation
12: if an RN $p$ enters from a neighboring strip or an RN $p \in P_{d_i}$, moves s.t. MBN $d_i$
    cannot cover $P_{d_i}$ then
13:    create a virtual MBN for $p$
14:    if the virtual MBN cannot be merged with any of its neighbors then
15:       create a new MBN to cover $p$
```

Base Case: Consider the leftmost OPT_S disk d_1 and its member point-set P_{d_1}. Assume there exist 3 $ALGO_S$ disks that cover at least one point from P_{d_1}. However, if this was the case then all of the points covered by these 3 $ALGO_S$ disks would lie within an x-range of $2D$ (i.e. $[[d_1^L, d_1^L + 2D])$, and would be re-organized as per the separate rule of the in-strip MAS algorithm. Once re-organized, we note that since there exist no uncovered points left of d_1^L, that as per the separate rule of the MAS algorithm, the first 2 re-organized disks would cover all points within the x-range $[d_1^L, d_1^L + \frac{4}{3}D]$, and thus the third re-organized disk could not cover any points from P_{d_1}, which is a contradiction.

Inductive Step: Assume the in-strip MAS uses no more than $2m$ disks to cover all the points covered by $d_1, \ldots, d_m$, i.e. $b_m \leq 2m$. Now consider the number of additional disks it takes for the algorithm to cover the points covered by $d_1, \ldots, d_m, d_{m+1}$. Since all of the points up to the rightmost point of d_m are already covered, by the same argument as the base case the algorithm will use at most 2 extra disks to cover the points covered by d_{m+1}. Thus, we have that,

$$b_{m+1} \leq b_m + 2 \leq 2m + 2 \leq 2(m+1). \tag{2.18}$$

□

The computation complexity of the MAS algorithm is a periodic $O(C(n))$ to evaluate the merge and the create rules, where $C(n)$ is the running time of the 1-center subroutine used. On the other hand, since point transfers are local (e.g. only take place between adjacent MBNs), the communication complexity is $O(1)$. In order to make the required decisions, we assume that an MBN has access to all nearby (i.e. within a distance of $3D$) MBNs' point-sets and locations.

2.5 Placing the Relay MBNs

Recall that in Section 2.3 we showed that the CDC problem can be decomposed into two subproblems. In this section, we focus on the second subproblem that deals with a situation in which a set of nodes (*Cover MBNs*) is given and there is a need to place the minimum number of nodes (*Relay MBNs*) such that the resulting network is connected. Recall that the distance between connected MBNs cannot exceed R. This problem is equivalent to the Steiner Tree Problem with Minimum number of Steiner Points (STP-MSP) [64].

In [64] a 4-approximation algorithm that places nodes along edges of the Minimum Spanning Tree (MST) which connects the Cover MBNs has been proposed. In [20] an improved MST-based algorithm that provides an approximation ratio of 3 has been proposed. These algorithms are simple and perform reasonably well in practice. However, their main limitation is that they only find *MST-based* solutions. Namely, since the Relay MBNs are in general placed along the edges of the MST, these algorithms cannot find solutions in which a Relay MBN is used as a central junction that connects multiple other Relay MBNs. An example demonstrating this inefficiency appears in Figure 2-10.

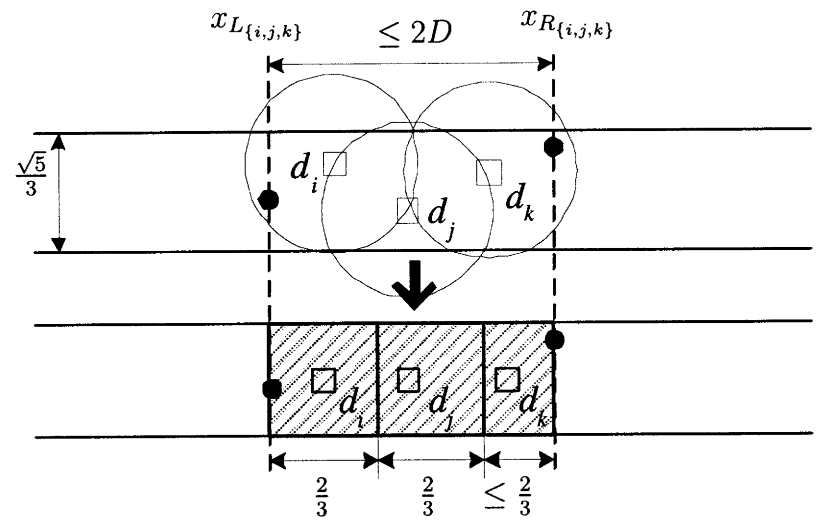

Figure 2-9: The Separation rule of the MAS algorithm

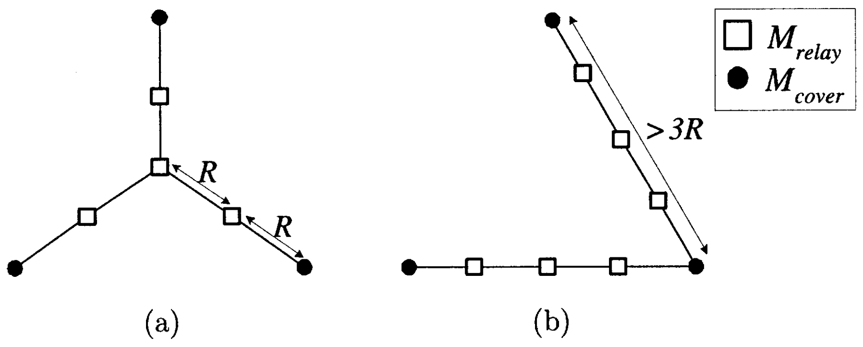

Figure 2-10: (a) Optimal STP-MSP solution (4 Relay MBNs). (b) MST-based solution (6 Relay MBNs).

Below we present and analyze a *Discretization Approach* which provides a theoretical footing towards the application of the vast family of discrete and combinatorial approaches (e.g. integer programming and local search) that can potentially rectify the above inefficiency. In particular, the approach transforms the STP-MSP from a Euclidean problem to a discrete problem on a graph. Although the transformed problem does not admit a constant factor approximation algorithm, in many practical cases it can be solved optimally. We will show that if such a solution is obtained, it is 2-approximation for the STP-MSP.

Our approach is based on an idea used by Provan [74] for dealing with the continuous analog of the STP-MSP problem, the well known Euclidean Steiner Tree (EST) problem [35]. In [74] it was proposed to discretize the plane and to solve a Network Steiner Tree problem [35] on the induced graph, yielding an efficient approximate solution for the EST. We utilize a similar approach towards solving the STP-MSP problem, which we present below. Note that our approach is quite different from the approach of [74], since the STP-MSP problem is more sensitive to discretizing the plane than the EST problem.

Define V_0 as the lattice of points in the plane generated by gridding the plane with horizontal/vertical spacing Δ, the exact value of which will be derived later. Next, define V_1 as the set of points associated with the pairwise intersections of radius R circles drawn around each of the Cover MBNs. For the intersection region of any two circles, add three equally spaced points along the line between the two intersection points. Let V_2 denote the set of these points. Finally, define $conv(M_{cover})$ as the convex hull of the of Cover MBNs. We can now define

$$V = \left\{ (V_0 \cup V_1 \cup V_2 \cup M_{cover}) \cap^* conv(M_{cover}) \right\}. \tag{2.19}$$

where we define a special intersection operator $\cap^*$ to ensure that we pick enough points to be in V such that $conv(V) \supseteq conv(M_{cover})$.

For all $u, v \in V$, if $d_{uv} \leq R$, we define an edge (u, v). We denote the set of edges by E and the *induced graph* by $G = (V, E)$. Let the node weights be denoted by w_v.

We now introduce the Node-Weighted Steiner Tree (NWST) problem [41],[60],[80], which has to be solved as part of our Discretization algorithm.

Problem NWST: Given a *node-weighted* undirected graph $G = (V, E)$ with zero-cost edges and a terminal set $M_{cover} \subseteq V$, find a minimum weight tree $T \subseteq G$ spanning M_{cover}.

Algorithm 6 Discretization

```
1: create the sets $V_0, V_1, V_2$, and $V$ {$\Delta$ derived below}
2: $w_v \leftarrow 1 \, \forall v \in V - M_{cover}$
3: $w_v \leftarrow 0 \, \forall v \in M_{cover}$
4: create the set $E$
5: find a minimum weight NWST on $G = (V, E)$
```

The set of nodes selected in step 5 correspond to the Relay MBNs in the STP-MSP solution. We assume that step 5 is performed by a β_{NWST}-approximation algorithm. The following theorem provides the performance guarantee of the above algorithm.

Theorem 2.5.1. *If $\Delta \leq \frac{R}{7}$, the Discretization algorithm is a $2\beta_{NWST}$-approximation algorithm for the STP-MSP.*

Our methodology in proving the theorem is as follows. We start by assuming the optimal STP-MSP tree is known, and we define an algorithm to construct a candidate Steiner tree T in G from this optimal tree. Notice that the optimal solution is of course not known, and therefore, T will not be constructed in practice. However, we will use the definition of T in order to bound the ratio between an *approximate solution* to the Node-Weighted Steiner Tree (NWST) problem in G to the *optimal solution* of the STP-MSP in the plane.

Recall that the set of terminals/Cover MBNs M_{cover} is given as input to the problem. Define $T_{OPT} = (M^*, E^*)$ as the optimal solution to the STP-MSP. The node set M^* is composed of the Cover MBNs M_{cover} and the optimal set of Relay MBNs denoted by M^*_{relay}. We now present an algorithm for the construction of a candidate tree $T = (M^T, E^T)$ in the graph $G = (V, E)$. An example of steps 4-5, 7, and 12-14 of the algorithm is illustrated in Figure 2-11.

Algorithm 7 Construction of a Feasible STP-MSP (CFS)

```
1: $M^T \leftarrow M_{cover}$
2: $E^T \leftarrow$ edges from $E^*$ connecting $M_{cover}$ nodes to each other
3: for all $u \in M^*_{relay}$ that have edges (in $E^*$) to a set of Cover MBNs (in $M_{cover}$) do
4:     add to $M^T$ a Relay MBN $u' \in V$ located at the nearest point to $u$ that can be
       directly connected to the same set of Cover MBNs
5:     add to $E^T$ edges connecting $u'$ and the set of Cover MBNs
6: for all $u \in M^*_{relay}$ that do not have edges (in $E^*$) to any Cover MBNs in $M_{cover}$
   do
7:     add to $M^T$ a Relay MBN $u' \in V$ located at the nearest point to $u$
8: for all Relay MBNs $u, v \in M^*_{relay}$ such that $(u, v) \in E^*$ do
9:     if $d_{u'v'} \leq R$ then
10:        add to $E^T$ an edge $(u', v')$
11:    else
12:        $w \leftarrow$ midpoint of the line segment $\overrightarrow{(u, v)}$
13:        add to $M^T$ a Relay MBN $w' \in V$ located at the nearest point to $w$
14:        add to $E^T$ edges $(u', w'), (w', v')$
```

In the following lemma we show that T is a feasible solution to the NWST problem in G.

Lemma 2.5.1. *If $\Delta \leq \frac{R}{7}$, then T, constructed by the CFS algorithm, is a Steiner tree in G.*

Proof. In this proof, we denote the Euclidean distance between nodes u and v by $|uv|$. We have to show that T connects all the nodes from M_{cover} by a tree whose nodes are in V and that the edges added to E^T are valid edges in E.

The nodes of T (i.e. M^T) are by definition in V, since they are selected from V. We can see that following Step 1 all the nodes from M_{cover} are included in T. Then, in Step 2 all the M_{cover} nodes that were directly connected to each other in T_{OPT} are similarly connected in T. In steps 3-5 all the Cover MBNs that were directly connected to a relay MBN in T_{OPT} are also similarly connected to new Relay MBNs (that connect the same groups of Cover MBNs) in T. The new Relay MBNs always exist and are always less than R away from their Cover MBNs, since V includes the intersections of radius R circles drawn around each of the Cover MBNs.

Up to this point all of the edges added to E^T are clearly of length at most R. We now show that this is the case for edges connecting new Relay MBNs as well.

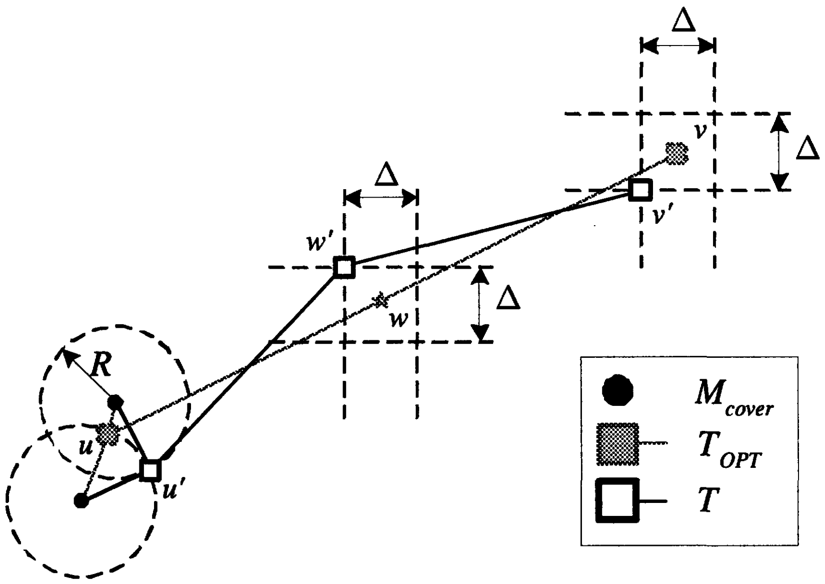

Figure 2-11: An example of the construction of the candidate tree T from the optimal STP-MSP tree T_{OPT}

In Step 7 each Relay MBN is replaced by a new Relay MBN which is a node in V. If two Relay MBNs are less than R from each other, they are connected in Step 10. It remains to show that if this is not the case, the new edges are shorter than R. Consider an edge (u, v), $u, v \in M^*_{relay}$ and the corresponding new edges in T - (u', w') and (w', v') (generated in Steps 13-14). We show that $|u'w'| \leq R$, thereby it is an edge in E (the proof for $|w'v'|$ is symmetric).

By the definition of the STP-MSP solution $|uw| \leq R/2$. In addition, by applying the triangle inequality to the distance between an arbitrary point w to the nearest grid point w' in V_0, we get that $|ww'| \leq \Delta$. Using these facts and the triangle inequality we have that,

$$|u'w'| \leq |u'u| + |uw| + |ww'| \leq |u'u| + \frac{R}{2} + \Delta. \tag{2.20}$$

Obtaining an upper bound on $|u'u|$ requires to take into account the case in which u is directly connected to multiple Cover MBNs. In such a case in step 4, u' may potentially have to be located at a specific point in $V_1 \cup V_2$ which is not necessarily its nearest point in V. Two scenarios have to be considered. In the first scenario, u'

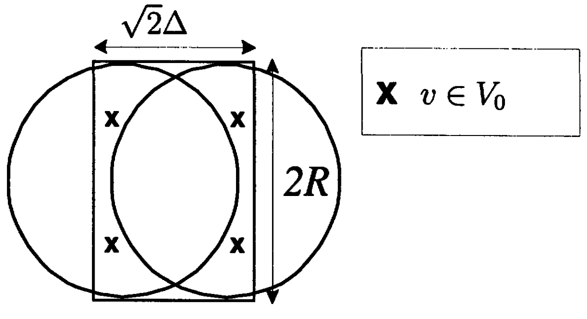

Figure 2-12: A loose upper bound on the area of any intersection region of 2 circles that *does not contain a grid point.*

can be located at a grid point in V_0. Namely, it can be placed at a grid point located in the intersection region of radius R circles centered around the Cover MBNs that u is directly connected to. In this case, $|u'u| \leq 2\Delta$, since u' can potentially be located at this (not necessarily the nearest) grid point.

In the second case, no point in V_0 is located in the relevant intersection region. In that case we can (loosely) bound the size of the intersection region by a $2R \times \sqrt{2}\Delta$ rectangle, as shown in Figure 2-12. Note that the $\sqrt{2}\Delta$ width of the rectangle corresponds the *diagonal* distance between grid points, and therefore, it is quite conservative. In the construction of V_2 we included 3 points along each line between two intersection points. Therefore, by using the triangle inequality, we get that $|u'u| \leq \Delta/\sqrt{2} + R/4$. Combining this with (2.20), we have that,

$$|u'w'| \leq \max\left(2\Delta, \frac{1}{\sqrt{2}}\Delta + \frac{R}{4}\right) + \frac{R}{2} + \Delta \qquad (2.21)$$

which is less than R if $\Delta \leq \frac{R}{7}$. □

In the following lemma we show that the number of Relay MBNs in T, denoted by $|M^T_{relay}| = |M^T| - |M_{cover}|$, is less than twice the number of Relay MBNs in the optimal solution of the STP-MSP (T_{OPT}).

Lemma 2.5.2. *In T, constructed by the CFS algorithm, $|M^T_{relay}| < 2|M^*_{relay}|$.*

Proof. In the CFS algorithm, each Relay MBN u in T_{OPT} is replaced by a Relay MBN u' in T (steps 4 and 7). For each edge connecting a pair of Relay MBNs in T_{OPT}, at most one additional MBN is added in T (w' in step 13). Since T_{OPT} is a tree, there can be at most $|M^*_{relay}| - 1$ such edges. Therefore, the total number of Relay MBNs in T is,

$$|M^T_{relay}| \leq |M^*_{relay}| + |M^*_{relay}| - 1 < 2|M^*_{relay}|. \tag{2.22}$$

$\square$

Proof of Theorem 2.5.1. Let the number of Relay MBNs in T_{OPT} and T be $|T_{OPT}| = |M^*_{relay}|$ and $|T| = |M^T_{relay}|$, respectively. Recall that in the Discretization algorithm, the Cover MBNs in G were assigned a weight of 0 and the other nodes were assigned a weight of 1. Let T^{NWST}_{OPT} be the optimal (minimum weight) Node-Weighted Steiner Tree (NWST) in G and denote its weight by $|T^{NWST}_{OPT}|$. Due to Lemma 2.5.1 when $\Delta \leq R/7$, T is a feasible solution to the NWST problem in G. Therefore, and due to Lemma 2.5.2,

$$|T^{NWST}_{OPT}| \leq |T| \leq 2|T_{OPT}|. \tag{2.23}$$

In Step 5 of the Discretization algorithm, the NWST problem in G is solved by a β_{NWST} approximation algorithm. We denote the obtained solution by T_{ALGO} and denote the number of Relay MBNs in this solution by $|T_{ALGO}|$. From (2.23) we get that

$$|T_{ALGO}| \leq \beta_{NWST}|T^{NWST}_{OPT}| \leq 2\beta_{NWST}|T_{OPT}|. \tag{2.24}$$

$\square$

It was shown in [60] that the NWST problem does not admit a constant factor approximation algorithm and that the best theoretically achievable approximation ratio is $\ln k$, where k is the number of terminals (in our formulation $k = |M_{cover}|$). For the case in which all node weights are equal, [41] indeed presented a $(\ln k)$-approximation algorithm. Thus, in general, the Discretization algorithm yields a worst case approximation ratio of $2 \ln |M_{cover}|$. However, in some cases the NWST problem can be solved optimally by discrete methods such as integer programming [80]. Since in such cases $\beta_{NWST} = 1$, the approximation ratio will be 2. Notice that it is likely that the Discretization algorithm will have better average performance than the MST-type algorithms, due to the use of Relay MBNs as central junctions.

Finally, it should be noted that the Discretization algorithm is centralized. Since this algorithm takes care of placing only the *Relay MBNs*, it might be feasible to implement it in a central location. Yet, if there is a need to solve the problem in a distributed manner, one of the MST-based algorithms [20] implemented with a distributed MST algorithm should be used. Although these algorithms are distributed, they do not deal very well with the mobility of Cover MBNs (i.e. a small change in the location of a Cover MBN may require repositioning several Relay MBNs). Thus, the development of distributed algorithms for the STP-MSP that take into account mobility remains an open problem.

2.6 Joint Solution

Using the decomposition method presented in Section 2.3, the overall approximation ratio of the CDC problem is the sum of the approximation ratios of the algorithms used to solve the subproblems. In this section, we note that the Discretized algorithm developed in the previous section can be applied towards solving the CDC problem. The result is that in *specific instances* when the Node-Weighted Steiner Tree (NWST) problem can be solved optimally (e.g. using integer programming), we can obtain a centralized 2-approximate solution for the CDC.

The key insight is that the CDC problem can be viewed as an extended variant of the STP-MSP problem. Namely, given a set of RNs (terminals) distributed in the plane, place the smallest set of MBNs (Steiner points) such that the RNs and MBNs form a connected network. Additionally, RNs must be *leaves in the tree*, and edges connecting them to the tree must be of length at most r. The remaining edges in the tree must be of at most R.

For the Discretization algorithm to apply, we need to make the following modifications. First, in the definition of the vertex set V, M_{cover} should be replaced with the set of RNs, N. Second, V_1 and V_2 should now be defined with respect to the

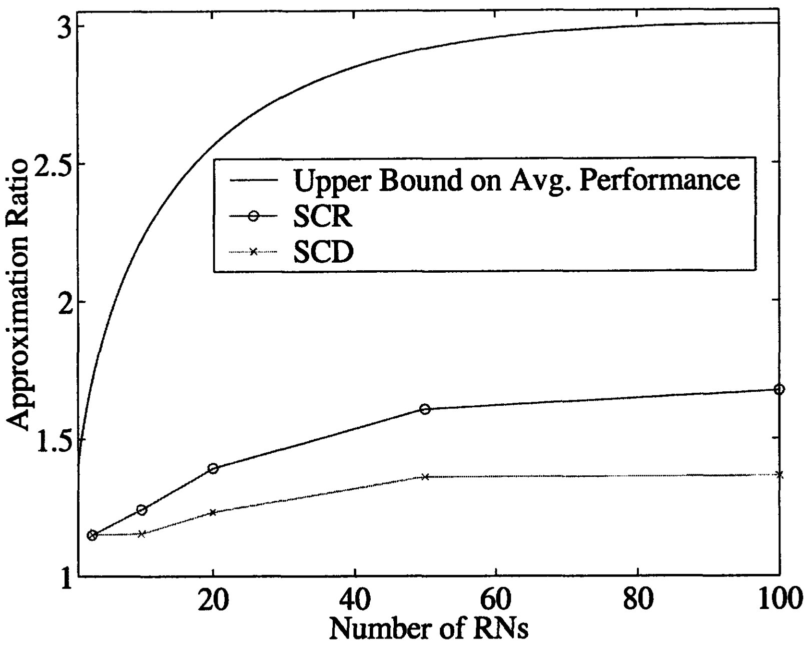

Figure 2-13: Ratios between the solutions by the SCD and SCR algorithms and the optimal solution, and an upper bound on average approximation ratios.

pairwise intersections of radius r circles drawn around each of the RNs. Finally, in the definition of the edge set E, RNs should only have edges to vertices in V within distance r, and no two RNs should have an edge between them. With these modifications, it can be shown that if $R \geq 2r$ and $\Delta \leq R/6$, the Discretization algorithm is a $2\beta_{NWST}$-approximation algorithm for the overall CDC problem.

2.7 Performance Evaluation

In this section we evaluate the performance of the algorithms via simulation. First, we evaluate the distributed GDC algorithms in both static and mobile environments. Then, we focus on the CDC problem and compare results obtained by the Discretization algorithm to results obtained by decomposing the problem. The results have been obtained by a simulation model of our algorithms, developed in Java.

For a network with static RNs, Figure 2-13 presents the average ratio between the solutions obtained by the SCD and SCR algorithms, and the optimal solution. For each data point, the average was obtained over 10 different random instances in

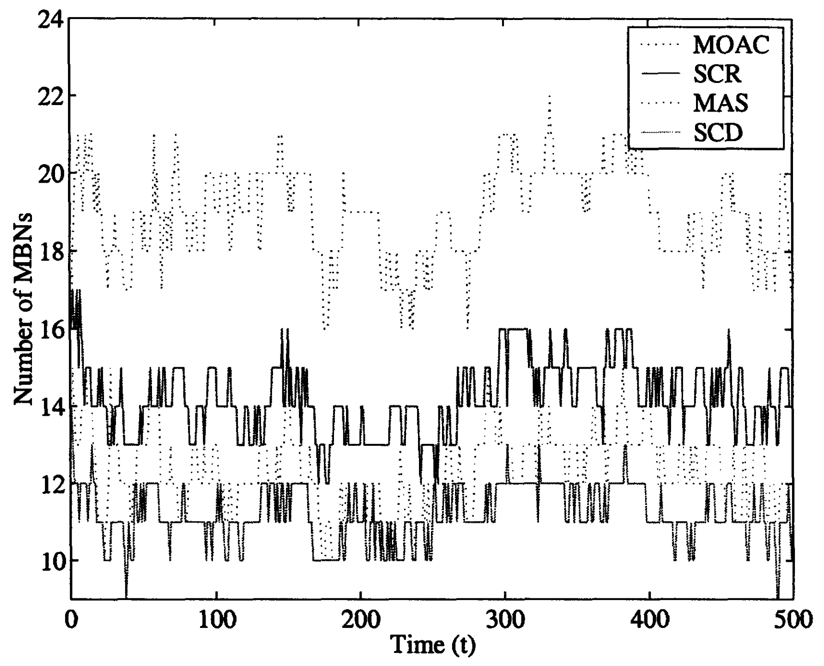

Figure 2-14: The number of Cover MBNs used by the GDC algorithms during a time period of 500s in a network of 80 RNs.

which the RNs are uniformly distributed in the plane. The optimal solutions were obtained by formulating *each instance* of the GDC problem as an Integer Program and solving it using CPLEX. It can be seen that although the worst case performance ratios of the SCR and SCD algorithms are 6 and 4.5, the average performance ratios are closer to 1.7 and 1.4, respectively. The figure also presents the upper bound on the average approximation ratios (β_{SCR} and β_{SCD}) derived in Theorem 2.4.3.[5] The large gap between the bound on the average approximation ratios and the actual ratios indicates that the bound is somewhat loose.

Table 2.1 shows the complexities and approximation ratios of the *distributed* GDC algorithms. It can be seen that there are clear tradeoffs between decentralization and approximation. These tradeoffs are further demonstrated by simulation. Figures 2-14 and 2-15 illustrate simulation results for a network with mobile RNs. The mobility model used is the Random Waypoint Model [57] in which RNs continually repeat

[5]Recall that in Theorem 2.4.3, we assume that the RNs are randomly distributed according to a two dimensional Poisson process. Therefore when the number of RNs is given, their positions are uniformly distributed in the plane.

Table 2.1: Time complexity (# of rounds), local computation complexity, and approximation ratio of the distributed GDC algorithms ($C(n)$ is the complexity of a decision 1-center algorithm).

Algorithm	*Time Complexity*	*Local Computation Complexity*	*In-Strip Approximation Ratio*
MOAC	$O(1)$	$O(\log n)$	3
SCR	$O(n)$	$O(\log n)$	2
MAS[6]	$O(1)$	$O(C(n))$	2
SCD	$O(n)$	$O(C(n)\log n)$	1.5

the process of picking a random destination in the plane and moving there at a random speed in the range $(0, V_{max}]$. We used a plane of dimensions $600m \times 600m$, set $V_{max} = 30m/s$, and set the RNs communication range as $r = 100m$.

Figure 2-14 depicts an example of the evolution (over a $500s$ time period) of the required number of MBNs used by the different GDC algorithms in a network with 80 RNs. As expected, the most distributed and least computationally complex algorithm (MOAC) performs the poorest, and the least distributed and most computationally complex algorithm (SCD) performs the best. Moreover, both algorithms that utilize 1-center subroutines (MAS and SCD) perform better than the MOAC and SCR algorithms, which reduce disks to rectangles. Figure 2-15 presents the average number of MBNs used over a $500s$ time period as a function of the number of RNs. Each data point is an average of 10 random instances. The same performance order as in Figure 2-14 is observed.

Next we compare solutions of the CDC problem obtained by the decomposition method to joint solutions obtained by the Discretization algorithm. Figure 2-16 depicts a random example of 10 RNs distributed in a $1000m \times 1000m$ area.[7] The communication ranges of the RNs and the MBNs are $r = 100m$ and $R = 200m$, respectively. In the decomposition method, we used an optimal disk cover (obtained

[6]The approximation ratio of the MAS algorithm holds when the algorithm is in steady state.

[7]We deliberately selected a small number of RNs in order to generate a partitioned network that requires Relay MBNs.

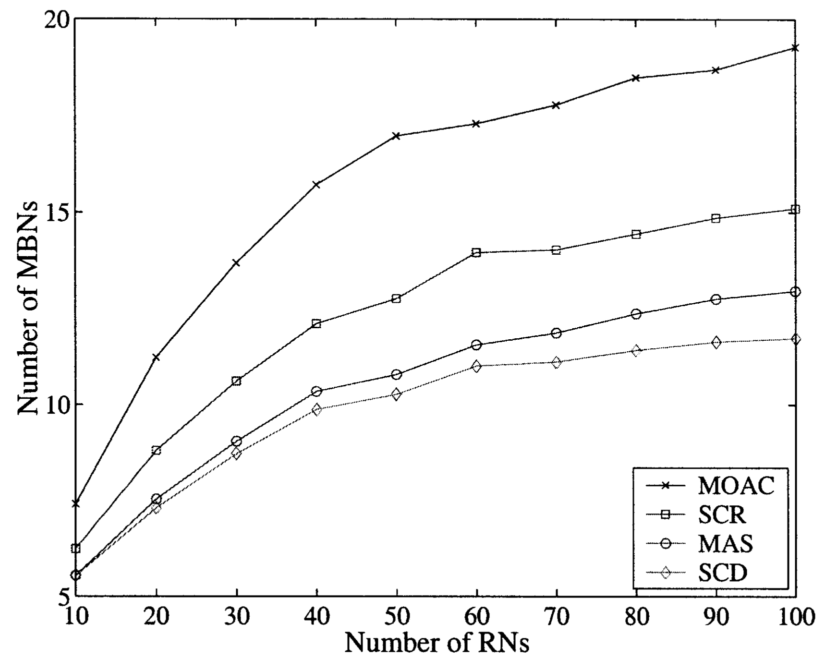

Figure 2-15: The average number of Cover MBNs used by GDC algorithms over a time period of 500s.

by integer programming) and the 3-approximation STP-MSP algorithm from [20]. The Discretization algorithm uses the NWST approximation algorithm from [60]. In this example, the joint solution requires 12 MBNs while the decomposition based solution requires 15 MBNs .

Figure 2-17 presents similar results for a more general case with the same parameters (area, r, and R). The Decomposition method used the SCD algorithm along with the MST algorithm [64] and along with the Modified MST-based algorithm [20]. Each data point is averaged over 10 random instances. It can be seen that the joint solution provides a significant performance improvement (about 25% for large number of RNs). Yet, while the decomposition method uses distributed algorithms, the joint solution must be obtained in a centralized manner. Thus, a reasonable compromise could be to place the Cover MBNs in a distributed manner and to place the Relay MBNs (e.g. Unmanned-Aerial-Vehicles) by a centralized Discretization algorithm.

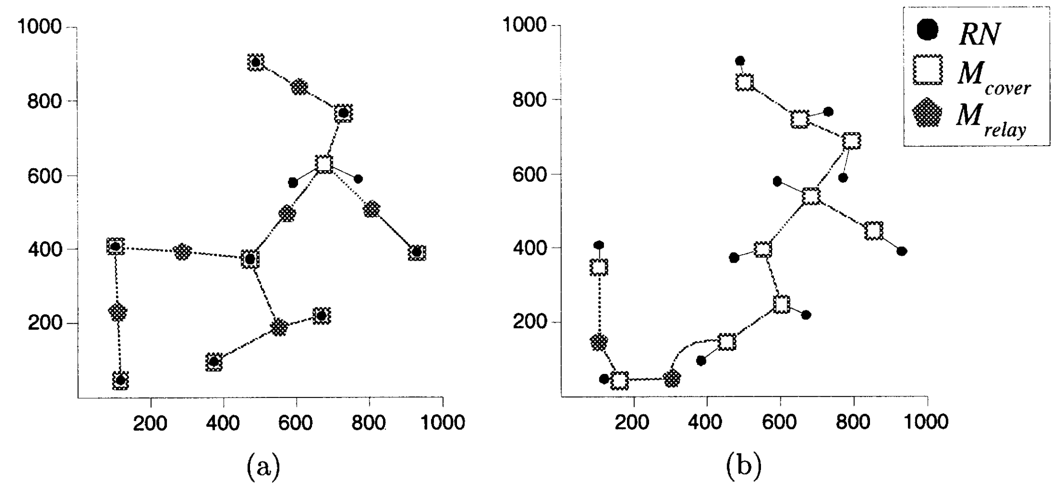

Figure 2-16: An example comparing solutions obtained by (a) an optimal Disk Cover and the STP-MSP algorithm from [20] and (b) the Discretization algorithm using an NWST algorithm [60].

2.8 Conclusions

The architecture of a hierarchical Mobile Backbone Network has been presented only recently. Such an architecture can significantly improve the performance, lifetime, and reliability of MANETs and Wireless Sensor Networks. In this chapter, we concentrate on placing and mobilizing backbone nodes, dedicated to maintaining connectivity of the regular nodes. We have formulated the Mobile Backbone Nodes placement problem as a Connected Disk Cover problem and shown that it can be decomposed into two subproblems. We have proposed a number of *distributed* algorithms for the first subproblem (Geometric Disk Cover), bounded their worst and average case performance, and studied their performance under mobility via simulation. As a byproduct, it has been shown that the approximation ratios of algorithms presented in [38] and [47] are 6 and 2 (instead of 8 and 3 as was shown in the past). A new approach for the solution of the second subproblem (STP-MSP) and of the joint problem (CDC) has also been proposed. We have demonstrated via simulation that when it is used to solve the CDC problem in a centralized manner, the number of the required MBNs is significantly reduced.

The work presented here is the first approach towards the design of distributed

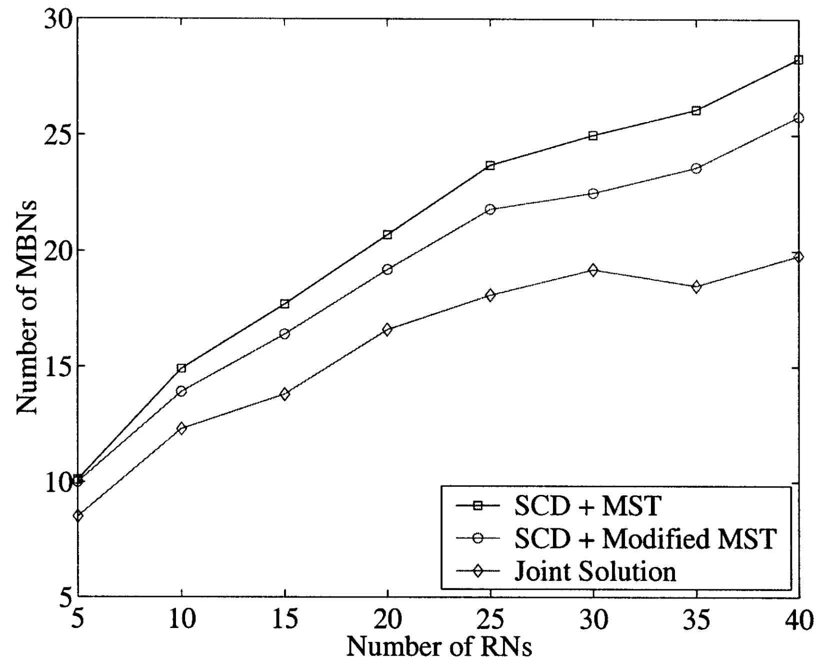

Figure 2-17: Number of MBNs as a function of the number of RNs computed by: (i) the decomposition approach using the SCD with the MST-based [64] algorithms, (ii) the decomposition approach using the SCD with the modified MST-based [20] algorithm, and the (iii) the Discretization algorithm.

algorithms for construction and maintenance of a Mobile Backbone Network. Hence, there are still many open problems to deal with. For example, it seems that the SCD algorithm can be generalized to yield a PTAS for the GDC problem in the strip. Also, moving away from the strip approach may be beneficial. Indeed, we present such *planar-based* distributed algorithms in appendix A. Thus, we intend to extend the MAS algorithm such that it will operate with disks in the plane. Finally, we note that there is a need for a distributed algorithm for the STP-MSP, capable of dealing with Cover MBNs mobility as well as for a mechanism for routing MBNs from their old locations to the new ones.

A major future research direction is to generalize the model to other connectivity constraints and other objective functions. Indeed, we address some of these issues in the chapter 3. Additionally, it would be desirable to consider the energy resources and the communication requirements of the RNs when making the mobility decisions.

Chapter 3

Joint Placement and Regular Node Assignment of a Fixed Number of Mobile Backbone Nodes

3.1 Introduction

An implicit assumption in previous formulations of the Mobile Backbone Network construction problem is that an arbitrary number of MBNs are available for deployment, and the goal is to minimize the number actually deployed. For example, such a problem formulation was given in chapter 2 as the *Connected Disk Cover* (CDC) problem. Specifically, the CDC problem aims to place the *minimum* number of MBNs such that (i) All RNs are covered by at least one MBN, and (ii) The MBNs form a connected network. In many scenarios however, a more appropriate (and perhaps realistic) assumption would be that the number of available MBNs is *fixed* a-priori, and the objective is to do the "best we can" with these fixed resources. As such, in this chapter we consider the problem of placing a fixed number of Mobile Backbone Nodes (MBNs), and assigning each Regular Node (RN) to exactly one MBN. The network objective we consider is to optimize RN throughput.

Note however, that the CDC-type formulation for MBN placement arises very naturally given the assumption of a discrete communications model, such as the "disk" connectivity model. In such a model, two nodes can communicate if they are within some fixed range, and cannot otherwise. However, while the disk model is a good first-order communications model, a more realistic model would account for the fact that the data rate at which two nodes can reliably communicate is actually a continuous function of the received Signal-to-Interference-and-Noise Ratio (SINR). The SINR in turn, depends on the wireless channel conditions and underlying PHY/MAC protocols (i.e. the System model). In this chapter and for the specific context of Mobile Backbone Networks, we distill these issues into the following general model: The "throughput" achieved by an RN transmitting to its assigned MBN is a *decreasing* function of (i) The distance between the RN and MBN, and (ii) The total number of RNs assigned to that MBN. The idea is that first factor models the loss due to wireless propagation, and the second models loss due to interference caused by multiple RNs trying to access a single MBN. We elaborate further on the mathematical specifics of the model, as well as provide examples in section 3.2.

With the above communications model, we are able to re-formulate the backbone construction problem in a manner significantly different from previous formulations, and thereby requiring significantly different solution methodologies. In particular, we consider the joint problem of *placing* a fixed number of MBNs, and *assigning* each RN to exactly one MBN, such that a throughput objective is maximized. We consider two objective functions, yielding two separate problems. The first is to maximize the throughput of the minimum throughput RN, which we term the Maximum Fair Placement and Assignment (MFPA) Problem. The second is to maximize the aggregate system throughput (i.e. sum of the throughputs achieved by each RN), which we term the Maximum Throughput Placement and Assignment (MTPA) problem.

It should be noted that in contrast to previous backbone construction problem formulations, the MFPA/MTPA involve a non-trivial assignment component. Specifically, a solution needs to balance assigning RNs to their closest MBNs and not assigning too many RNs to any particular MBN. Thus for the overall problems, not only do K MBNs need to be placed at arbitrary locations on the plane, but once placed there are K^N different RN to MBN assignments, among which the optimal one must be chosen, where N is the number of RNs.

Despite this, we are able to develop an optimal polynomial time algorithm for the MFPA problem for fixed K. We also develop an optimal solution for a restricted version of the MTPA problem for $K \leq 2$. As will be described later, the key lies in exploiting certain geometric properties of the placement portion of the problem, and certain combinatoric structure for the associated assignment subproblem. We also develop approximation and heuristic algorithms for both problems.

As a final point, to our knowledge the joint placement and assignment problems considered in this chapter have not been addressed before. Thus the primary goal of this chapter is to provide a theoretical framework and develop basic optimal solutions. We leave the development of more efficient, distributed and mobility-handling algorithms for future work.

This chapter is organized as follows. In Sections 3.2 and 3.3 we formulate the problem and give illustrative examples. Section 3.4 presents an optimal solution for the MFPA problem. In section 3.5, we discuss solutions for a restricted version of the MTPA problem. In section 3.6, we present approximation and heuristic algorithms for both problems. Finally, in section 3.7 we evaluate the performance of the algorithms via simulation.

3.2 Problem Formulation

We consider a set of N Regular Nodes (RNs), distributed in the plane and assume that a set of $K < N$ Mobile Backbone Nodes (MBNs) are to be deployed. We denote the set of RNs by $P = \{1, 2, \ldots, N\}$ and the set of MBNs by $M = \{m_1, m_2, \ldots, m_K\}$. For every RN i, let $m(i)$ denote the MBN to which i has been assigned, (e.g $m(i) = k$ if i is assigned to m_k), and let $d(i, m(i))$ represent the distance between them. In general, let $d(i, j)$ represent the distance between nodes i and j. Next, for every MBN m_k, let P_k denote the set of RNs assigned to it. Note that for any feasible solution, we have $\bigcup_k P_k = P$. Finally, we refer to the tuple of an MBN and its assigned RNs as a

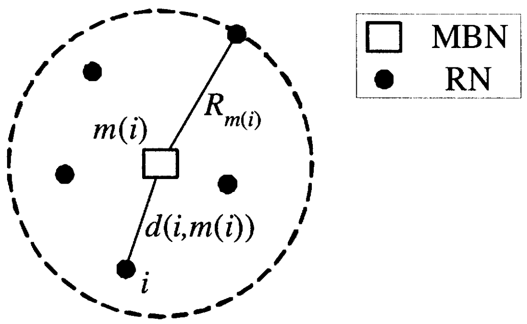

Figure 3-1: Example of a Cluster.

cluster. For cluster k corresponding to (m_k, P_k), we define the *cluster radius* R_k as, $R_k = max_{j \in P_k} d(j, m_k)$. The number of RNs assigned to MBN m_k, $|P_k|$, is referred to as the *cluster size*. An example of a cluster is shown in Fig. 3-1.

For the communications model, we assume that the throughput of an RN i transmitting to its assigned MBN $m(i)$ is some *function* $H\left(d(i, m(i)),\ |P_{m(i)}|\right)$, that is *decreasing* in both it's arguments. As mentioned earlier, the dependence of $H()$ on $d(i, m(i))$ models wireless propagation loss, and the dependence on $|P_{m(i)}|$ reflects loss due to interference at MBN $m(i)$. Note that in this communications model we assume that RNs from different MBNs do not interfere with each other, e.g. different clusters operate on different frequencies.

To gain some intuition about the form $H()$ could take, consider the following two system examples: (i) Slotted Aloha-based, and (ii) CDMA-based. In the Slotted Aloha based model, we assume that all RNs assigned to an MBN m_k transmit within a slot with equal probability, $1/|P_k|$. Additionally, we associate a "distance penalty" proportional to $d^{-\alpha}$ for an RN located a distance d away from m_k, where α represents the path loss exponent. This could, for example, reflect extra coding that needs to be used in order to deal with the propagation loss. The resulting throughput of a node i in this system is therefore simply the probability that exactly one RN transmits in a slot, multiplied by the distance penalty, i.e.,

$$\begin{aligned}TP_{SA}(i) &= \frac{1}{|P_{m(i)}|}\left(1-\frac{1}{|P_{m(i)}|}\right)^{|P_{m(i)}|-1}\left(\frac{1}{d(i,m(i))^{\alpha}}\right)\\ &\approx \left(\frac{1}{e\cdot|P_{m(i)}|\cdot d(i,m(i))^{\alpha}}\right)\\ &\triangleq H_{SA}\left(d(i,m(i)),|P_{m(i)}|\right)\end{aligned}\tag{3.1}$$

where we have left out most of the constants for simplicity, and we use the approximation that $(1-1/x)^{x-1}\to 1/e$ even for small values of $x\geq 1$. Note that (3.1) is of the desired form for $H()$, i.e. decreasing in both $d(i,m(i))$ and $|P_{m(i)}|$. Next, consider a CDMA-based system in which power control is employed. Specifically, in order to combat the near-far problem, all RNs assigned to an MBN $m(i)$ equalize their received power (equal to 1, for simplicity) at $m(i)$ to that of the farthest away RN. Thus the throughput achieved by every RN within a cluster is the same, and is proportional to its Signal-to-Interference-and-Noise Ratio (SINR) at $m(i)$, i.e.,

$$\begin{aligned}TP_{cdma}(i) &= \frac{\frac{1}{R^{\alpha}_{m(i)}}}{(\frac{1}{R^{\alpha}_{m(i)}})(|P_{m(i)}|-1)+\eta}\\ &= \frac{1}{|P_{m(i)}|+\eta\cdot R^{\alpha}_{m(i)}-1}\\ &\triangleq H_{cdma}\left(R_{m(i)},|P_{m(i)}|\right)\end{aligned}\tag{3.2}$$

where η represents the noise at MBN $m(i)$, and $R_{m(i)}$ the radius of cluster $m(i)$. Again, note the form of the throughput function is as desired, since it is decreasing in both distance and cluster size. For the purpose of intuition, we will carry these two examples throughout the paper, whenever possible directly applying to them the general results that we derive.

We now give a precise formulation for the two problems that will be addressed in this chapter: (i) The Maximum Fair Placement and Assignment (MFPA) Problem and (ii) Maximum Throughput Placement and Assignment (MTPA) problem.

Problem MFPA: Given a set of RNs (P) distributed in the plane, place K MBNs (M) and assign each RN i to exactly one MBN $m(i)$ such that,

$$\min_{i \in P} TP(i) = \min_{i \in P} \left\{ H\left(d(i, m(i)), |P_{m(i)}| \right) \right\} \tag{3.3}$$

is maximized.

Problem MTPA: Given a set of RNs (P) distributed in the plane, place K MBNs (M) and assign each RN i to exactly one MBN $m(i)$ such that,

$$\sum_{i \in P} TP(i) = \sum_{i \in P} H\left(d(i, m(i)), |P_{m(i)|} \right) \tag{3.4}$$

is maximized.

As a final point, we enforce the following additional conditions on the $H()$ function,

1. $H(R, X) > 0, \forall R \geq 0, X \geq 1$.

2. $H(R, X) < \infty, \forall R \geq 0, X \geq 1$ (only for MTPA)

Notice that condition (2) is needed for the general MTPA problem as stated above to be well defined. Otherwise, any solution in which an MBN is placed on top of an RN could yield infinite aggregate throughput (i.e. artificially exploiting the so-called "near-field" effect). Since $K < N$, this is not an issue for the MFPA problem, i.e. the worst case throughput RN cannot have an MBN on top of it.

3.3 Illustrative Examples

In this section we attempt to give some additional intuition regarding the complexity of the joint placement and assignment problems addressed in this chapter. To begin, consider a 1 MBN instance of the MFPA problem. With just one MBN, we immediately note that the assignment portion of the problem is trivial (i.e. all N RNs are assigned to the one MBN). Furthermore, the associated placement portion of the problem can be solved optimally by placing the single MBN so as to minimize the

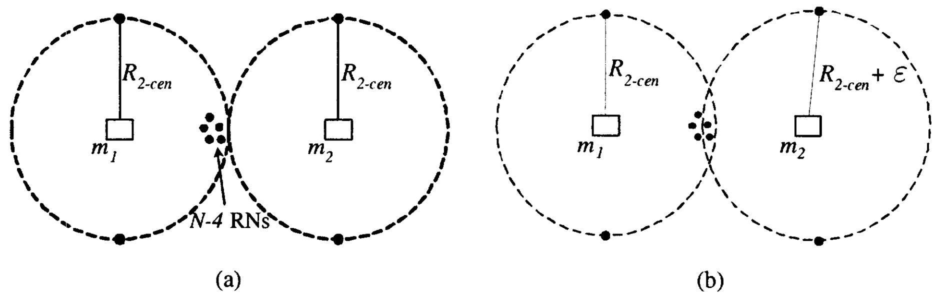

Figure 3-2: $K = 2$ MFPA example. (a) 2-Center Solution. (b) Optimal Solution.

farthest distance from any RN. This is precisely the well known 1-center problem[1], for which several efficient polynomial time algorithms exist [2]. Applying one of these algorithms solves the 1 MBN MFPA problem optimally.

Next, consider the 2 MBN example illustrated in Fig. 3-2. Fig. 3-2(a) shows the MFPA solution if we simply apply a 2-center algorithm, and assign RNs to their nearest MBN. As shown, the worst case RN attains a throughput of $H(R_{2-cen}, n-2)$ in this case, where R_{2-cen} is the 2-center radius. However, by increasing the radius of the second cluster by a small amount, i.e. enough to enclose half of the $n-4$ RNs clustered together, the optimal solution can potentially increase the worst case RNs' throughput to $H(R_{2-cen} + \epsilon, \frac{n}{2})$; this is shown in Fig. 3-2(b). Clearly depending on the exact form of $H()$, this improvement can be quite significant. As demonstrated in this simple example, even if we are given a placement of the MBNs, the assignment problem is non-trivial, as it may potentially be beneficial to assign RNs to farther away MBNs.

Thus the main difficulty of the MFPA and MTPA problems for $K > 1$ can be summarized as follows. First, there are an infinite number of potential locations for the MBNs (i.e. anywhere on the plane). Second, for any particular placement of K MBNs, there are K^N different assignments of RNs to MBNs (i.e. each RN can be assigned to one of K MBNs).

[1]In general, the K-center problem places K MBNs such that the farthest distance from any RN to its nearest MBN is minimized.

3.4 MFPA Solution

The key to our approach in solving the MFPA problem is to decouple the placement and assignment problems in a way that does not affect the optimality of the resulting decoupled solution. We start with the following observation and lemma. The observation applies to any feasible MFPA solution, and follows from the fact that the overall minimum throughput RN must be the minimum throughput RN in its own cluster.

Observation 3.4.1. *Let RN i have minimum throughput among all RNs, and let $m(i)$ be its assigned MBN. Then, the throughput of i can be expressed as a function of its cluster's radius and size, i.e. $TP(i) = H(R_{m(i)}, |P_{m(i)}|)$.*

Lemma 3.4.1. *Let $P_1^*, P_2^*, \ldots, P_K^*$ represent the optimal MFPA assignments of RNs to MBNs $m_1, m_2, \ldots, m_K$ respectively. Then, there exists an optimal solution to the overall MFPA problem in which the MBNs are placed at the* 1-center locations *of $P_1^*, P_2^*, \ldots, P_K^*$.*

Proof. Consider an optimal solution to the MFPA problem in which the MBNs are *not* placed at the 1-center locations of $P_1^*, \ldots, P_K^*$. Next, consider the solution obtained by moving all of the MBNs to their respective 1-center locations. By definition of the 1-center, doing this never increases the radius of any of the K clusters. Therefore, since the cluster sizes $|P_1^*|, \ldots, |P_K^*|$ are fixed, then by observation 3.4.1 the throughput of the worst case throughput RN does not decrease. □

The consequence of the above Lemma is that for the placement problem, the finite space of 1-center locations contains at least one solution of optimal cost. Additionally, the associated cluster radii of each of the K clusters are by definition 1-center radii. Thus as a first step, we have reduced the search space from an infinite number of locations on the plane, to a finite set of 1-center locations (with associated 1-center radii).

At first glance, the total number of 1-center locations/radii might seem prohibitively large and thus our reduction of limited use. For example, every subset

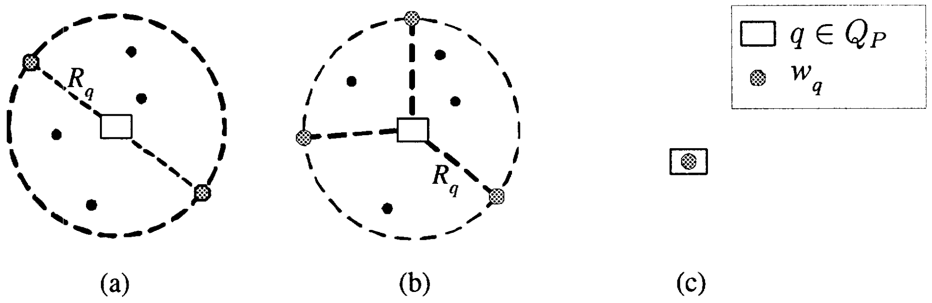

Figure 3-3: Illustration of the forms of 1-center (location,radius) tuples. (a) Midpoint of a pair of points. (b) Circumcenter of a triplet of points. (c) On top of a single point.

of RNs has an associated 1-center location and radius, and there are 2^N subsets. However, it turns out that all of these locations/radii come from a relatively small (i.e. polynomial in N) set of candidates. To show this, we need the following fact, illustrated in Fig. 3-3, regarding the 1-center of a set of RNs P [73],

***Fact* 3.4.1.** *The unique 1-center location and radius of a set of RNs P, denoted $1C(P)$ and $R(P)$, is* defined *by either:*

1. *A* pair *of RNs $i, j \in P$. If this is the case, then $1C(P)$ is situated at the* midpoint *of i,j, and $R(P) = d(i,j)/2$.*

2. *A* triplet *of RNs $i, j, k \in P$ that form an* acute triangle. *If this is the case, then $1C(P)$ is situated at the* circumcenter[2] *of $\{i, j, k\}$ and $R(P)$ is the* circumradius.

3. *A* single *RN $i \in P$. This is the degenerate case where $P = \{i\}$ is a singleton set, and $1C(P)$ is situated on i itself, and $R(P) = 0$.*

Indeed, the actual 1-center $(1C(P), R(P))$ tuple has minimum $R(P)$ such that all RNs are within distance $R(P)$ of the location $1C(P)$. Let Q_P denote the full set of candidate 1-center locations, as described in fact 3.4.1 with respect to the original set of RNs P. Note that since each $q \in Q_P$ is defined by either 1,2 or 3 RNs in P, it follows that that Q_P has cardinality at most $\binom{N}{1} + \binom{N}{2} + \binom{N}{3}$. Additionally, as

[2]For a triplet of RNs, the *circumcenter* is the center of the circle that has all three RNs on its boundary. The radius of this circle is the *circumradius*.

described in Fact 3.4.1 and shown in Fig. 3-3, for each $q \in Q_P$, we associate R_q to denote the *1-center radius* of a cluster whose 1-center location is q, and the set w_q to denote the set of *defining RNs* for q. Note that though several locations in the set Q_P may be coincident, all w_q's are distinct. We now state the following lemma, which follows by construction of Q_P and fact 3.4.1.

Lemma 3.4.2. *The 1-center (location, radius) tuple of* any subset $T \subseteq P$ *corresponds to some* (q, R_q) *tuple,* $q \in Q_P$.

Combining Lemmas 3.4.1 and 3.4.2 and Fact 3.4.1, we can conclude that restricting our placements of MBNs to the set Q_P still allows us to find the optimal solution to the overall MFPA problem. Moreover, we can restrict ourselves to solutions whereby if an MBN m_k is placed at location $q \in Q_P$, all of the RNs assigned to it must be within distance R_q, i.e. $d(i, m_k) \leq R_q$, $\forall i \in P_k$. Otherwise, by Fact 3.4.1 q cannot be the unique 1-center location of P_k, i.e. there must exist some other location $q' \in Q_P$ that is the actual 1-center location of P_k, with corresponding 1-center radius $R_{q'}$. As per lemma 3.4.1, moving m_k to location q' cannot decrease the MFPA objective.

For clarity, we illustrate the exhaustive search over all placements among locations in Q_P as the high-level framework presented below. We use the following notation for the overall solution. Let $m_1^*, \ldots, m_k^*$ represent the optimal locations of the K MBNs, $m^*(1), \ldots, m^*(N)$ the optimal RN to MBN assignments, and U^* the associated optimal cost.

Up to this point, we have not discussed the assignment subproblem, which we need to solve as a subroutine in step 5 of the high-level framework. It turns out that the specific methodologies used to solve this problem for $K = 2$ and $K > 2$ are quite different, as we describe below.

3.4.1 $K = 2$ MFPA Assignment Subproblem

With the placement locations and radii fixed, for $K = 2$ the resulting MFPA assignment subproblem turns out to be easy to solve. In this situation, as depicted in Fig. 3-4(a), we define $C(1)$ and $C(2)$ as the sets of RNs that lie *exclusively* within radius

Algorithm 8 High-Level Optimal MFPA Framework

```
1: initialize $U^* = -\infty$
2: create the set $Q_P$ by enumerating over all defining subsets of size 1, 2 and 3 of $P$.
3: for all $\binom{|Q_P|}{K}$ placements of K MBNs $m_1, \dots, m_K$ do
4:   if all RNs are within $R_j$ of at least 1 MBN $m_j$ in current MBN placement then
5:     calculate the optimal MFPA assignments $m(i), \forall i \in P$, given the current MBN placement and subject to the constraint that $m(i) = k$ only if $d(i, m_k) \leq R_k$. Let $U$ represent the corresponding worst case RN throughput.
6:     if $U > U^*$ then
7:       set $U^* \leftarrow U$, update $m^*(i), m_k^*, \forall i \in P, k \in K$
8: return $U^*$, $m_1^*, \dots, m_K^*$ and $m^*(1), \dots, m^*(N)$
```

R_1 and R_2 of MBNs m_1 and m_2 respectively. Similarly, let $C(1,2)$ denote the "common set" of RNs that lie within the radii of both m_1 and m_2. The main idea is that since the radii are fixed, RNs in $C(1), C(2)$ *must* be assigned to m_1, m_2 respectively. Moreover, in assigning the remaining RNs in $C(1,2)$, it is only the *number* assigned to each MBN that effects the MFPA objective. Thus we can search over the $|C(1,2)|+1$ different possibilities and pick the one that maximizes the throughput of worst case throughput RN.

The worst case computational complexity of the overall MFPA algorithm for $K = 2$ is therefore $O(N^7)$. This follows from the fact that $|Q_P| \leq N^3$ and we need to solve $\binom{|Q_P|}{2}$ assignment problems, each of which takes $O(N)$ time.

3.4.2 General K MFPA Assignment Subproblem

The MFPA assignment subproblem for $K > 2$ is significantly more difficult than for $K \leq 2$. To get a sense of the additional difficulty, consider the 2 vs. 3 MBN example illustrated in Fig. 3-4. For 2 MBNs m_1, m_2, there is only one type of "common set" of RNs, i.e. $C(1,2)$, yielding at most $O(N)$ ways to assign different numbers of RNs to each MBN.

For $K > 2$ MBNs, the number of ways to divide different numbers of RNs *within a single common set* generalizes to $O(N^{K-1})$. However, the real difficulty lies with the fact that for $K > 2$, there can potentially be many types of common sets. For

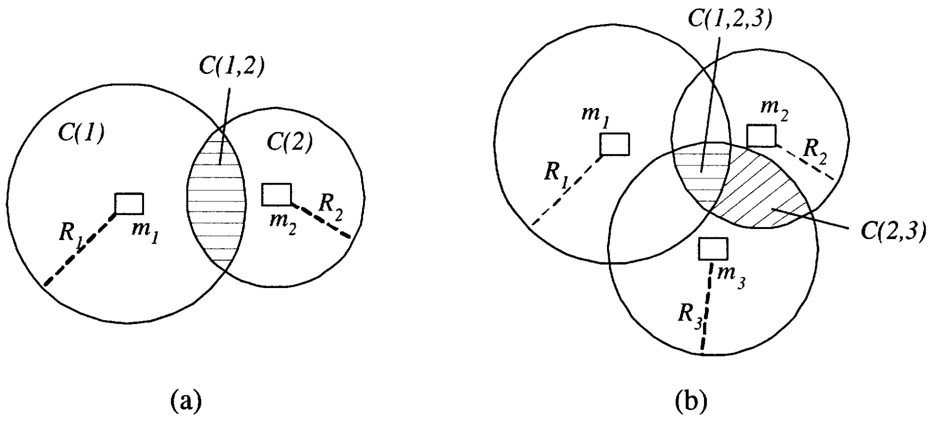

Figure 3-4: (a) $K = 2$ vs. (b) $K = 3$ examples of assignment subproblem.

example, in Fig. 3-4(b), RNs in the set $C(1, 2, 3)$ can be assigned to any of the 3 MBNs, whereas RNs in $C(2, 3)$ can only be assigned to either m_2 or m_3. Therefore, the total number of ways the RNs within all of these different common sets can be divided between K MBNs is $O((N^{K-1})^I)$, where I represents the number of distinct common sets. Observing that each MBN location and radius represents a circular region, we can actually bound I by K^2 [1]. This results in a total complexity of $O(N^{K^3})$ to enumerate all possible assignments. While this is still polynomial in N, spending this complexity for *each* of the $O(N^{3K})$ assignment subproblems yields an overall algorithm definitely outside the realm of practicality (e.g. even for $K = 3$).

With a more practical solution desired, we now develop an optimal algorithm for the general K MFPA assignment subproblem that is polynomial in *both* K and N. To this end, we start by formulating the MFPA assignment subproblem using a mathematical programming notation. Define indicator variables x_{ij} to equal to 1 if RN i is assigned to MBN m_j. Next, define indicator constants z_{ij} to be equal to 1 if $d(i, m_j) \leq R_j$. The resulting formulation can be written as,

$$\begin{aligned} \max \quad & \min_{j \in M} \quad H\big(R_j, \sum_{i \in P} x_{ij}\big) && (3.5) \\ s.t. \quad & \sum_{j \in M} x_{ij} = 1, \forall i \in P && (3.6) \\ & x_{ij} \leq z_{ij}, \forall i \in P, j \in M && (3.7) \\ & x_{ij} \in \{0, 1\} && (3.8) \end{aligned}$$

where constraints (3.6) ensure that every RN is assigned to exactly 1 MBN, constraints (3.7) that we only make valid assignments, and constraints (3.8) integrality of the final assignment. Defining the *increasing* function $F() = 1/H()$, since $H() > 0$, we can re-write the objective function in (3.5) as,

$$\min \quad \max_{j \in M} \quad F\big(R_j, \sum_{i \in P} x_{ij}\big) \qquad (3.9)$$

Applying one more transformation, we have,

$$\begin{aligned} \min \quad & W && (3.10) \\ s.t. \quad & \sum_{i \in P} x_{ij} \leq g(W; R_j), \forall j \in M && (3.11) \\ & \sum_{j \in M} x_{ij} = 1, \forall i \in P && (3.12) \\ & x_{ij} \leq z_{ij}, \forall i \in P, j \in M && (3.13) \\ & x_{ij} \in \{0, 1\} && (3.14) \end{aligned}$$

where we have used the common trick of converting a *minimax* objective function into a simple *min* objective function by introducing an extra real valued variable W and moving the *max* part of the objective function into the constraints. We define $g(W; R_j)$ to be the *inverse* with respect to $\sum_i x_{ij}$ of $F(R_j, \sum_i x_{ij})$, i.e.,

$$g\Big(F(R_j, \sum_i x_{ij})\ ;\ R_j\Big) = \sum_i x_{ij} \tag{3.15}$$

which we assume exists. Note that this assumption is justified given that $F()$ is a monotonically increasing function, and therefore constitutes a one-to-one (in $\sum_i x_{ij}$) continuous function. As an example, for the Slotted Aloha $H()$ given in (3.1) we have that $g(W; R_j) = W/(e \cdot R_j^{\alpha})$.

At this point, we note that the above *optimization* problem can be solved by way of solving a series of *feasibility* problems (e.g. fix W, and see if there exist x_{ij}'s that satisfy constraints (3.11)-(3.14)). One way of doing this is by performing a binary search over the space of all possible values of W. Specifically, if the problem is feasible for a given W, we can conclude the optimal value of W, denoted W^*, is such that $W^* \leq W$. Otherwise, $W^* \geq W$ can be concluded. The second way uses the following observation, and allows us to obtain an exact solution for W^*.

Observation 3.4.2. *The optimal W^* must satisfy $g(W^*; R_j) \in \mathbb{Z}$. That is, $g(W^*; R_j)$ must be integral.*

Proof. Since $g(W; R_j)$ is the inverse of the increasing function $F(\sum_i x_{ij}; R_j)$, it too must also be increasing (i.e. in W). Next, suppose the optimal W^* did not satisfy $g(W^*; R_j) \in \mathbb{Z}$. Since the x_{ij}'s are integral, this implies that the left hand side of constraint (3.11) must also be integral. Thus since $g(W; R_j)$ is increasing in W, it follows that we could have further reduced W^* until $g(W^*; R_j)$ reached $\lfloor g(W^*; R_j) \rfloor$, while still satisfying constraint (3.11). This contradicts the minimality of W^*. $\square$

We can combine this observation with the fact that there are at most $K \cdot N$ distinct integer feasible values for $g(W^*; R_j)$. Specifically, for each R_j (of which there are K), W^* can be one of $F(R_j, b), b = 1, \ldots, N$. Therefore, we can *exactly* find the optimal W^* by solving $K \cdot N$ feasibility problems.

The remaining question is: Given a value for W, how can we efficiently find (or not find) an assignment of x_{ij}'s that answers the feasibility question? To this end, we will now show that the feasibility problem can be transformed into a classical graph

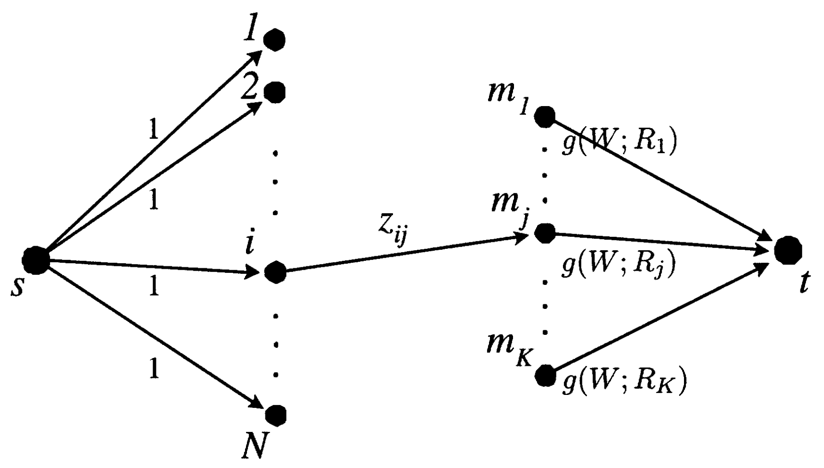

Figure 3-5: Construction of the Flow Graph $G = (V, E, C)$ for a given W.

problem, *Integer Max-Flow*, for which several efficient *polynomial time* algorithms exist [3].

The Integer Max-Flow problem is defined as follows: We are given a *flow graph* $G = (V, E, C)$, where C defines an integer set of *capacities* c_{ij} on each edge $(i, j) \in E$, and a *source* vertex s and a *sink* vertex t, $s, t \in V$. The objective is to assign a set of positive integer *flows* f_{ij} on each each edge $(i, j) \in E$ such that the aggregate flow from s to t, equal to $\sum_j f_{sj}$, is maximized. The f_{ij}'s must obey the following constraints:

1. $f_{ij} \leq c_{ij}, \forall (i, j) \in E$ (capacity constraints)

2. $\sum_i f_{ij} - \sum_k f_{jk} = 0, \forall j \in V \backslash \{s, t\}$ (flow conservation)

3. $\sum_i f_{is} = \sum_j f_{tj} = 0$ (source and sink property)

Returning to our problem, we start by constructing a flow graph $G = (V, E, C)$ in the following manner, depicted in Fig. 3-5. Let $P \in V$ represent a set of vertices corresponding to each RN, and similarly $M \in V$ for the MBNs. Next, define source and sink vertices $s, t \in V$. Next, define N source edges (s, i) with capacities $c(s, i) = 1$, $\forall i \in P$. Next, define edges between nodes (i, j) with capacities $c(i, j) = z_{ij}$, $\forall i \in P, j \in M$. Finally, define K sink edges (j, t) with capacities $c(j, t) = g(W; R_j)$,

$\forall j \in M$. At this point we run a Max-Flow algorithm to find the maximum (integral) flow between s and t in G. Given the Max-Flow solution, we interpret a non-zero flow $f_{ij} = 1$ on an edge of type $(i, j), i \in P, j \in M$ to mean that in the assignment solution RN i should be assigned to MBN j. Our main result lies in the following lemma, which we only prove for one direction; the converse holds by construction.

Lemma 3.4.3. *For a given W, the MFPA assignment subproblem is feasible if and only if the Max-Flow from s to t has value equal to N.*

Proof. Assume an integer max-flow of value N is found. To show this corresponds to a feasible solution to the MFPA assignment subproblem, it suffices to show that all of the constraints (3.11)-(3.13) are satisfied. Constraints (3.12) are satisfied since if the max-flow is equal to N, it must mean that all source edges carry a flow of 1. Thus by flow conservation, this implies that each RN (at the endpoint of each of the source edges) is assigned to exactly 1 MBN. Next, note that constraints (3.13) are satisfied by the fact that if edge $(i, j), i \in P, j \in M$ has non-zero flow across it, then by construction it's capacity, which is equal to z_{ij} must be equal to 1. Finally, constraints (3.11) are satisfied since if more than $g(W; R_j)$ RNs are assigned to any MBN m_j, this would correspond to edge (j, t) having a greater flow than it's assigned capacity. □

The preceding lemma gives us the final piece of the puzzle needed in order to construct an efficient algorithm for the MFPA assignment subproblem. The algorithm is given below.

We conclude the section by noting that the best Integer Max-Flow algorithm has running time $O(KN^2 \log N)$ [33]. Therefore, the algorithm depicted above has $O(K^2N^3 \log N)$ complexity. The result is a *worst case* complexity $O(N^{3K+3} \log N)$ algorithm for the *fixed* K MFPA problem. As will be shown in section 3.7, this algorithm can be applied to solve instances with relatively small K and N.

Algorithm 9 Fixed K MFPA assignment algorithm

```
1: initialize $W^* \leftarrow \infty$
2: for $k = 1$ to $K$ do
3:   for $b = 1$ to $N$ do
4:     set $W \leftarrow F(R_k, b)$
5:     if $W < W^*$ then
6:       construct flow graph $G = (V, E, C)$ as follows:
7:        set $V \leftarrow P \bigcup M \bigcup \{s, t\}$
8:        set $E \leftarrow E \bigcup \{(s, i)\}$, $c(s, i) \leftarrow 1, \forall i \in P$
9:        set $E \leftarrow E \bigcup \{(i, j)\}$, $c(i, j) \leftarrow z_{ij}, \forall i \in P, j \in M$
10:       set $E \leftarrow E \bigcup \{(j, t)\}$, $c(j, i) \leftarrow \lfloor g(W; R_j) \rfloor, \forall j \in M$
11:      solve $s - t$ Max-Flow on $G$. Let $f_{ij}$ be the flows on each edge $(i, j)$ and
         $F_{max}$ the max-flow value.
12:      if $F_{max} = N$ then
13:        set $m(i) \leftarrow j$ if $f_{ij} = 1$, $\forall i \in P, j \in M$
14:        set $W^* \leftarrow W$
15: return $W^*$, $m(1), \ldots, m(N)$
```

3.5 MTPA Solution

It turns out the general MTPA problem as formulated in 3.4 is significantly more difficult to optimally solve than the MFPA problem. For example, consider the MTPA problem for $K = 1$ MBN (i.e. ignore the assignment subproblem). At first glance it would seem like the MTPA problem looks like the well known 1-median/Fermat-Weber problem (numerically solvable in polynomial time [2]), in which one seeks to place the MBN in the location that minimizes the sum of the distances to each RN. However, the general MTPA objective is actually to maximize the sum of *arbitrary decreasing functions* of each of the distances; the difference is quite substantial. For example, consider a very simple decreasing function $H(d_i) = 1/(d_i + \gamma)$, where d_i represents the distance from RN i to the placed MBN and γ some positive constant. Clearly minimizing $\sum_i d_i$ achieves a significantly different objective from maximizing $\sum_i 1/(d_i + \gamma)$ (for which to our knowledge no optimal algorithm exists).

Thus we consider a restriction on the general MTPA problem, in which we enforce the condition on the $H()$ function that all RNs within a cluster get the same throughput, which is a function of the cluster radius and size, i.e.,

$$TP'(i) = H\Big(R_{m(i)}, |P_{m(i)}|\Big), \forall i \in P \tag{3.16}$$

The reasoning behind this particular restriction is two-fold. First, the above expression yields a lower bound on the general MTPA objective, i.e. since $H(d(i, m(i)), |P_{m(i)}|) \geq H(R_{m(i)}, |P_{m(i)}|), \forall i \in P$. It is therefore still useful to optimize. Second, this approach allows us to heavily leverage the discussion we have evolved through this chapter for the MFPA problem. To start, for $K = 1$ the 1-center algorithm optimally solves the restricted version of the MTPA problem.

For $K > 1$, we note that Observation 3.4.1 along with Lemmas 3.4.1-3.4.2 all apply to the restricted MTPA problem. Therefore, the high-level framework in section 3.4 solves the placement portion of the problem. Additionally, for $K = 2$ the simple MFPA assignment algorithm in section 3.4.1 also solves the restricted MTPA assignment subproblem, as long as the appropriate (i.e. MTPA) objective function is used.

For $K > 2$, the brute force approach discussed in the beginning of section 3.4.2 applies to the restricted MTPA problem. However, the fixed K MFPA assignment algorithm does not solve the fixed K restricted MTPA assignment problem. The reason for this is that while the MTPA problem can also be written as a Mixed-Integer-Linear-Program similar to (3.5), because the objective function is a *max sum* (as opposed to a *max min*), the max-flow technique cannot be applied.

3.6 Lower Complexity Heuristics

Although the algorithms developed so far in this chapter find optimal solutions in polynomial time, their complexity is still prohibitively high unless both K and N are quite small. For example for $K = 3$, $N = 35$, the running time of the optimal MFPA algorithm was 3 hours on a Pentium 2.4GHz computer.

Thus in this section, our goal is to develop *suboptimal* approaches that have significantly less running time than the optimal approach, but still perform comparably well. We will discuss 2 such approaches: (i) An approximation algorithm that is based

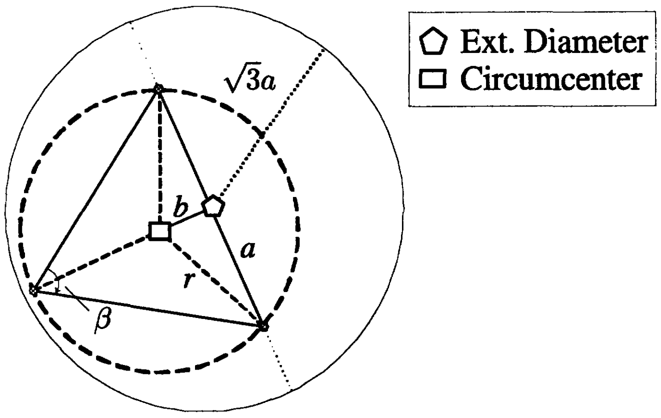

Figure 3-6: Extended Diameter-type vs. Circumcenter-type placement.

on cutting down the number of candidate MBN placements, and (ii) A simple and fast heuristic algorithm, but with no worst case performance guarantee. For the most part, the discussion applies to both the MFPA and restricted MTPA problems. For brevity, we will describe the algorithms in the context of the MFPA problem, noting any key issues specific to the restricted MTPA when appropriate.

3.6.1 Extended Diameter Algorithm (EDA)

As was discussed in section 3.4, the complexity of the optimal MFPA algorithm is dominated by the number of (optimality-preserving) possible placements, $\binom{|Q_P|}{K} = O(N^{3K})$. Indeed, the set Q_P is of size $O(N^3)$ because we had to consider all possible locations/radii corresponding to circumcenters/circumradii of triplets of RNs (see Fact 3.4.1). If we did not consider such "circumcenter-type" locations, but instead only considered locations defined by (i) the midpoint of pairs of RNs (i.e. "diameter-type") and (ii) single RNs (i.e. "singular type"), the number of possible placements would immediately be reduced to $O(N^{2K})$. This is the main idea behind the approach in this section.

Recall that in the high-level framework, we only consider placements at locations $q \in Q_P$, and assignments such that if an RN i is assigned to MBN m_k located at $q \in Q_P$, then $d(i, m_k) \leq R_q$. We denote such solutions as *valid.* However, an issue that comes up when we remove circumcenter-type locations from Q_P is that a valid solution may not even exist (e.g. consider 3 RNs that form a equilateral triangle).

To compensate for this, we define *extended-diameter* type locations, shown in Fig. 3-6, whose locations are the same as the original diameter-type locations, but whose associated radii are $\sqrt{3}$ times larger. Let Q'_P denote the set of all extended-diameter and singular-type locations with respect to a set of RNs P. Note that a direct analog with lemma 3.4.2 applies, i.e. Q'_P contains all extended-diameter and singular-type locations (with associated radii) with respect to any subset of RNs $T \subseteq P$. The next lemma ensures that placements among locations in Q'_P are guaranteed to contain a valid MFPA solution.

Lemma 3.6.1. *For a set of RNs P, there exists a valid solution to the MFPA problem with placements at locations in Q'_P.*

Proof. To prove the lemma, we need to show that for every circumcenter-type location/radii tuple in Q_P, there exists an extended-diameter-type location/radii tuple in Q'_P that covers the same set of RNs. To this end, consider some circumcenter-type placement, and the extended-diameter location corresponding to the midpoint of the *longest* side (of length $2a$) of the acute triangle formed by the circumcenters' defining RNs. The situation is depicted in Fig. 3-6. Let b be the distance between the extended-diameter and circumcenter locations. Next, let r denote the circumradius. By the triangle inequality, we know that the distance between the extended diameter location and any RN covered by the circumcenter placement is at most $b + r$. Therefore, we have that,

$$\begin{aligned} b + r &= r + \sqrt{r^2 - a^2} = \frac{a}{sin\beta} + \sqrt{\frac{a^2}{sin^2\beta} - a^2} \\ &\leq \frac{2a}{\sqrt{3}} + \frac{a}{\sqrt{3}} = \sqrt{3}a \qquad (3.17) \end{aligned}$$

where we have used a geometric property of circumcenters that $r = \frac{a}{sin\beta}$. Additionally, we have used the observation that since the defining triangle is acute and since the extended-diameter location under consideration is defined by the longest edge of the triangle, that $\pi/3 \leq \beta \leq \pi/2$. □

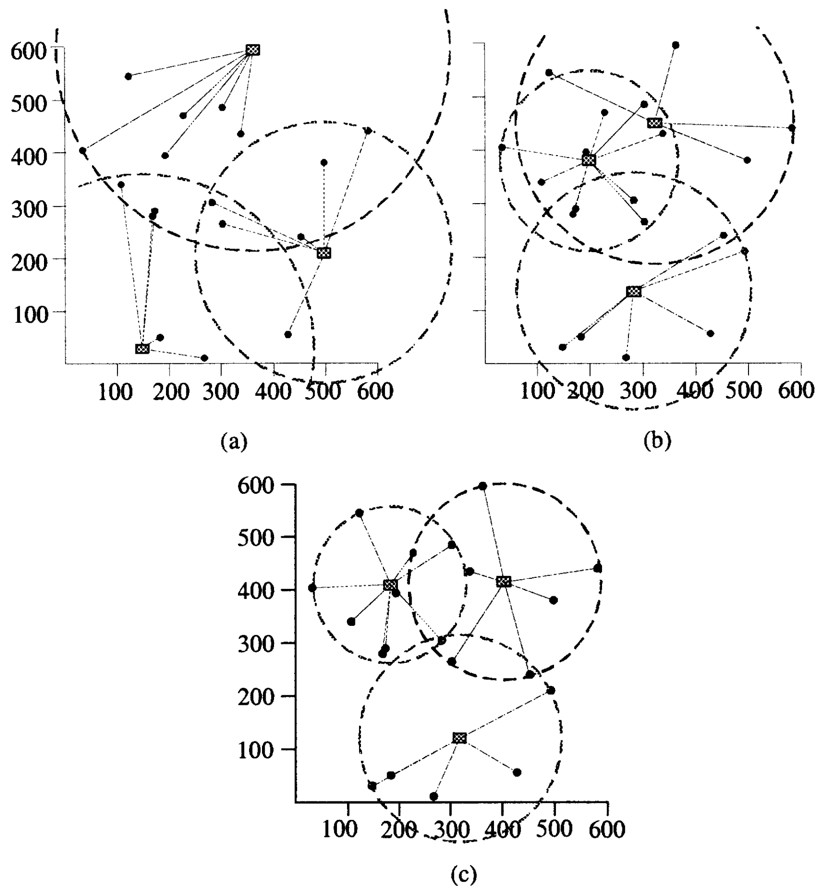

Figure 3-7: Example of solutions found for a $K = 3$, $N = 20$ instance of the MFPA problem with the Slotted Aloha throughput function. (a) Unoptimized Farthest Point Heuristic (FPH) (b) Unoptimized Extended Diameter Algorithm (c) Optimal MFPA algorithm.

Finally, define the *extended-diameter 1-center* of a set of RNs P as the location in Q'_P that minimizes the maximum distance from any RN in P. We now state the analog of lemma 3.4.1 applied to this context, whose proof follows from lemma 3.6.1.

Lemma 3.6.2. *Let $P_1^*, P_2^*, \ldots, P_K^*$ represent the optimal assignments of RNs to MBNs $m_1, m_2, \ldots, m_K$ respectively. Then, there exists a solution to the overall MFPA problem in which MBNs are placed at the extended-diameter 1-centers of $P_1^*, P_2^*, \ldots, P_K^*$. Also, the objective value of this solution is at least $H(\sqrt{3}R^*, |P^*|)$, where R^* and $|P^*|$ represent the worst case cluster radius and size of the optimal solution.*

We define the Extended-Diameter Algorithm (EDA) for the fixed K MFPA as well as the $K = 2$ MTPA problem, as basically the optimal algorithms described earlier, with Q'_P used in place of Q_P. The only difference is a final optimization step, in which after the suboptimal extended-diameter placement is decided, we move each of the MBNs to the actual 1-center location of their assigned RNs.

By the preceding discussion, the EDA algorithm is a $\frac{H(\sqrt{3}R^*,|P^*|)}{H(R^*,|P^*|)}$-approximation algorithm for the MFPA. Note that for a path loss exponent $\alpha = 2$ this ratio evaluates to $1/3$ for both the Slotted-Aloha and CDMA throughput functions. Finally, the worst case running time of the algorithm is $O(N^5)$ for $K = 2$, and $O(N^{2K+3} \log N)$ for general but fixed K.

3.6.2 Farthest Point Heuristic (FPH)

This next algorithm is simply an adaptation of Gonzalez's Farthest Point Heuristic (FPH) [39], with an additional optimization step tailored to our setting. The algorithm works as follows: Initialize the algorithm by placing an MBN on top an arbitrary RN, and assign all RNs to this MBN. Place the next MBN on top of the RN farthest from its assigned MBN, and re-assign RNs to their nearest MBN. Repeat the previous step until all K MBNs are placed. The above placement can be "optimized" by moving each MBN to the 1-center location of its assigned RNs. The running time of the unoptimized version of this algorithm is $O(N \log K)$, and assuming the use of a practical 1-center algorithm, the optimized version takes $O(KN \log N)$ time [2].

3.7 Simulation Results

In this section we compare the performance of the various algorithms presented in this chapter via simulation. To this end, we begin with an example of running the algorithms on a single $K = 3$ MBNs, $N = 20$ RNs, MFPA instance, shown in Fig. 3-7. We assume the RNs are randomly distributed in a 600×600 plane, and we use the Slotted-Aloha $H()$ throughput function given in (3.1), with $\alpha = 2$.

As can be seen, the optimal solution achieves the ideal balance between lightly loading clusters of large radii vs. heavily loading clusters of smaller radii. By contrast, the FPH solution potentially creates enormous radius clusters. Moreover, since

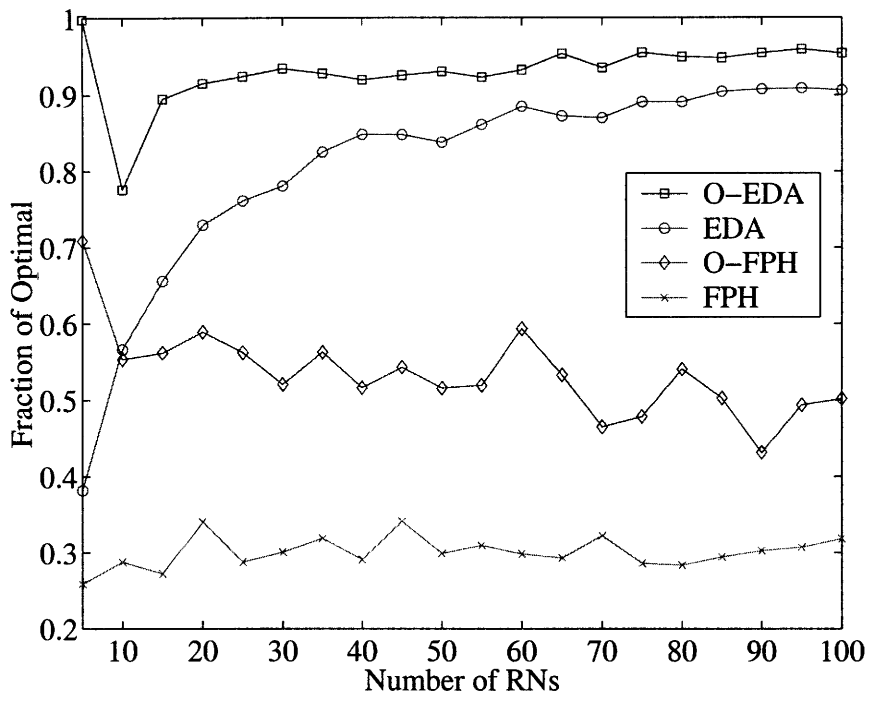

Figure 3-8: Average case simulation for $K = 2$ for the MFPA problem with Slotted Aloha throughput function

nothing intelligent is done by the FPH regarding the assignment problem (i.e. just assign RNs to their closest MBN), the large radius clusters can also get heavily loaded. The EDA does better, in that even though its cluster radii are larger than optimal, it intelligently assigns RNs in a way that achieves optimal load balancing among the placed clusters.

Figs. 8 and 9 show an average case plot for varying numbers of RNs, and $K = 2$ and $K = 3$ MBNs. All of the parameters are the same as for the previous scenario, and we average each data point over 10 random instances. The results are presented in terms of the average ratio of the throughput achieved by the suboptimal algorithms as compared to that achieved by the optimal algorithms described in sections 3.4. In both figures, we can notice that the optimization step significantly improves the performance of the heuristics. However, as exhibited by the poor performance of both the optimized and regular FPH, the optimization step can only help insofar as lowering the cluster radius if possible; it cannot make up for already-made poor assignment decisions.

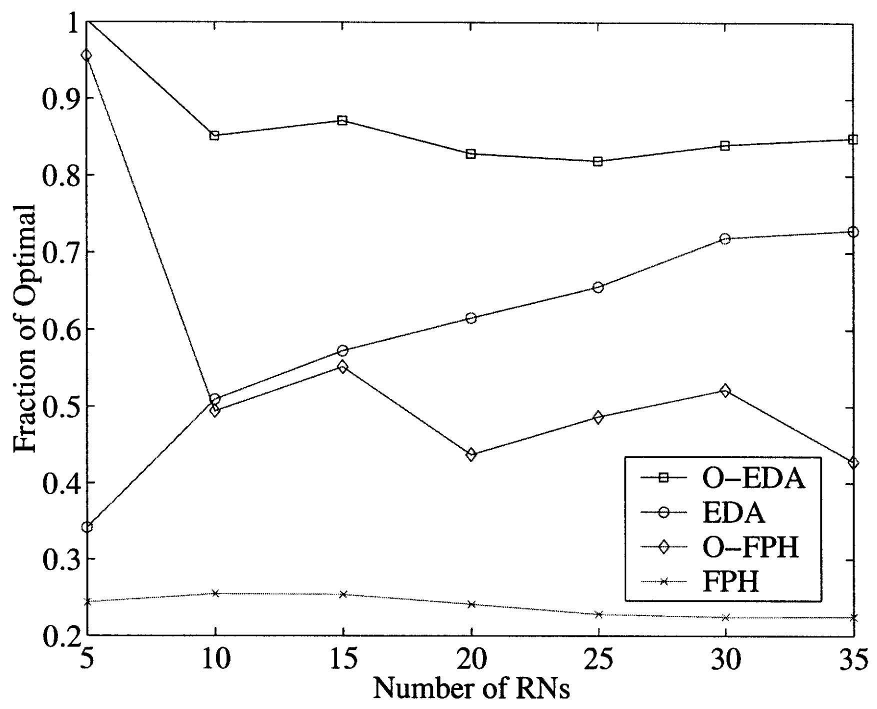

Figure 3-9: Average case simulation for $K = 3$ for the MFPA problem with Slotted Aloha throughput function

Finally, Fig. 10 shows an average case simulation for the $K = 2$ restricted MTPA problem with the CDMA throughput objective function from (3.2). We set $\eta = 10^{-4}$ in order to normalize the SNR somewhat, and add 1 to the denominator so as to maintain $H() < \infty$ as mentioned in section 3.2. Note that the O-EDA achieves aggregate throughput very close to optimal. In fact, all of the algorithms perform significantly better (relative to optimal) for the MTPA objective than for the MFPA objective, albeit with different $H()$ functions. Nevertheless, this would seem to indicate that the max-sum (i.e. MTPA) objective is less *sensitive* to suboptimal MBN placement/assignment than the max-min (i.e. MFPA) objective.

3.8 Conclusion

The recently studied Mobile Backbone Network architecture can significantly improve the performance, lifetime and reliability of MANETs and WSNs. In this chapter, we have focused on the key problem of how to jointly place the Mobile Backbone Nodes

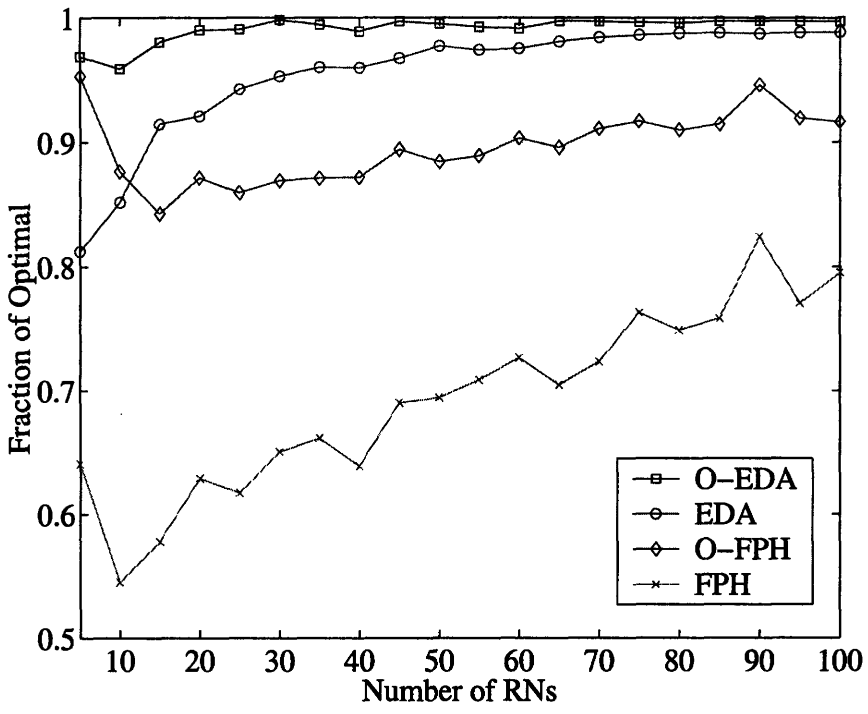

Figure 3-10: Average case simulation for $K = 2$ for the MTPA problem with CDMA throughput function

(MBNs), and assign every Regular Node to exactly one MBN. To this end, we have formulated two problems under a general communications model. The first is the Maximum Fair Placement and Assignment (MFPA) problem in which the objective is to maximize the throughput of the minimum throughput RN. The second is the Maximum Throughput Placement and Assignment (MTPA) problem, in which the objective is to maximize the aggregate throughput of the RNs. Our main result is a novel optimal polynomial time algorithm for the MFPA problem for fixed K. We have also provided an optimal solution for a restricted version of the MTPA problem for $K \leq 2$. We have developed two heuristic algorithms for both problems, including an approximation algorithm with bounded worst case performance loss. Finally, we have presented simulation results to evaluate the performance of the various algorithms developed in the paper.

To our knowledge the problems presented in this chapter have not been considered before. Thus for this paper, our primary goal has been to provide a theoretical framework, as well as basic optimal solutions. Future work involves the development of more efficient, distributed and mobility-handling algorithms for both the MFPA and MTPA problems.

Chapter 4

Optimal Mobile Backbone Path Planning

4.1 Introduction

Previous formulations of the Mobile Backbone Network construction problem have been based on knowledge of the "current" locations of the RNs. Specifically, at any given time, the MBNs are placed and mobilized *reactively* based on RNs' locations at that time. Indeed, this was the approach taken in chapters 2 and 3 of this thesis. Yet, in many practical scenarios entire RN trajectories are known a-priori (e.g. as way-points for particular missions). If this is the case, then placing the MBNs by solving a placement problem independently at each time step is, in general, suboptimal. In particular, it would be desirable to solve for the entire optimal *sequence of placements* for the MBNs at once. In this chapter, we address this MBN *path planning* problem both from a discrete and continuous perspective. For our exposition, we consider planning the path of a single MBN given the trajectories of the RNs. Our goal is to optimize throughput metrics along the lines of the those presented in chapter 3, averaged over the time horizon under consideration.

It is important to note that if the throughput metrics are simply time-averaged and no consideration is given to the actual movement of the MBN, then there is a straightforward way to calculate the optimal MBN paths. Specifically, combining the optimal solutions at each time step yields the overall optimal path. For example, we can obtain such solutions by employing the optimal algorithms developed in Chapter 3 independently at each time step. However, such an objective function can result in undesirable solutions for instances in which the required MBN motion in consecutive time steps is very large, even when the actual RN movement is small. Fig. 4-1, adapted from [12], shows a 3 RN example of such a situation. The figure shows the bottom 2 RNs moving a distance d. It can be shown that in response to this specific motion, the optimal 1-center[1] actually moves a distance $x = \sqrt{\frac{d^2+2dR}{2}}$, where R is the radius of the circle formed by the RNs as shown in the figure. Recall that in section 3.3 it was shown that for a single MBN, the placement that maximizes the MFPA objective is precisely the 1-center of the RNs. Expressed as a ratio, the MBN must move $\sqrt{\frac{1}{2}+\frac{R}{d}}$-times faster than the RNs. Hence this ratio can be made arbitrarily large by simply increasing R. Additionally, there can be scenarios in which it is undesirable to have large MBN movements even in response to large RN movements, e.g. limited MBN velocity, energy efficiency, MBN location predictability, etc.

To address this issue, we introduce a penalty and/or constraint (we consider both) on the MBN velocity. This immediately causes a dependence between the solutions at each time step, thereby considerably increasing the difficulty of the overall path planning problem. For example, at a particular time instance, it may be the case that significant (throughput) sub-optimality is necessary to allow for greater throughput at a future time instance. We provide an illustration of such a situation in section 4.3. Thus choosing the optimal time sequence of placements would seem to necessitate an algorithm that solves for the entire path at once, as opposed to a simpler (e.g. greedy) solution wherein independently solved instantaneous solutions are concatenated together. In section 4.4 we develop such an algorithm, based on dynamic-programming, in which the entire MBN path is directly solved for. In section 4.4.1 we characterize its performance with respect to the optimal solution. Interestingly, in section 4.5.1 we are able to show that for the single time step velocity constrained MBN path

[1]The 1-center problem places a single MBNs such that the farthest distance from any RN to the MBN is minimized.

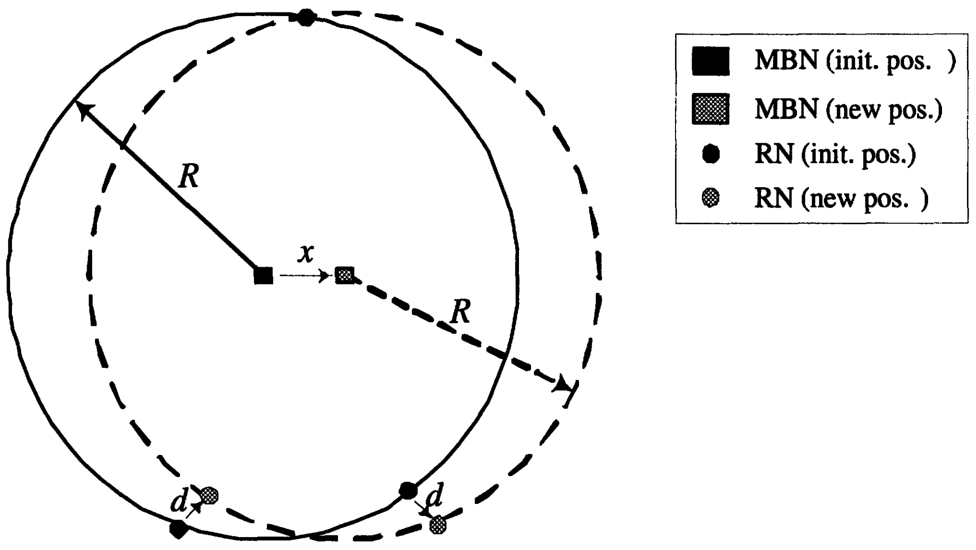

Figure 4-1: An example in which small RN movements cause a large deviation in the optimal MBN position.

planning problem, an optimal solution can be obtained via a combinatoric algorithm. We use this algorithm as a subroutine for a greedy approach developed in section 4.5. We compare both approaches via simulation in section 4.9.

The preceding discussion implicitly assumed a discrete sequence of time steps, over which the MBN path planning problem is considered. An alternate approach is from a continuous perspective. Under such a formulation, we assume RN trajectories are given as continuous functions of time over the finite time horizon. Thus to solve for the optimal trajectory of the MBN we can employ techniques from Optimal Control (OC) theory. In general this theory has been very well studied in the literature. However, as we will elaborate upon in section 4.7, there are several complexities associated with this approach for the MBN path planning problem in particular. One such complexity is that it turns out the formulation falls into a category of OC problems known as *Singular Control Problems.* Such problems are in general more difficult to solve optimally [18],[7]. Thus in this work we formulate a simpler problem instance involving a single RN, and are able to gain significant insight into the optimal MBN trajectory by examining the OC optimality conditions. In 4.9 we numerically simulate an example RN trajectory and compare the resulting performance to the discrete algorithms developed in sections 4.4 and 4.5.

Finally, to our knowledge the MBN path planning problem with a throughput maximization objective has not been considered in the literature. Yet, as mentioned in section 1.2.3 several closely related problems and formulations have been considered. The goal of this chapter is to provide a basic formulation from both a discrete and continuous perspective, as well a characterization of several natural solution methodologies.

The remainder of this chapter is organized as follows. In section 4.2 we provide a discrete problem model and formulation. We next develop a dynamic-programming based approximation algorithm in section 4.4. This is followed our development of a greedy algorithmic approach in section 4.5. We discuss relaxing the hard velocity constraint in the base discrete formulation in section 4.6. In section 4.7 we formulate an MBN path planning problem as a continuous time optimal control problem, and in section 4.8 we discuss extensions to this formulation. Finally, in section 4.9 we present simulation results comparing the various approaches developed in this chapter.

4.2 Discrete Problem Formulation

We consider a network consisting of N RNs $P = \{p_1, p_2, \ldots, p_N\}$ and a single MBN M. Our interest is over a finite time horizon $[0, T]$, discretized by Δt-spaced time steps $t = 0, 1, \ldots, K$, $K = T/\Delta t$. We assume all of the nodes in the network are situated on a 2-dimensional plane. We denote by $p_i(t) \triangleq (p_{i_x}(t), p_{i_y}(t))$, $t = 0, 1, \ldots, K$, the x-y position of RN p_i at time step t. Similarly, we define $M(t) \triangleq (M_x(t), M_y(t))$ for the MBN M. Let $d[u, v]$ denote the Euclidean distance between two nodes u and v. Next, we let $d_i(t)$ denote the distance between RN p_i and the MBN M at time step t, i.e., $d_i(t) = d[M(t), p_i(t)]$. Finally, let $d_{max}(t)$ represent the distance from the MBN to the farthest RN at time step t, i.e., $d_{max}(t) = \max_i d_i(t)$.

We assume the trajectories of the RNs are known a-priori over the full time horizon $t = 0, 1, \ldots, K$. Thus the goal is to compute a path $M^* = M(0), M(1), \ldots, M(K)$

for the MBN given this information. We assume the initial position of the MBN M_0 is fixed and known, i.e. $M(0) = M_0$. Finally, we enforce a hard constraint that the maximum speed of the MBN is upper bounded by V, i.e., $d[M(t-1), M(t)] \leq V\Delta t$, $\forall t = 1, \dots, K$. We discuss relaxing this constraint in section 4.6.

In this work we are concerned with maximizing the time average MFPA throughput objective from (3.3), i.e. $\frac{1}{K}\sum_{t=1}^{K} H[d_{max}(t)]$. As was described in detail in section 3.2, we assume $H()$ is a decreasing function that represents the throughput received by the worst case throughput RN. It can also serve as a proxy to describe the system throughput under certain system models, e.g. the CDMA Model described in section 3.3. We term the MBN path planning problem with time average MFPA objective function the *MPP-MFPA* problem, and formulate it below.

Problem MPP-MFPA: Given the RN trajectories $p_i(t), \forall i \in P, t = 0, \dots, K$ and initial MBN position $M(0) = M_0$. Compute the MBN path $M^* = M(0), M(1), \dots, M(K)$ such that the average MFPA throughput metric is maximized, subject to the maximum MBN speed bounded by V. Mathematically, the MPP-MFPA problem is expressed as,

$$\max_{M^*} \quad \frac{1}{K}\sum_{t=1}^{K} H[d_{max}(t)] \tag{4.1}$$

$$s.t. \quad d[M(t-1), M(t)] \leq V\Delta t, \quad \forall t = 1, \dots, T \tag{4.2}$$

$$M(0) = M_0 \tag{4.3}$$

4.3 Illustrative Example

In this section we provide an example that will allow us to gain insight into the MPP-MFPA problem. The example, involving a single RN travelling on a line is illustrated in Fig. 4-2. The trajectory of the RN is shown on top, and we assume that the MBN's speed is bounded by $V = 2$, and that $\Delta t = 1$, i.e. $K = T$. In the example both the RN and MBN start at the same location on the line, i.e. $M(0) = p(0) = 0$. Note that in the MPP-MFPA formulation, the speed of the RN is not bounded, and in the

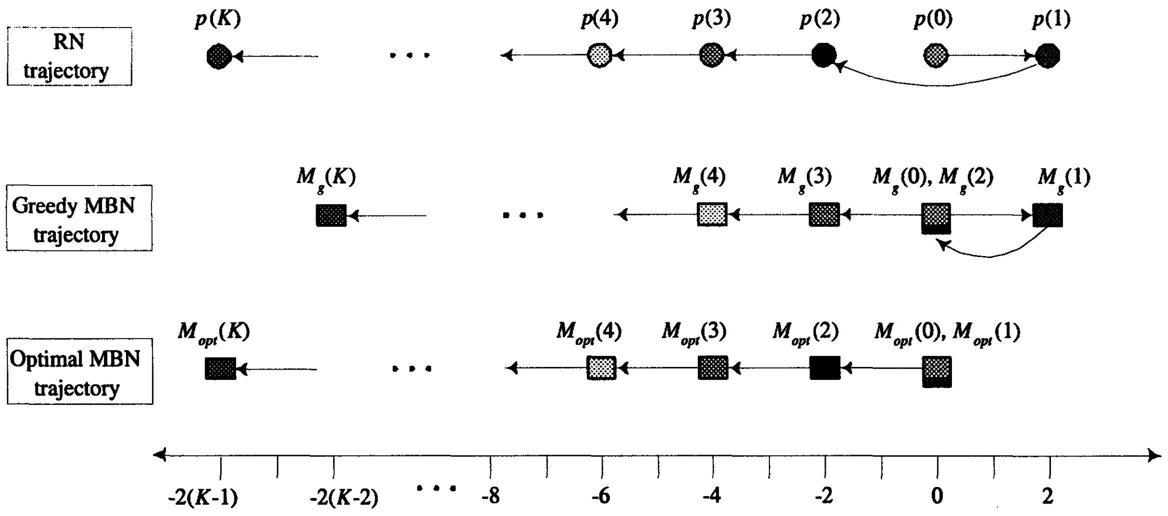

Figure 4-2: Single RN, Single MBN, 1-D example of greedy vs. optimal approach to MPP problem. Assume $M_g(0) = M_{opt}(0) = p(0) = 0$ and $V = 2$, $\Delta t = 1$.

example it travels at speed equal to 4 between time steps 1 and 2. In many scenarios, RNs might not travel faster then the MBNs. Yet, as shown in Fig. 4-1 the 1-center of the RNs certainly can travel much faster than any particular RN movement. Thus one can also think of the RN in the example as a proxy for the 1-center of a number of RNs.

An MBN path obtained by applying a greedy (i.e. myopic) approach that tries to maximize the instantaneous MFPA objective at every time step is shown in the middle of the figure. Notice that for all time steps $t \geq 2$, the greedy MBN trails the RN by a distance of 2. This results in an MPP-MFPA objective of $\frac{1}{K}[H(0) + (K-1)H(2)]$. By contrast, the optimal MBN path involves accepting some sub-optimality in the first time step by staying at position 0 at time step $t = 1$. However, doing this allows the optimal MBN to follow the RN exactly for all time steps $t \geq 2$, yielding an MPP-MFPA objective of $\frac{1}{K}[H(2) + (K-1)H(0)]$. Depending on the exact form of $H()$, this can be significantly larger than that achieved by the greedy approach.

The example shows that for certain problem instances a greedy solution approach can be highly sub-optimal. For such instances, solutions obtained by solving for the entire path at once are necessary for good performance.

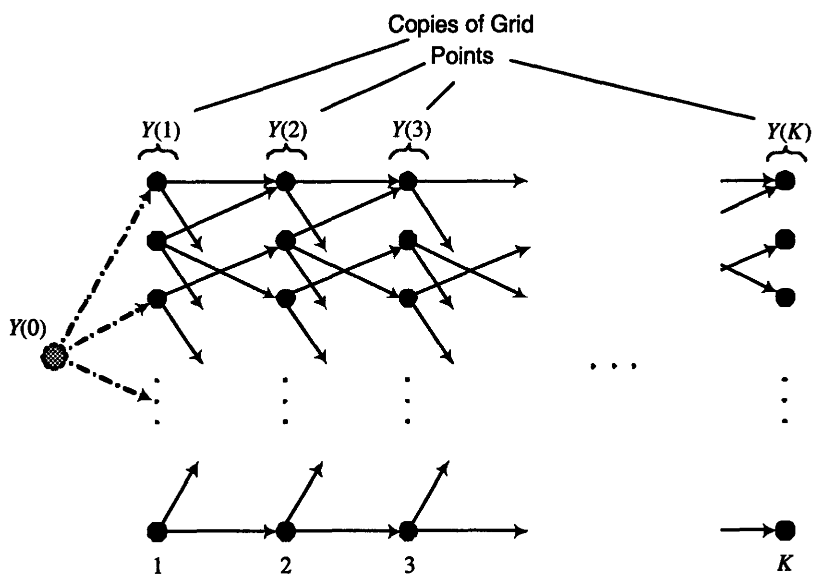

Figure 4-3: Illustration of Trellis Structure. Edges between vertices at consecutive time steps are drawn only if the grid points they represent are at most $V\Delta t$ distance apart.

4.4 DP-based Approximation (DPA) Algorithm

In this section we provide a Dynamic-Programming (DP) based approximation algorithm to solve the MPP-MFPA problem. We start by gridding the plane with vertical and horizontal spacing $\epsilon \leq V\Delta t$. Next, we construct K copies of the resultant grid points, denoted by $Y(1), Y(2), \ldots, Y(K)$, where a grid point $v \in Y(t)$ will represent a potential location for the MBN at time t. For notational convenience, we define the set $Y(0)$ to denote just a single point, M_0, i.e. the given starting position of the MBN.

We next define an edgeweighted graph $G = (V', E)$, illustrated in Fig. 4-3, as follows. Let the vertex set V' consist of all the $Y(t)$'s, $t = 0, 1, \ldots, K$. We add an edge (u, v) to E between $u \in Y(t)$, $v \in Y(t+1)$, $t = 0, \ldots, K-1$, if $d(u, v) \leq V\Delta t$, where $d(u, v)$ is the distance between grid points u and v. Constructing the edge set in this way restricts the MBN to only travel between grid points in successive time steps that are at most a distance $V\Delta t$ apart. Finally, we define the weight $w(u, v)$ of an edge $(u, v) \in E$, $u \in Y(t)$, $v \in Y(t+1)$, $t = 0, \ldots, K-1$ to equal to the *instantaneous* *throughput value assuming the MBN is located at v at time $t+1$.* Specifically, this is expressed as $w(u,v) = H[\max_i\{d(v, p_i(t+1))\}]$.

We now state the following lemma, which forms the main justification for the algorithm.

Lemma 4.4.1. *Assume the MBN is restricted to travel between grid points during time steps $t = 1, \ldots, K$. The optimal MPP-MFPA path* subject to this restriction *is equivalent to the longest (maximum weight) path in G from the vertex $Y(0)$ to some vertex $v \in Y(K)$*

Proof. Consider a path in the graph $Q = q(0), q(1), \ldots, q(K)$, $q(t) \in Y(t), \forall t$ where by construction $q(0) = M_0$. If we add up the edge weights along the path Q we obtain the expression,

$$Weight(Q) = \sum_{t=1}^{K} H[\max_i\{d(q(t), p_i(t))\}] \tag{4.4}$$

Clearly maximizing the above is equivalent to maximizing (4.1) subject to the grid-point restriction. □

Note that in general graphs, finding a longest path is NP-complete. However, the specific graph structure implied by the graph G' is known as a *Trellis Graph*, or more generally, a *Directed Acyclic Graph.* In such graphs, the longest path can be easily found in a manner similar to finding the shortest path. Specifically, at each vertex we can maintain the maximum weight (as opposed to the minimum weight) path from the source to that vertex.

Furthermore, we take advantage of the repeated edge structure between vertices in $Y(t)$ and $Y(t+1)$, $t = 1, \ldots, K-1$. This greatly simplifies the longest path problem, and instead of requiring a computationally complex algorithm such as Dijkstra or Bellman-Ford, we can draw an analogy from coding theory to note that a longest path in a Trellis Graph can be found using the Viterbi algorithm[49]. This is adapted into the DPA algorithm, which is presented below. For the algorithm, we utilize the following notation. Let $Y(1)$ denote a single copy of the grid points. We denote the distance metric of the longest path from M_0 to grid point v at time t as $q_v(t)$. We denote the predecessor grid point along this path as $p_v(t)$.

Algorithm 10 DPA Algorithm

```
Initialization:
1: set $q_v(t) = -\infty$, $p_v(t) = \emptyset$, $\forall v \in Y(1), t = 1, \dots, K$.
2: set $LP(0) = M_0$.
3: for $v \in Y(1)$ s.t. $d[M_0, v] \le V\Delta t$ do
4:     set $q_v(1) = H[\max_i\{d(v, p_i(1))\}]$, $p_v(1) = M_0$.
Metric Update:
5: for $t = 2, 3, \dots, K$ do
6:     for $u \in Y(1)$ do
7:         for $v \in Y(1)$, s.t. $d(u, v) \le V\Delta t$ do
8:             if $q_u(t-1) + H[\max_i\{d(v, p_i(t))\}] > q_v(t)$ then
9:                 set $q_v(t) = q_u(t-1) + H[\max_i\{d(v, p_i(t))\}]$ and $p_v(t) = u$
Calculate Longest Path:
10: let $v = \operatorname{argmax}_{u \in Y(1)} q_u(K)$ and $LP(K) = v$
11: for $t = K, K-1, \dots, 1$ do
12:     set $v = p_v(t)$
13:     set $LP(t) = v$
14: return $LP = \{LP(0), LP(1), \dots, LP(K)\}$
```

As a final point, the computational complexity of the algorithm is equal to $O\big(|Y(1)| \cdot (\lceil \frac{2V\Delta t}{\epsilon} \rceil)^2 \cdot K\big)$, where $|Y(1)|$ is the total number of grid points. Note that for a plane of dimensions $L \times L$, $|Y(1)| = (\lfloor \frac{L}{\epsilon} \rfloor + 1)^2$. We obtain the complexity result by examining the Metric Update (lines 5-9) portion of the algorithm. Note that the outer two loops iterate for K time steps, and over $|Y(1)|$ vertices respectively. Finally, for each vertex $v \in Y(1)$, we can upper bound the number of surrounding grid points that are within distance $V\Delta t$, by $(\lceil \frac{2V\Delta t}{\epsilon} \rceil)^2$.

4.4.1 Analysis

As mentioned in Lemma 4.4.1, the DPA algorithm finds the optimal MPP-MFPA path subject to the constraint that the MBNs must only travel between grid points. Yet, it would be desirable to calculate how close this solution approximates the original unconstrained optimal MPP-MFPA solution. The Lemma below shows that for an unbounded plane, this can be a function of both the grid spacing ϵ as well as the end time step K. We define $d_{max}^{opt}(t)$ to be the distance from the MBN to the farthest RN at time step t in the optimal solution (i.e. not constrained to lie on grid points). Additionally, we define a *grid square* to denote the induced square area from a square configuration of 4 *corner grid points.* We use the notation $a \in A$ to denote that the point a is located within the grid square A. Finally, two grid squares are *adjacent* if they share at least one corner grid point.

Theorem 4.4.1. *For an unbounded plane, the MPP-MFPA objective value of the solution path found by the DPA algorithm is* at least $\frac{1}{K}\sum_{t=1}^{K} H(d_{max}^{opt}(t) + 2\sqrt{2}t\epsilon)$.

In order to prove the above theorem, we will first need the following two lemmas.

Lemma 4.4.2. *Assume there exist two points $a \in A$ and $b \in B$ such that $d[a, b] = V\Delta t$. Then, there exists a grid square C, adjacent to B, such that the following two facts hold: (i) We can always find a corner grid point $c' \in C$ such that $d[a, c'] \leq V\Delta t$, and (ii) For any corner grid point $a' \in A$ we can always find a corner grid point $c' \in C$ such that $d[a', c'] \leq V\Delta t$.*

Proof. Assume the grid squares A and B are *not* co-linear, and without loss of generality[2], that A is situated some number of grid squares below and to the left with respect to B. This is depicted in Fig. 4-4-a. If this is the case, then it must be that the bottom-left corner grid point in B, denoted b', must be the closest point in B to any point in A. Define C as the grid square immediately bottom-left of B (i.e. B and C share the corner grid point b'). Thus we can conclude that $d[a, b'] \leq d[a, b] = V\Delta t$. Next, note that the top-right corner grid point in A, denoted a', must be the closest point in A to any point in B. Thus, we have that $d[a', b'] \leq V\Delta t$. Finally, we can reach, within distance $V\Delta t$, a corner grid point in C from any corner grid point $a'' \in A$. We do this by simply translating the line segment between a' and b' such that a' is shifted to a''. This is shown in the figure.

Next, assume the grid squares A and B are co-linear, and without loss of generality, that A is some number of grid squares left of B. This is depicted in Fig. 4-4-b. If

[2]The same analysis can be applied for all other non-co-linear configurations of A and B

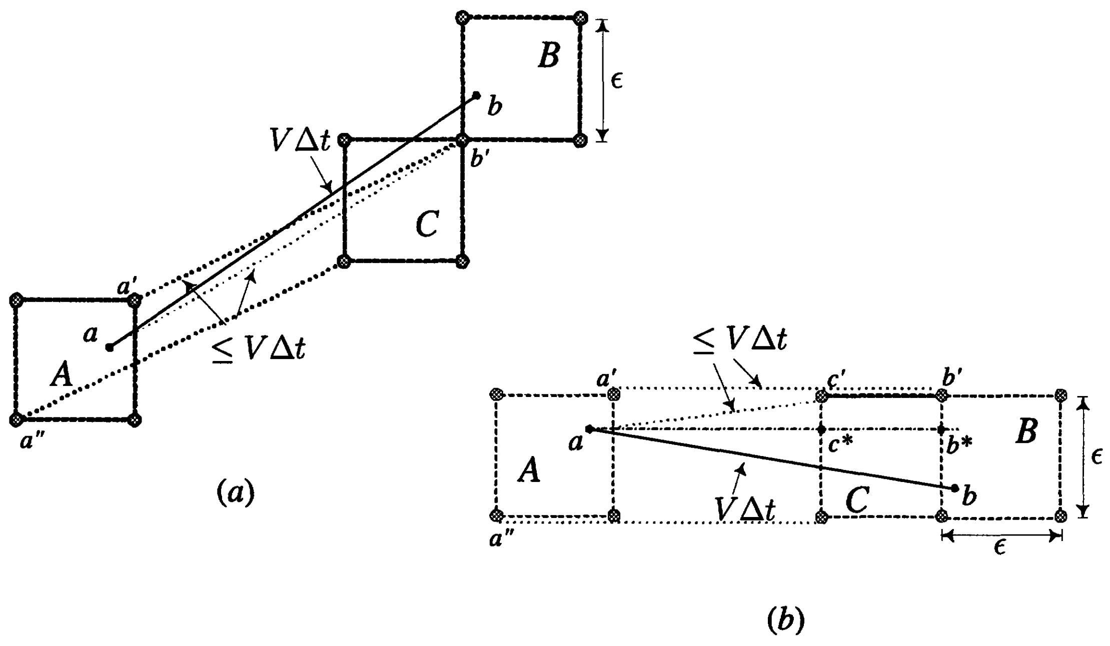

Figure 4-4: Illustration depicting the proof of Lemma 4.4.2. (a) Case where grid squares A and B are not co-linear (b) Case where grid squares A and B are co-linear

this is the case, then it must be that the top-right corner grid point in A, a', and the top-left corner grid point in B, b', are at least as close to each other as any other two points in A,B. Define C as the grid square immediately left of B (i.e. b' is one of two corner grid points shared by C and B). Thus we can conclude that $d[a',b'] \leq d[a,b] = V\Delta t$. Note that we can translate the line segment between a' and b' to show that, within distance $V\Delta t$, from any corner grid point $a'' \in A$, we can reach some corner grid point in C. Finally, let c' be the top-left corner grid point of C. We show that $d[a,c'] \leq V\Delta t$ as follows. First, we project the line segment between a and b onto the horizontal axis, and let $c^* \in C$ and $b^* \in B$ be the intersection points with the left sides of C and B respectively. Applying the triangle inequality, we have that,

$$d[a,c'] \leq d[a,c^*] + \epsilon = d[a,b^*] \leq d[a,b] \leq V\Delta t \qquad (4.5)$$

□

Lemma 4.4.3. *For any two points $a \in A$ and $b \in B$ such that A and B are* adjacent *grid squares, $d[a,b] \leq 2\sqrt{2}\epsilon$.*

Proof. The largest distance between any two points within the same grid square is $\sqrt{2}\epsilon$. Applying the triangle inequality, we obtain the desired result. $\square$

Proof of Theorem 4.4.1. Consider an optimal MBN path $Q_{OPT} = Q(0), Q(1), \dots, Q(K)$, where $Q(0) = M_0$. Without loss of generality, we assume the optimal MBN travels at maximum speed over the entire time horizon, i.e. $d[Q(t-1), Q(t)] = V\Delta t$, $\forall t = 1, \dots, K$. Intuitively, this follows from the observation that the slower the optimal MBN travels, the easier it becomes for the DPA MBN to close the gap with it over time.

Our aim is to construct a candidate path $M' = M'(0), M'(1), \dots, M'(K)$ that traverses from $M'(0) = M_0$ to $M'(K)$, consisting only of grid point locations. Let $d^{M'}_{max}(t)$ denote the distance from $M'(t)$ to the farthest RN at time step t, i.e. similar to the definition of $d^{opt}_{max}(t)$. Similarly, we define $d^{dpa}_{max}(t)$ with respect to the solution found by the DPA algorithm. By lemma 4.4.1, we know that the path found by the DPA algorithm will have at least as high MPP-MFPA objective value as that of M', i.e. $\frac{1}{K}\sum_{t=1}^{K} H[d^{dpa}_{max}(t)] \geq \frac{1}{K}\sum_{t=1}^{K} H[d^{M'}_{max}(t)]$. Thus we will proceed to lower bound the objective value of M' with respect to the optimal solution, i.e. we will show that $\frac{1}{K}\sum_{t=1}^{K} H[d^{M'}_{max}(t)] \geq \frac{1}{K}\sum_{t=1}^{K} H[d^{opt}_{max}(t) + 2\sqrt{2}t\epsilon]$. We will show this by *upper bounding* $d^{M'}_{max}(t)$, i.e. we will show that $d^{M'}_{max}(t) \leq d^{opt}_{max}(t) + 2\sqrt{2}t\epsilon$, $\forall t$.

We construct the candidate path M' as follows, starting from $M'(0) = Q(0)$ (not necessarily located at a grid point). From the *first part* of Lemma 4.4.2 we can set $M'(1)$ to lie at a grid point in a grid square at most one away from (i.e. adjacent to) the grid square containing $Q(1)$. Specifically, since $d[M'(0), Q(1)] = V\Delta t$, the lemma tells us that $d[M'(0), M'(1)] \leq V\Delta t$. Next, we can apply lemma 4.4.3 to show that $d[M'(1), Q(1)] \leq 2\sqrt{2}\epsilon$. Applying the triangle inequality, we therefore have that $d^{M'}_{max}(1) \leq d^{opt}_{max}(1) + 2\sqrt{2}\epsilon$.

Similarly, starting from $M'(1)$ (located at a grid point), we can apply the *second part* of Lemma 4.4.2 and set $M'(2)$ to lie at a grid point in a grid square at most two away from the grid square containing $Q(2)$. We do this as follows, where we let A and B denote the grid squares containing $Q(1)$ and $Q(2)$ respectively. By the lemma, we have that since $d[Q(1), Q(2)] = V\Delta t$, that from any grid point $a \in A$ we can reach a grid point $c \in C$ such that $d[a, c] \leq V\Delta t$, where C denotes a grid square adjacent to B. Finally, we can translate the line segment between a and c such that a is shifted to $M'(1)$. We define $M'(2)$ as the resultant shifted (grid point) location of c. Since $M'(1)$ is located in a grid square adjacent to A, we can conclude that $M'(2)$ is in a grid square at most two away from B. Applying lemma 4.4.3 and the triangle inequality, we therefore have that $d^{M'}_{max}(2) \leq d^{opt}_{max}(2) + 2 \cdot (2\sqrt{2}\epsilon)$.

We can repeat this application of the second part of Lemma 4.4.2 for an arbitrary time step t to obtain the desired result. $\square$

Since the above Theorem simply shows a lower bound on the MPP-MFPA objective achievable by the DPA algorithm, we do not know if this is a true reflection of the worst case performance of the algorithm. Yet, as reflected in the following Theorem, it turns out the lower bound is tight in the sense that the difference between $d^{dpa}_{max}(t)$ and $d^{opt}_{max}(t)$ can potentially increase without bound as a function of the number of time steps (i.e. assuming fixed Δt).

Theorem 4.4.2. *For an unbounded plane, there exists a worst case problem instance in which* $\lim_{t \to \infty} \{d^{dpa}_{max}(t) - d^{opt}_{max}(t)\} = \infty$.

Proof. We illustrate two worst case situations that result in the performance bound of the theorem in Fig. 4-5. Both examples involve a single RN p travelling in a single direction with speed V, and $M_0 = p(0)$. Thus the optimal solution in both instances involves the MBN following the exact same trajectory as the RN, travelling a distance $V\Delta t$ in each time step. This yields $d^{opt}_{max}(t) = 0$, $\forall t$. Yet, starting from M_0 the DPA solution can only travel between grid points and therefore cannot exactly follow an arbitrary trajectory. Thus to prove the theorem we need only provide an RN trajectory that results in the DPA MBN path being limited to travel (in each time step) a constant distance strictly less than $V\Delta t$ *in the direction of the optimal MBN* (i.e. projected onto its trajectory). Since this difference accumulates linearly with each time step, we obtain the result of the theorem.

We first assume that $V\Delta t$ is not integer divisible by ϵ, and consider a horizontal straight-line RN trajectory, illustrated in Fig. 4-5-a. Due to the non-divisibility

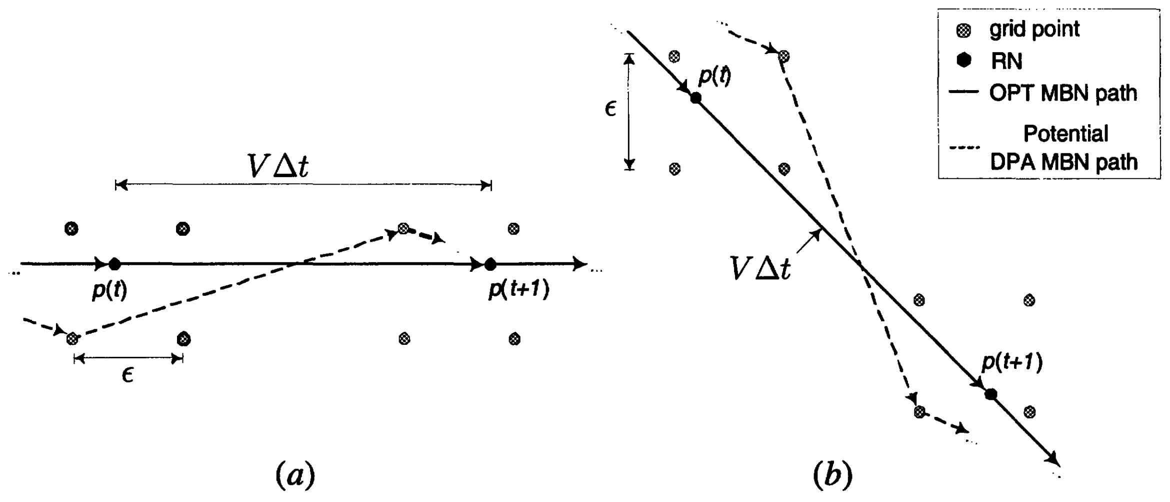

Figure 4-5: Illustration depicting the proof of Theorem 4.4.1. (a) Worst case scenario when $V\Delta t$ is not integer divisible by ϵ. (b) Worst case scenario when $V\Delta t$ is integer divisible by ϵ.

assumption, no two horizontally separated pair of grid points can be exactly $V\Delta t$ apart.

Next, assume $V\Delta t$ is integer divisible by ϵ and consider a diagonal (e.g. $-45°$) trajectory, illustrated in Fig. 4-5-b. Now, because of the integer divisibility assumption, all diagonally separated grid points are some integer times $\sqrt{2}\epsilon$ apart. Thus since $\sqrt{2}$ is irrational, it follows that no two diagonally separated grid points can be exactly a distance $V\Delta t$ apart. □

It should be noted that Theorems 4.4.2 and 4.4.1 are in the context of an unbounded planar area. Yet, the practical scenarios we are interested in involve a bounded area, in which the worst case result of Theorem 4.4.2 would not hold. In such scenarios, it is likely that the difference between the DPA and optimal solutions would be capped by a function involving the area's dimensions.

4.5 Greedy Approach to the MPP-MFPA problem

In this section we develop a greedy approach towards solving the MPP-MFPA problem. As mentioned in section 4.3, for certain problem instances the performance of such an approach can be quite poor in the worst case. Yet, as will be shown via simulation in section 4.9, for many problem instances the approach performs quite reasonably. More importantly, the approach is significantly less computationally complex than the DPA algorithm described in section 4.4.

The most natural greedy approach to a multi-step optimization problem aims to optimize the 1-step instantaneous problem at each time step. This is the approach we take in this section, and present a high level algorithm below.

Algorithm 11 High Level MPP-MFPA Greedy Algorithm

```
1: Initialize $M(0) = M_0$
2: for t=1,2,...,K do
3:     Compute the location for $M(t)$ that maximizes $H[d_{max}(t+1)]$, subject to
       $d[M(t-1), M(t)] \leq V\Delta t$
4: return $M^* = M(0), M(1), \ldots, M(K)$
```

The key step in the above high level algorithm is the solution of the 1-step optimization problem in line 3. A more complete formulation of this problem is as follows.

Problem 1-step MPP-MFPA: Given the RN positions at time $t+1$, $p_i(t+1), \forall i \in P$ and previous MBN position, $M(t)$. Calculate the optimal MBN position at time $t+1$, $M(t+1)$, such that the MFPA throughput metric at time $t+1$ is maximized, subject to the maximum MBN velocity bounded by V. Mathematically, the 1-step MPP-MFPA problem is expressed as,

$$\max_{M(t+1)} \quad H[d_{max}(t+1)] \tag{4.6}$$

$$s.t. \quad d[M(t), M(t+1)] \leq V\Delta t \tag{4.7}$$

With the above formulation, we note that the 1-step MPP-MFPA problem can be viewed as a *constrained 1-center problem.* Specifically, since $H()$ is a decreasing function in $d_{max}(t+1)$, minimizing $d_{max}(t+1)$ will maximize the objective function in (4.6). If not for the velocity constraint in (4.7), the problem would reduce to finding the unconstrained 1-center, for which several efficient polynomial time algorithms

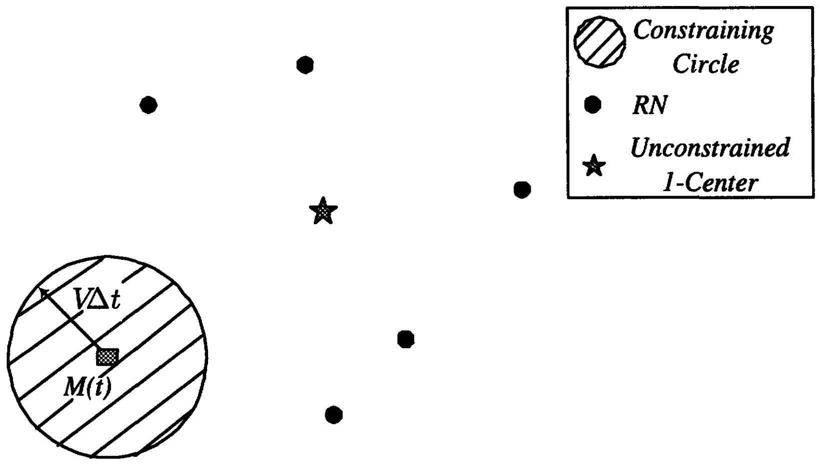

Figure 4-6: Illustration of constrained 1-center instance in which the unconstrained 1-center is outside the constraining circle.

exist [2]. Yet, with the constraint in mind we can view the problem as one in which we need to to find the 1-center of the RNs at time $t + 1$ such that it lies within a circle of radius $V\Delta t$ around $M(t)$. This is depicted in Fig. 4-6.

The general convex polygon constrained 1-center problem has been previously addressed in [15]. However, the algorithm from [15] cannot be directly adapted due to the fact that it requires considering each polygon vertex separately. In our problem, we are constrained by a circle, which cannot really be viewed as a polygon (and even if it could, it would have an undefined or infinite number of vertices). In the next section we develop a simple algorithm to solve the circular constrained 1-center problem.

4.5.1 Circular Constrained 1-Center (CC-1C) Algorithm

We begin with the following observation, which provides the first step in our algorithm to solve the circular constrained 1-center problem. Let C denote the constraining circle of radius $V\Delta t$ with center $M(t)$

Observation 4.5.1. *If the solution to the unconstrained 1-center problem lies within the circle C, then this is the solution to the constrained 1-center problem.*

Thus the main difficulty lies in solving the constrained problem *when the unconstrained solution lies outside C* (e.g. shown in Fig. 4-6). The following lemma provides the first key to solving this problem, where we have defined δC as the boundary of the circle C.

Lemma 4.5.1. *Assume the solution to the unconstrained 1-center problem lies outside the circle C. Then, the solution to the constrained 1-center problem must lie on δC.*

Proof. By the previous discussion, the solution to the constrained 1-center problem involves minimizing $d_{max}(t+1)$ subject to the circular constraint. The proof involves first showing that $d_{max}(t+1)$ is *convex* in $M(t+1) \triangleq [M_x(t+1), M_y(t+1)]$. This will allow us to conclude that from a given MBN placement at $M(t+1)$, changing the solution along the gradient direction $\nabla d_{max}(t+1)$ will decrease $d_{max}(t+1)$. Note that while $d_{max}(t)$ is not differentiable at certain points, directional derivatives exist everywhere [56].

Next, we assume that the circular constrained optimal $M(t+1)$, denoted $M^*(t+1)$, is an *interior* point of C. However, if this were the case, then there must exist another location $M'(t+1)$ along the direction $\nabla d_{max}(t+1)$ such that $M'(t+1)$ is within C (i.e. either also interior to C or on δC). Thus $M'(t+1)$ must yield a lower value of $d_{max}(t+1)$, contradicting the optimality of $M^*(t+1)$.

To show $d_{max}(t+1)$ is convex in $[M_x(t+1), M_y(t+1)]$ consider its full expansion. We have removed the $(t+1)$ dependence for legibility.

$$\begin{aligned} d_{max} &= \max_{i \in P} d_i \\ &= \max_{i \in P} \left\{ \sqrt{[M_x - p_{i_x}]^2 + [M_y - p_{i_y}]^2} \right\} \end{aligned} \tag{4.8}$$

A common fact from optimization theory is that a function that is a maximum of a set of convex functions is also convex. Since the Euclidean distance function $d_i()$ is convex [56], the result follows. □

Lemma 4.5.1 allows us to restrict our search for $M(t+1)$ to the locus of points defined by δC. We define the *Constrained Minimum Spanning Circle* $(CMSC)$ for the RNs at time $t+1$, as the circle with center at the optimal location of $M(t+1)$

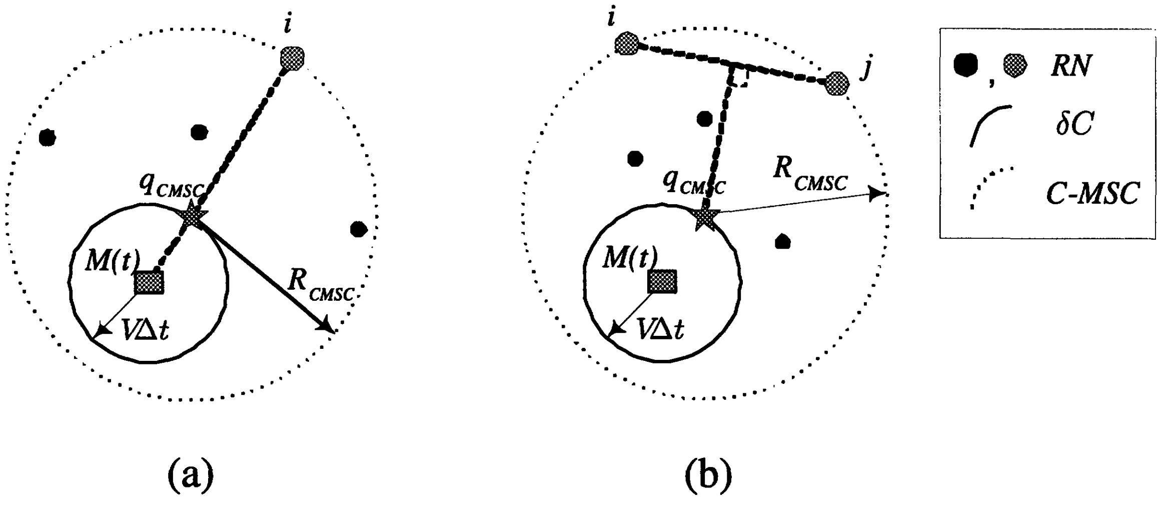

Figure 4-7: Illustration of the two unique ways the constrained 1-center can be defined. (a) By a single RN. (b) By a pair of RNs.

and radius equal to the corresponding value of $d_{max}(t+1)$. We denote the center and radius of the $CMSC$ as q_{CMSC} and R_{CMSC} respectively. Consider the following lemma regarding the $CMSC$, illustrated in Fig. 4-7.

Lemma 4.5.2. *Assume the unconstrained 1-center is outside the constraining circle C. Then there are are two unique ways the $CMSC$ can be defined.*

1. *By a single RN $i \in P$. If this is the case, then q_{CMSC} is located at the* first *intersection between δC and the directed line segment $\overrightarrow{iM(t)}$. R_{CMSC} is equal to $d(q_{CMSC}, i)$.*

2. *By a pair of RNs $i, j \in P$. If this is the case, then q_{CMSC} is located at an intersection point between δC and the perpendicular bisector of i and j. The intersection point is chosen to minimize $R_{CMSC} = d(q_{CMSC}, i)$.*

Proof. To prove the Lemma, first recall that by Lemma 4.5.1, the optimal q_{CMSC} must lie on δC. We now go through several cases regarding the farthest RN(s) from q_{CMSC}. First assume exactly one RN is farthest from q_{CMSC}. In this case, in order to minimize R_{CMSC} subject to the constraint that $q_{CMSC} \in \delta C$ it is a simple geometric fact that q_{CMSC} and R_{CMSC} are defined as in the first part of the Lemma. The same holds true with respect to the second part of the Lemma if we assume that exactly two RNs are simultaneously farthest from q_{CMSC}. Finally, consider the case in which exactly $k \geq 3$ RNs are simultaneously farthest. In this case, we have a situation in which *all pairs* of the k farthest RNs must be equidistant from the center. Yet, a pair of equidistant RNs coupled with the constraint that the center must be on δC uniquely determines a center location (e.g. as per the second part of the Lemma). Therefore, $k \geq 3$ simultaneously farthest RNs represents an over-determined situation, wherein the corresponding q_{CMSC}, R_{CMSC} tuple would have been considered under the second part of the Lemma. □

Lemma 4.5.2 can be thought of as the constrained analog of Fact 3.4.1 from Chapter 3. However, it should be noted that in Fact 3.4.1 we had to consider triplets of farthest RNs, in addition to singles and doubles. The difference is due to the circle boundary constraint brought upon by Lemma 4.5.1. Indeed, Lemma 4.5.1 can be thought of as reducing the degree of freedom of the problem from 3 equidistant RNs uniquely defining a center, to just 2 RNs. Below we present the Circular Constrained 1-Center (CC-1C) algorithm, which finds the finds the constrained 1-center.

Algorithm 12 CC-1C Algorithm

```
1: Compute the unconstrained 1-center location, $q_{UC}$, using an algorithm from [2]
2: if $q_{UC}$ is within $C$ then
3:    return $M(t+1) = q_{UC}$
4: Set $R_{min} = \infty$
5: for all single RNs $i \in P$ do
6:    Let $q$ be the first intersection point between the line segment $\overrightarrow{iM(t)}$ and $\delta C$.
7:    Let $R_q$ be the distance between $q$ and $i$, $R_q = d(q,i)$
8:    if $d(q,j) \leq R_q, \forall j \in P$ and $R_q < R_{min}$ then
9:       Set $M(t+1) = q$ and $R_{min} = R_q$
10: for all pairs of RNs $i, j \in P$ do
11:    if the perpendicular bisector of $i, j$ and $\delta C$ intersect then
12:       Let $q$ be the intersection point between the perpendicular bisector of $i, j$ and
          $\delta C$ that yields the lowest value of $d(q,i)$.
13:       Let $R_q$ be the distance between $q$ and $i$, $R_q = d(q,i)$
14:       if $d(q,k) \leq R_q, \forall k \in P$ and $R_q < R_{min}$ then
15:          Set $M(t+1) = q$ and $R_{min} = R_q$
16: return $M(t+1)$
```

The algorithm works by directly applying the constructive implications of the previous discussion. Specifically, we start by checking whether the condition outlined in observation 4.5.1 holds. Assuming it does not, we next check all possible q_{CMSC}, R_{CMSC} tuples as outlined in Lemma 4.5.2 to see if they define a valid CMSC (i.e. if they cover all the RNs). The valid CMSC with minimum radius is taken as the overall solution.

The computational complexity of the CC-1C algorithm is $O(N^2)$, where N is the number of RNs. This is because the for loop in line 10 considers all pairs of RNs, and thus results in the most complex operation. The solution of the unconstrained problem (line 1) can be found with $O(N \log N)$ computational complexity [2],[51]. Note that the complexity would have been $O(N^3)$ if we had to consider triplets of RNs. Thus the result of Lemma 4.5.1 yields considerable complexity savings.

As a final point regarding the overall greedy algorithm, consider a problem instance in which the unconstrained 1-center locations at consecutive time steps are always within a distance $V\Delta t$ of each other. Also assume that M_0 is within $V\Delta t$ the unconstrained 1-center location in the first time step. If this is the case, then by the presence of line 1 in the CC-1C algorithm, the greedy approach will find the exact optimal solution. By contrast, the DPA algorithm would still only find an approximate solution, since the MBN placements would be restricted to grid points.

4.6 Discrete Formulation with Relaxed Velocity Constraint

In this section we propose a variation of the base discrete MPP-MFPA formulation given in (4.1). In particular, we relax the hard constraint on the MBN velocity, instead adapting a penalty function $G()$ on the MBN movements. This type of formulation can model situations in which we would like to discourage large MBN movements, but allow such movements if there is sufficient throughput gain to be had. The modified formulation is given below.

Problem Relaxed MPP-MFPA: Given the RN trajectories $p_i(t), \forall i \in P, t = 0, \ldots, K$ and initial MBN position $M(0) = M_0$. Calculate the MBN path $M^* = M(0), M(1), \ldots, M(K)$ such that the average MFPA throughput metric plus a velocity cost is maximized. Mathematically, the Relaxed MPP-MFPA problem is expressed as,

$$\max_{M^*} \quad \frac{1}{K} \sum_{t=1}^{K} \left\{ H[d_{max}(t)] + G\left(\frac{d[M(t-1), M(t)]}{\Delta t} \right) \right\} \tag{4.9}$$

$$s.t. \qquad M(0) = M_0 \tag{4.10}$$

where $G()$ is a *decreasing* function in its argument. By forcing $G()$ to decrease with increasing value of $\frac{d[M(t-1),M(t)]}{\Delta t}$, this penalizes MBN paths with large movements.

To solve the above formulation, we attempt to leverage the discussion thus far. Indeed, the DPA algorithm from section 4.4 can be modified in a simple manner to solve the Relaxed MPP-MFPA problem. Specifically, we need to apply the following modifications:

- Initialize $q_v(1) = H[\max_i\{d(v, p_i(1))\}] + G\left(\frac{d[M_0, v]}{\Delta t}\right)$, $\forall v \in Y(1)$ (line 3)
- Allow MBN movements between *any two* grid points (i.e. remove the distance restriction from line 7)
- Replace $H[\max_i\{d(v, p_i(1))\}]$ with $H[\max_i\{d(v, p_i(1))\}] + G\left(\frac{d[u, v]}{\Delta t}\right)$ throughout the Metric Update phase (i.e. lines 8, 9)

By arguments along the same lines as those proposed in section 4.4, we can conclude that the DPA algorithm with the above modifications will solve the Relaxed MPP-MFPA problem. However, the analysis of section 4.4.1 does not directly apply. It is likely, however, that similar results can be shown with a similar analysis technique. It is important to note that this modified DPA is considerably more computationally complex than the original DPA. In particular, since MBNs can travel between any two grid points in a time step the resulting complexity is $O\big(|Y(1)|^2 \cdot K\big)$

for the modified DPA, as opposed to $O\big(|Y(1)| \cdot (\lceil \frac{2V\Delta t}{\epsilon} \rceil)^2 \cdot K\big)$ for the original DPA. If the maximum MBN speed V is not too large, the difference can be quite significant.

We next attempt to leverage the Greedy Approach towards solving the Relaxed MPP-MFPA problem. To this end, we can apply the same high-level algorithm given in section 4.5. We next turn our attention to the single step solution. Thus we formulate the 1-step Relaxed MPP-MFPA problem in a vein similar the original 1-step MPP-MFPA problem in section 4.5 below.

Problem 1-step Relaxed MPP-MFPA: Given the RN positions at time $t+1$, $p_i(t+1), \forall i \in P$ and previous MBN position $M(t)$. Calculate the optimal MBN position at time $t+1$, $M(t+1)$ such that the Relaxed MPP-MFPA throughput metric at time $t+1$ is maximized. Mathematically, the 1-step Relaxed MPP-MFPA problem is expressed as,

$$\max_{M(t+1)} H[d_{max}(t+1)] + G\left(\frac{d[M(t), M(t+1)]}{\Delta t}\right) \tag{4.11}$$

We begin by observing that, *given* a value of $d[M(t), M(t+1)]$, this fixes the second term of (4.11). Therefore, we can use the CC-1C algorithm from section 4.5.1 to maximize the first term of (4.11). Next, we observe that $d_{max}(t+1)$ is monotonically decreasing with increasing values of $d[M(t), M(t+1)]$ in the range $[0, d^*]$, where $d^* = d[M(t), M^*_{UC}(t+1)]$ is the distance between $M(t)$ and the *unconstrained* 1-center of the RNs at time $t+1$, $M^*_{UC}(t+1)$. Clearly we would never have the MBN travel farther than d^*, since this simply decreases $G[d()]$ while having no effect on $H[d()]$.

The above observations motivate the following basic heuristic for the 1-step Relaxed 1-center problem, where k represents a parameter that trades off complexity vs. performance. Determine the unconstrained 1-center $M^*_{UC}(t+1)$, and set $d^* = d[M(t), M^*_{UC}(t+1)]$. Solve the circular constrained 1-center problem by utilizing the CC-1C algorithm for $k+1$ circles of radii $[0, d^*/k, 2d^*/k, \ldots, (k-1)d^*, d^*]$. Set $M(t+1)$ to equal the location which yields the maximum value of $H[\max_i d[M(t+$

$1), p_i(t+1)]] + G(\frac{d[M(t),M(t+1)]}{\Delta t})$.

Obtaining the optimal solution for the 1-step relaxed MPP-MFPA problem appears somewhat more complicated than for the base 1-step MPP-MFPA problem. The reason is that instead of being able to minimize $d_{max}(t)$ as a proxy for maximizing the overall objective function, for the relaxed 1-step relaxed MPP-MFPA problem there does not appear to be an obvious way to get around having to directly optimizing the entire objective function. This likely requires assuming specific properties for $H()$ and $G()$ and we do not pursue this in this section.

4.7 Continuous Problem Formulation

Up to this point, we have discretized time, and for the DPA algorithm we discretized space as well. Doing this allowed us to develop algorithms that obtain good solutions to the MPP-MFPA problem. Yet, an alternative approach is from a continuous perspective. We develop such a formulation below, as the Continuous MPP-MFPA problem. Throughout the section, we use bold font to denote a vector (as opposed to a scalar) quantity.

Problem C-MPP-MFPA: Given the RN trajectories $p_i(t), \forall i \in P, 0 \le t \le T$ and initial MBN position $\boldsymbol{M}(0) = M_0$. Calculate the optimal MBN path $\boldsymbol{M}(t), 0 \le t \le T$ such that the time average MFPA throughput metric is maximized, subject to the maximum MBN velocity bounded by V. Mathematically, the C-MPP-MFPA problem is expressed as,

$$\max_{\boldsymbol{M}(t)} \quad \frac{1}{T}\int_0^T H[d_{max}(t)] \tag{4.12}$$

$$s.t. \quad |\dot{\boldsymbol{M}}(t)| \le V, \quad 0 \le t \le T \tag{4.13}$$

$$\boldsymbol{M}(0) = M_0 \tag{4.14}$$

where $\dot{\boldsymbol{M}}(t)$ denotes the time derivative of $\boldsymbol{M}(t)$. The above formulation of the MPP-MFPA can be seen as the direct continuous analog of the discrete-time formulation given in section 4.2 (e.g. take $\Delta t \to 0$). We start by noting that we can adapt the above formulation to that of an *optimal control problem.* Specifically, we can let $\boldsymbol{M}(t)$ represent the system state and $\boldsymbol{u}(t) = \dot{\boldsymbol{M}}(t)$ the control. We can thus re-pose the problem in standard form as an optimal control problem with bounded control [18] as follows.

$$\begin{aligned}
\max_{\boldsymbol{M}(t)} \quad & \tfrac{1}{T}\int_0^T H[d_{max}(t)] && (\textit{objective}) && (4.15)\\
s.t. \quad & \dot{\boldsymbol{M}}(t) = u(t) && (\textit{system dynamics}) && (4.16)\\
& |\boldsymbol{u}(t)| \leq V && (\textit{control variable constraints}) && (4.17)\\
& \boldsymbol{M}(0) = M_0 && (\textit{initial condition}) && (4.18)
\end{aligned}$$

where we note the fact that $\boldsymbol{u}(t) = [u_x(t)\ u_y(t)]^T$ and $\boldsymbol{M}(t) = [M_x(t)\ M_y(t)]$ are 2-dimensional vectors of functions. The goal is to solve for the optimal control function $\boldsymbol{u}(t)$ and in turn determine the optimal state trajectory $\boldsymbol{M}(t)$. To do this, we follow the method prescribed in [18],[7] based on Calculus of Variations theory. Specifically, we start by forming the variational Hamiltonian, J, which in the optimal solution is maximized.

$$J = H[d_{max}(t)] + \boldsymbol{\lambda}(t)^T \cdot \boldsymbol{u}(t) \tag{4.19}$$

where $\boldsymbol{\lambda}(t) = [\lambda_x(t)\lambda_y(t)]^T$ is a lagrange multiplier vector. From the form of J, we can immediately conclude that the $\boldsymbol{u}(t)$ that maximizes J for $\boldsymbol{\lambda}(t) \neq 0$ is,

$$\boldsymbol{u}(t) = \frac{V}{\sqrt{\lambda_x(t)^2 + \lambda_y(t)^2}}\boldsymbol{\lambda}(t) \tag{4.20}$$

Geometrically, this can be viewed as a vector of length V in a direction co-linear with $\boldsymbol{\lambda}(t)$. Next, we utilize the Euler-Lagrange equation $\dot{\boldsymbol{\lambda}}(t) = -\frac{\partial J}{\partial \boldsymbol{M}}$ to obtain the following relation,

$$\dot{\boldsymbol{\lambda}}(t) = -\begin{bmatrix} \frac{\partial H}{\partial M_x} \\ \frac{\partial H}{\partial M_y} \end{bmatrix} \tag{4.21}$$

The above equation gives us the key information needed to solve for $\boldsymbol{\lambda}(t)$. To proceed further, we will assume a specific functional form for $H()$. Specifically, we will consider the CDMA Throughput function from (3.2), i.e.,

$$H_{cdma}[d_{max}(t)] = \frac{1}{d_{max}(t)^{\alpha} + b} \tag{4.22}$$

where $\alpha \geq 2$ represents the path loss exponent and b a positive constant. For simplicity we have ignored all other multiplicative constants. Next, for the purposes of illustration we assume just a single RN p, e.g.,

$$d_{max}(t) = d(t) = \sqrt{[M_x(t) - p_x(t)]^2 + [M_y(t) - p_y(t)]^2} \tag{4.23}$$

This assumption allows us to temporarily do away with the complexities (e.g. non-differentiability at certain points) associated with the function $d_{max}(t)$. In the next section, we discuss how to deal with more than one RN. With this assumption, the Euler-Lagrange equations from (4.21) yield,

$$\begin{bmatrix} \dot{\lambda}_x(t) \\ \dot{\lambda}_y(t) \end{bmatrix} = \frac{\alpha d(t)^{\alpha-2}}{[d(t)^{\alpha} + b]^2} \begin{bmatrix} M_x(t) - p_x(t) \\ M_y(t) - p_y(t) \end{bmatrix} \tag{4.24}$$

At this point, there are two cases: a singular ($\boldsymbol{\lambda}(t) = \mathbf{0}$) or normal ($\boldsymbol{\lambda}(t) \neq \mathbf{0}$) arc/interval. The normal case was discussed earlier, yielding (4.20) when $\boldsymbol{\lambda}(t) \neq \mathbf{0}$. For the singular arc case, suppose there $\boldsymbol{\lambda}(t) = \mathbf{0}$ for some time interval $[t_1, t_2]$. Thus it must be that $\dot{\boldsymbol{\lambda}}(t) = \mathbf{0}, \forall t \in (t_1, t_2)$. Then by (4.24) and (4.16), it must be the case that,

$$\begin{bmatrix} M_x(t) \\ M_y(t) \end{bmatrix} = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} \quad and \quad \begin{bmatrix} u_x(t) \\ u_y(t) \end{bmatrix} = \begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \end{bmatrix} \tag{4.25}$$

Looking at (4.20) and (4.25), we can fully characterize the control $u(t)$ in terms of the lagrange multiplier function $\lambda(t)$, i.e.,

$$\boldsymbol{u}(t) = \begin{cases} \frac{V}{\sqrt{\lambda_x(t)^2+\lambda_y(t)^2}}\boldsymbol{\lambda}(t), & \boldsymbol{\lambda}(t) \neq \mathbf{0} \\ \dot{\boldsymbol{p}}(t), & \boldsymbol{\lambda}(t) = \mathbf{0} \end{cases} \tag{4.26}$$

The above gives a very interesting insight into the optimal MBN path $\boldsymbol{M}(t)$ for the single RN case. In particular, from (4.25) we have that either (i) $\boldsymbol{M}(t)$ is coincident with $\boldsymbol{p}(t)$ and is staying coincident with $\boldsymbol{p}(t)$ by travelling with velocity $\boldsymbol{u}(t) = \dot{\boldsymbol{p}}(t)$ (singular arc) , or (ii) $\boldsymbol{M}(t)$ is away from $\boldsymbol{p}(t)$ (or just instantaneously coincident with $\boldsymbol{p}(t)$), and is travelling at maximum velocity V in some direction. Thus the conditions tell us that if $\boldsymbol{M}(t)$ is away from $\boldsymbol{p}(t)$, *it will never travel with velocity slower than* V. Note however, that the conditions do not mean that if $\boldsymbol{M}(t)$ is coincident with $\boldsymbol{p}(t)$, that it must stay coincident (i.e. enter a singular arc). Indeed, this uncertainty is one of the reasons that such *singular control problems* are difficult to deal with. We give an example of such a situation in Fig. 4-8. In the example, at time t^*, the MBN can stay coincident with the RN (i.e. stay in the singular arc) since the RN is moving with just velocity V. Yet, if it does so then starting at time t^{**} (at which time the RN has sudden jump in velocity) it will lag the RN until the end of the time horizon. By contrast, the optimal path diverges from the RN path at time t^*, but rejoins it at time t^{**}. This example can be thought of as the continuous analog to the discrete example in Fig. 4-2.

We complete the characterization of the optimal solution by gathering the following system of equations. These equations must be solved to determine the optimal $\lambda(t)$ and thus the optimal $u(t)$ and $M(t)$.

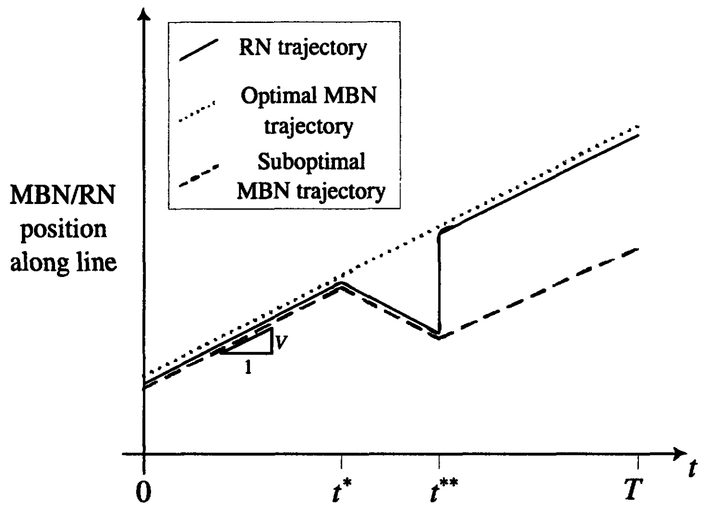

Figure 4-8: Single RN, Single MBN, 1-D continuous time MPP example.

$$\begin{bmatrix} \dot{\lambda}_x(t) \\ \dot{\lambda}_y(t) \end{bmatrix} = \frac{\alpha d(t)^{\alpha-2}}{[d(t)^\alpha + b]^2} \begin{bmatrix} M_x(t) - p_x(t) \\ M_y(t) - p_y(t) \end{bmatrix} \quad \text{(\textit{E-L Equation})} \quad (4.27)$$

$$\boldsymbol{u}(t) = \begin{cases} \frac{V}{\sqrt{\lambda_x(t)^2+\lambda_y(t)^2}}\boldsymbol{\lambda}(t), & \boldsymbol{\lambda}(t) \neq \mathbf{0} \\ \dot{\boldsymbol{p}}(t), & \boldsymbol{\lambda}(t) = \mathbf{0} \end{cases} \quad \text{(\textit{Control Equations})} \quad (4.28)$$

$$\dot{\boldsymbol{M}}(t) = \boldsymbol{u}(t) \quad \text{(\textit{System Dynamics})} \quad (4.29)$$

$$\boldsymbol{M}(0) = M_0 \quad \text{(\textit{Initial state})} \quad (4.30)$$

$$\boldsymbol{\lambda}(T) = 0 \quad \text{(\textit{Final Condition})} \quad (4.31)$$

The final condition in (4.31) is due to the form of the objective function in (4.16). The above system of differential equations constitutes what is known as a *Two-Point Boundary Value Problem* (TPBVP). In contrast to an Initial Value Problem (IVP), where all conditions are specified at the initial time, a TPBVP splits the conditions at both the initial and final times. Because of this, obtaining the numerical solution to TPBVPs (closed form results are quite rare) is considerably more difficult than that of IVPs. A common method for numerically solving TPBVP's is known as the *Shooting Method*, e.g. [50].

Further complicating matters is the potential presence of singular arcs in the solution, which has a direct influence on the form of $u(t)$. Furthermore, since $\lambda(t)$

and $u(t)$ are co-dependent on each other, obtaining a solution is greatly aided if we have a-priori knowledge of the sequence of normal and singular arcs. For this the DPA algorithm discussed in section 4.4 can be utilized. In particular, for a given problem instance we can first employ the DPA algorithm with a relatively coarse grid spacing. From the resulting solution, we can qualitatively glean the normal-singular arc sequence. Knowing this, we can execute a shooting method with fine step size and obtain close to an exact optimal MBN trajectory. Note that the reason we cannot simply run the DPA with fine gridding is its high computational complexity which is a function of the grid spacing. This was discussed in greater detail in section 4.4. By contrast, the computational complexity of the shooting method is approximately linear in the number of integration steps[3]. As a final point, it is important to note that the discrete methodology utilized by the DPA is an independent methodology from the TVBVP solution utilized by the continuous methodology.

4.8 Extensions to the Continuous Formulation

The first obvious extension to the continuous formulation discussed in the previous section is how to deal with multiple RNs. As mentioned earlier, the main difficulty has to do with the form of $d_{max}(t) = \max_i d[\boldsymbol{M}(t), \boldsymbol{p}_i(t)]$, which is non-differentiable for certain values of $\boldsymbol{M}(t) \triangleq (M_x(t), M_y(t))$. Thus a relatively straightforward way to deal with this is to use the function $d_{est-max}(t) = \sum_i (d[\boldsymbol{M}(t), \boldsymbol{p}_i(t)])^h$, to act as an estimate for $d_{max}(t)$. The intuition is that for a large enough value of h, the maximum $(d[M(t), p_i(t)])^h$ will dominate the sum (e.g. similar to the L_∞ metric).

The next extension is to relax the hard velocity constraint, as we did for the discrete problem in section 4.6. Thus, we can formulate (but do not solve) the continuous analog of the relaxed MPP-MFPA problem.

Problem Relaxed C-MPP-MFPA: Given the RN trajectories $p_i(t), \forall i \in P, 0 \leq t \leq T]$ and initial MBN position $M(0) = M_0$. Calculate the optimal MBN path

[3]This assumes that the correct starting conditions are guessed. These starting conditions can be determined by a relatively fast iterative gradient or bisection search as part of the shooting method.

$M(t), 0 \leq t \leq T$ such that the time average MFPA throughput metric less a penalty imposed on the MBN velocity is maximized. Mathematically, the C-MPP-MFPA problem is expressed as,

$$\max_{M(t)} \quad \frac{1}{T}\int_0^T \left\{ H[d_{max}(t)] + G\Big(\mid \dot{M}(t) \mid \Big) \right\} dt \tag{4.32}$$

where $G()$ is as defined in section 4.6.

As a final point regarding the continuous formulation discussion, it important to recognize that the optimal control solution we have employed are *necessary conditions* for stationarity. Indeed, these are not guaranteed to be the globally optimal solutions. To guarantee global optimality, one must examine the second order conditions which are generally quite complicated and do not yield much additional intuition. Yet, we can be confident that the solution derived in the previous section *for the assumed functional forms* are close to if not globally optimal. As we will show via simulation in the next section, the continuous solution outperforms the DPA algorithm, which tends to the optimal discrete solution for small grid spacing and fixed time horizon (i.e. Theorem 4.4.1). Furthermore, as the time step spacing gets smaller, the optimal discrete solution converges to the optimal continuous solution. In general, however, global optimality is not the case and for many functional forms the necessary conditions of the optimal control solution can yield the variational calculus equivalent of a saddle or minimum extreme point.

4.9 Simulation Results

In this section we present simulation results comparing the various approaches developed in this chapter. To this end, we begin with a simple single RN, 1-dimensional MPP-MFPA example, illustrated in Figs. 4-9 and 4-10. Both plots show an RN travelling on a line with position function $p(t) = t^4 e^{-t} + t$, $t \in [0, 10]$. This function was chosen somewhat arbitrarily, though it serves to reveal several insights into the workings of the various algorithms. We assume the MBN starts at position $\boldsymbol{M}_0 = (0.1, 0)$, its speed is bounded by $V = 2$, and that the CDMA throughput function from (4.22)

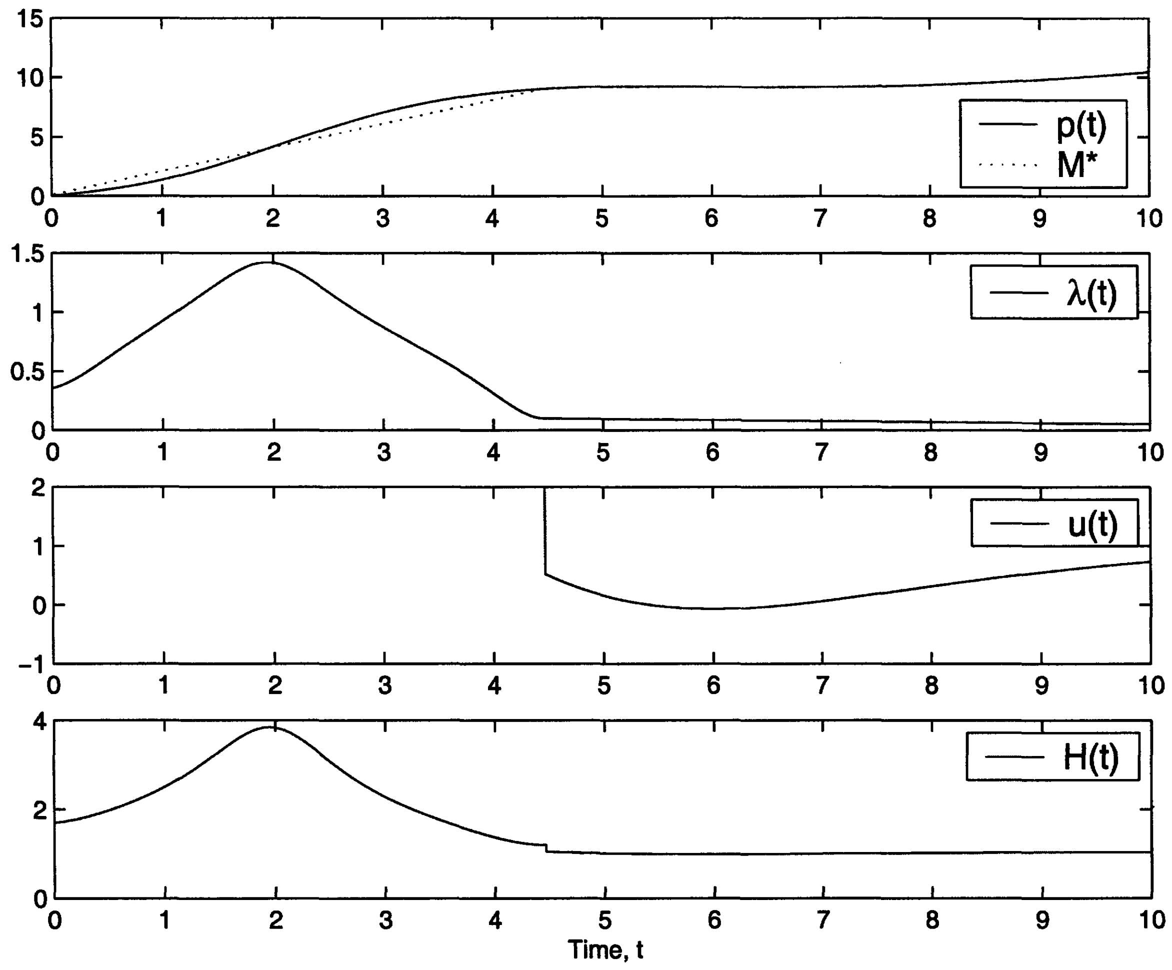

Figure 4-9: MPP-MFPA Single RN instance travelling on a line with position function $p(t) = t^4e^{-t} + t$ with Optimal Control Solution. The MBN velocity is bounded by $V = 2$, and the CDMA throughput function is used with $\alpha = 2$, $b = 1$. Also plotted are the corresponding Lagrange Multiplier function $\lambda(t)$, Control Function $u(t)$ and Variational Hamiltonian $Y(t)$.

is used with parameters $\alpha = 2$, $b = 1$.

Fig. 4-9 shows the detailed optimal control solution, found by using the shooting method to solve the Two-Point Boundary Value Problem represented by (4.27)-(4.31). As can be seen, the solution consists of a normal arc, followed by a singular arc. Specifically, in the normal arc (i.e. $\lambda(t) \neq 0$) the MBN travels at full speed and is away from $p(t)$ until $t \approx 4.6$. After this, the solution enters a singular arc[4](i.e. $\lambda(t) = 0$), and stays coincident with $p(t)$ until the end of the time horizon.

[4]In the plot, $\lambda(t)$ is not exactly equal to 0 during the singular arc. The reason for this is that

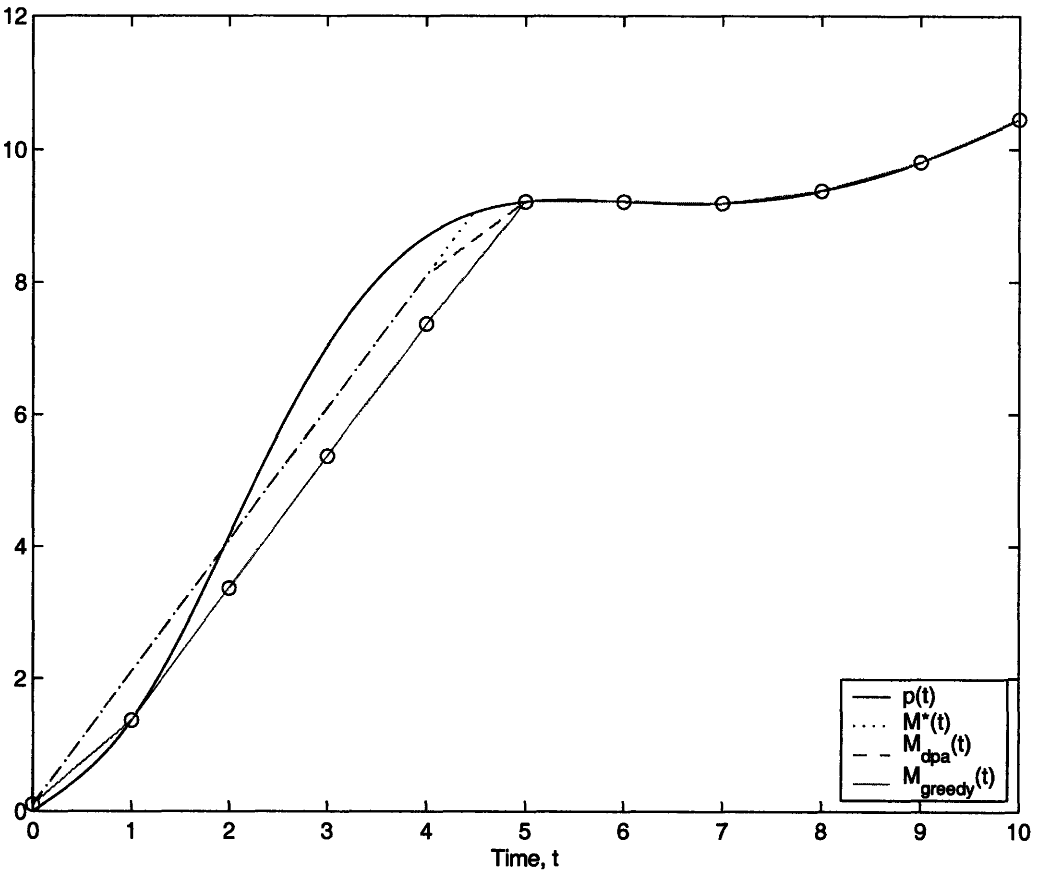

Figure 4-10: MPP-MFPA Single RN instance travelling on a line with trajectory $p(t) = t^4e^{-t}+t$. The MBN velocity is bounded by $V = 2$, and the CDMA throughput function is used with $\alpha = 2$, $b = 1$. Shown are a comparison of the Optimal Control Solution, DPA solution with grid spacing $\epsilon = 0.02$ and Greedy solution. Circles are used to depict the greedy algorithm's placements at each time step.

For the same 1-D RN position function, Fig. 4-10 shows a comparison of the optimal control solution with the solution achieved via the DPA and Greedy algorithms for the time horizon $t \in [0, 10]$. Both discrete algorithms use $\Delta t = 1$, and the DPA algorithm uses grid spacing $\epsilon = 0.02$. Quantitatively, the greedy algorithm obtains 92% of the MPP-MFPA objective achieved by the optimal control algorithm, and the the DPA algorithm 99%. Qualitatively, we see that the DPA solution exactly follows the optimal solution until time $t = 4$, at which point there is a slight deviation. In contrast, we see that the greedy lags the optimal solution during $1 \leq t \leq 5$ due to

a small positive numerical threshold of 0.08 was used in the Matlab simulation for the switch over from normal to singular control.

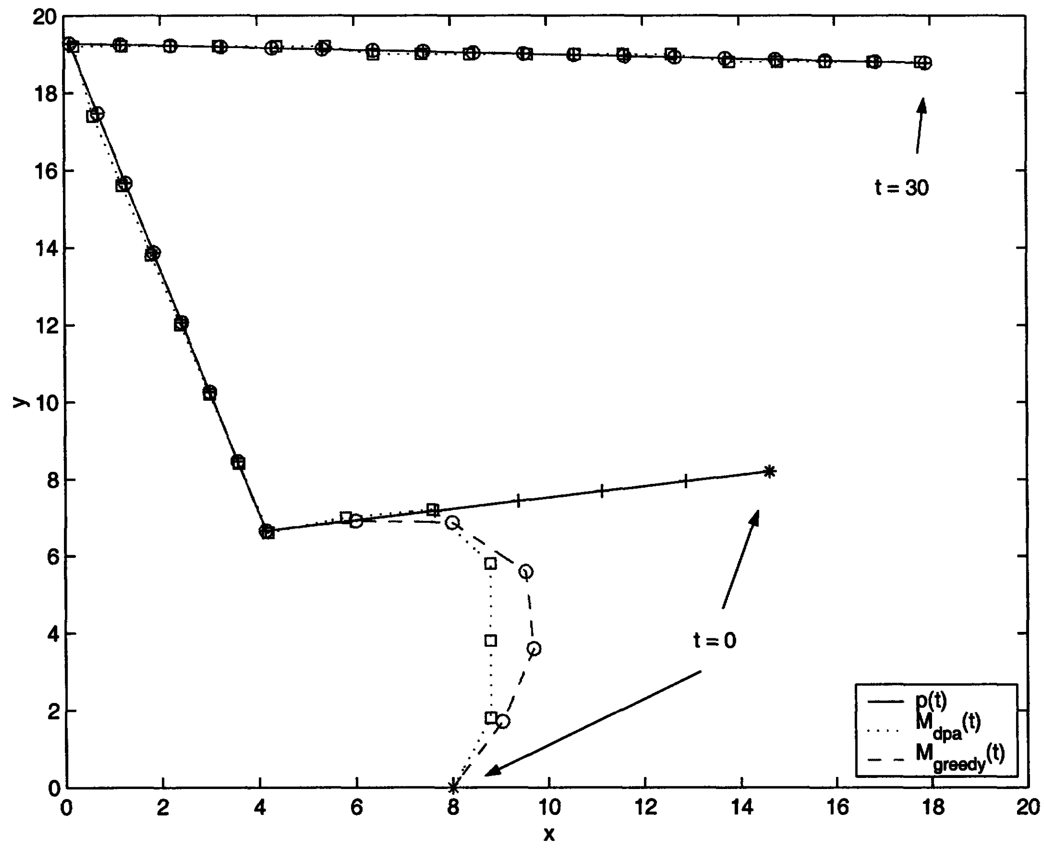

Figure 4-11: MPP-MFPA Single RN, 2-D example. The RN travels according to a random waypoint model. Both DPA and Greedy Approaches use $\Delta t = 1$. The plot show the MBNs spatial movement with respect to the RN, and '*' is used to depict the starting locations.

the choice of not travelling at full speed between during $0 \le t \le 1$.

Additional insight can be gained into comparing the greedy and DPA approach if we examine Figs. 4-11 and 4-12. Fig. 4-11 shows a single RN instance travelling in a 20×20 2-dimensional plane according to a Random Waypoint Model. In such a mobility model, RNs continually repeat the process of choosing a random location in the plane and travel there at a randomly chosen constant speed in the range $[V_{min}, V_{max}]$. We chose $V_{min} = 0.5, V_{max} = 2$, and assumed the MBN speed was bounded by $V = 2$. We consider a time horizon $t \in [0, 30]$ with $\Delta t = 1$ for both algorithms, and $\epsilon = 0.2$ for the DPA. Finally, we assume the MBN starts at $\boldsymbol{M}_0 = (8, 0)$, and denote starting points with a star. From the figure, we can see that early in the time horizon the greedy deviates from the DPA, but because the RN is not

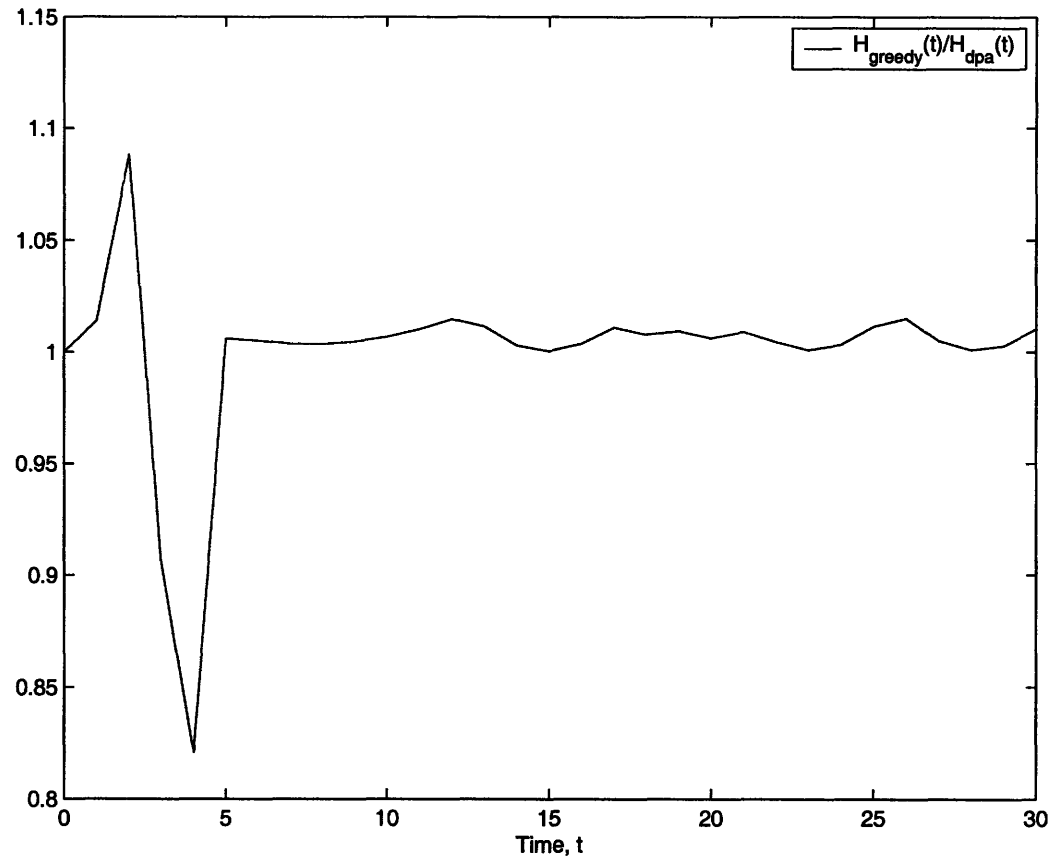

Figure 4-12: Evolution of the greedy to DPA performance ratio with respect to time. Plot corresponds to the 2D random waypoint example in Fig. 4-11.

moving faster than the MBN it is able to catch up by time step $t = 5$. As the performance ratio plot in figure 4-12 would indicate, it is up to here that the DPA algorithm seems to be performing better than the greedy algorithm. Specifically, the DPA is worse than the greedy during $0 \leq t \leq 2$, but better for $2 \leq t \leq 5$. For time steps $t \geq 5$ however, the greedy MBN is able to stay exactly on top of the RN, whereas the DPA MBN is restricted to travel between grid points. Again this is reflected in the instantaneous time performance ratio plot at the bottom since for time steps $t \geq 5$ the throughput achieved by the greedy algorithm is either as good as the DPA or slightly better. Indeed, in general it would seem that in situations when the MBN can travel at a speed as fast or faster than the RNs the greedy algorithm can perform quite well.

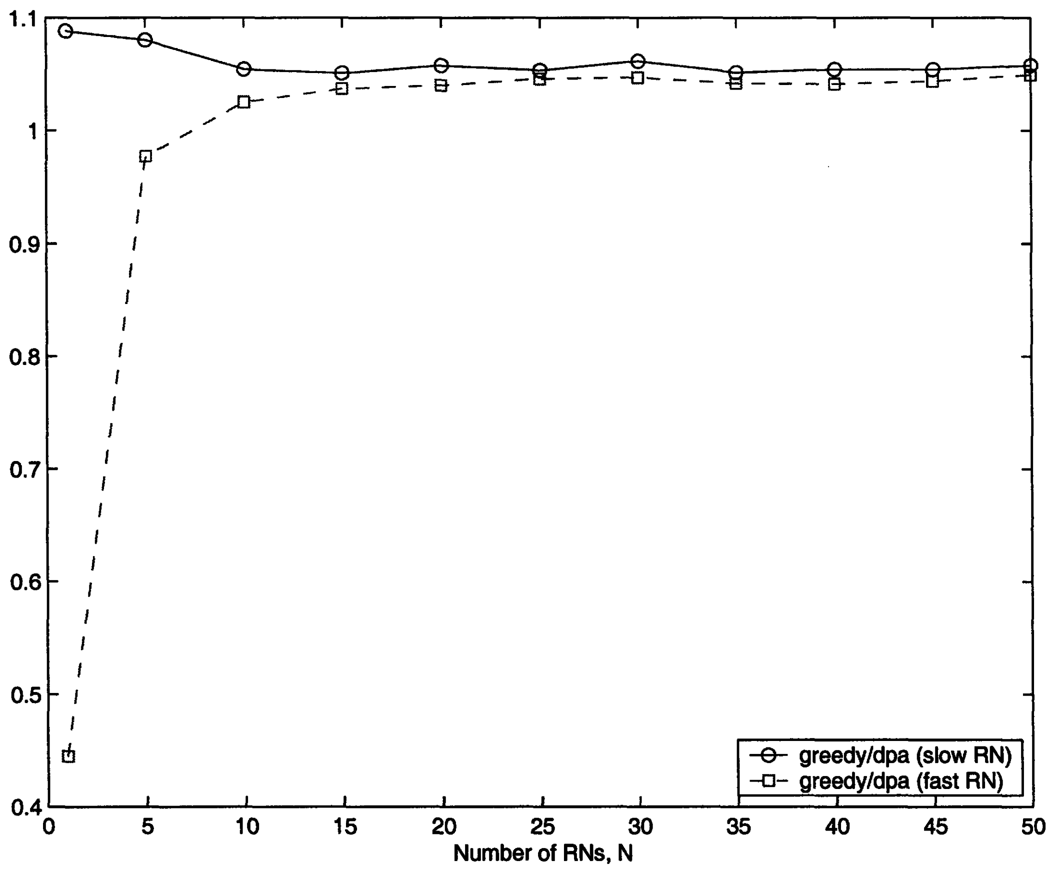

Figure 4-13: Constrained MBN speed MPP-MFPA average case plot for varying numbers of RNs, over a time horizon $t \in [0, 100]$ and $\Delta t = 1$ for both algorithms.

This observation is confirmed by the plot depicted in Fig. 4-13. It shows the average ratio (over 10 runs) between the MPP-MFPA objective achieved by the greedy to that achieved by the DPA algorithm for varying numbers of RNs over a time horizon $t \in [0, 100]$, with $\Delta t = 1$ and MBN speed bounded by $V = 2$. The mobility model used for the RNs is a random waypoint model over a 50×50 dimension plane, for a "fast RN" and "slow RN" scenario. For the "fast RN" scenario we assume $[V_{min}, V_{max}] = [5, 10]$, and we assume $[V_{min}, V_{max}] = [0.5, 2]$ for the "slow RN" scenario. As can be seen in the plot, for faster speeds and small numbers of RNs, the DPA algorithms outperforms the Greedy algorithm. However, once the number of RNs passes 10, the greedy performs as well or better than the DPA algorithm. This can likely be attributed to the fact that for larger number of RNs, the 1-center is more stable and slower moving, and thus the greedy can get to the exact location even

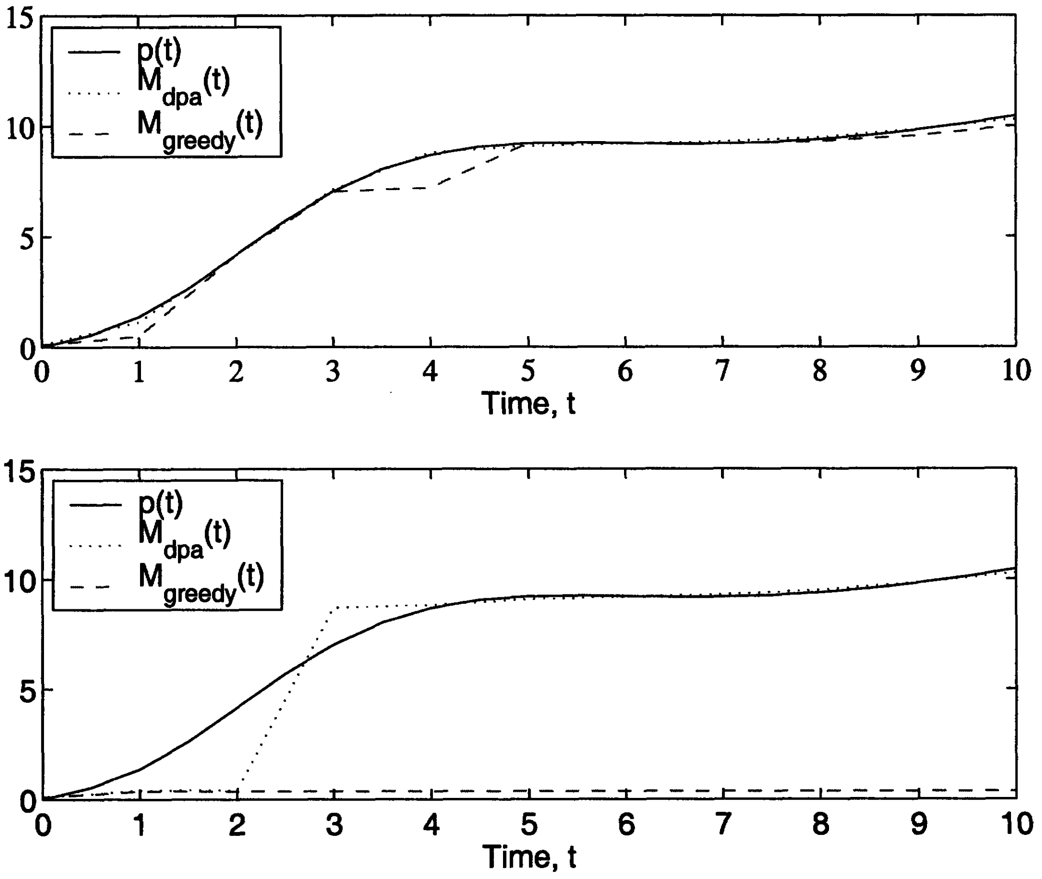

Figure 4-14: Relaxed MPP-MFPA Single RN instance travelling on a line with trajectory $p(t) = t^4e^{-t}+t$. Shown are a comparison of the DPA solution and the Relaxed Greedy Heuristic, over a time horizon $t \in [0, 10]$, and $\Delta t = 1$ for both algorithms. The relaxed objective function is $\frac{1}{d_{max}(t)^2+1} + \frac{c}{d[M(t-1),M(t)]^2+1}$ with the parameter c set as $c = 1$ in the top plot, and $c = 1.1$ in the bottom plot.

though the MBN is slower than individual RNs. In contrast, the DPA algorithm is limited to an approximate location (i.e. on a grid point). Indeed, for slower RNs the greedy outperforms the DPA algorithm for this reason.

Fig. 4-14 shows a single RN 1-D example plot comparing the discrete algorithms for the relaxed MPP-MFPA problem discussed in section 4.6. We consider the same RN position function as for Figs. 4-9 and 4-10 over the time horizon $t \in [0, 10]$. For the penalty function, we assume the form $G(x) = \frac{c}{x^\alpha+b}$ for $G()$, where $\alpha = 2$ and $b = 1$ are the same as for the CDMA throughput function $H()$. The motivation for $H()$ and $G()$ having similar forms is so that the tradeoffs can clearly be explored. We assume

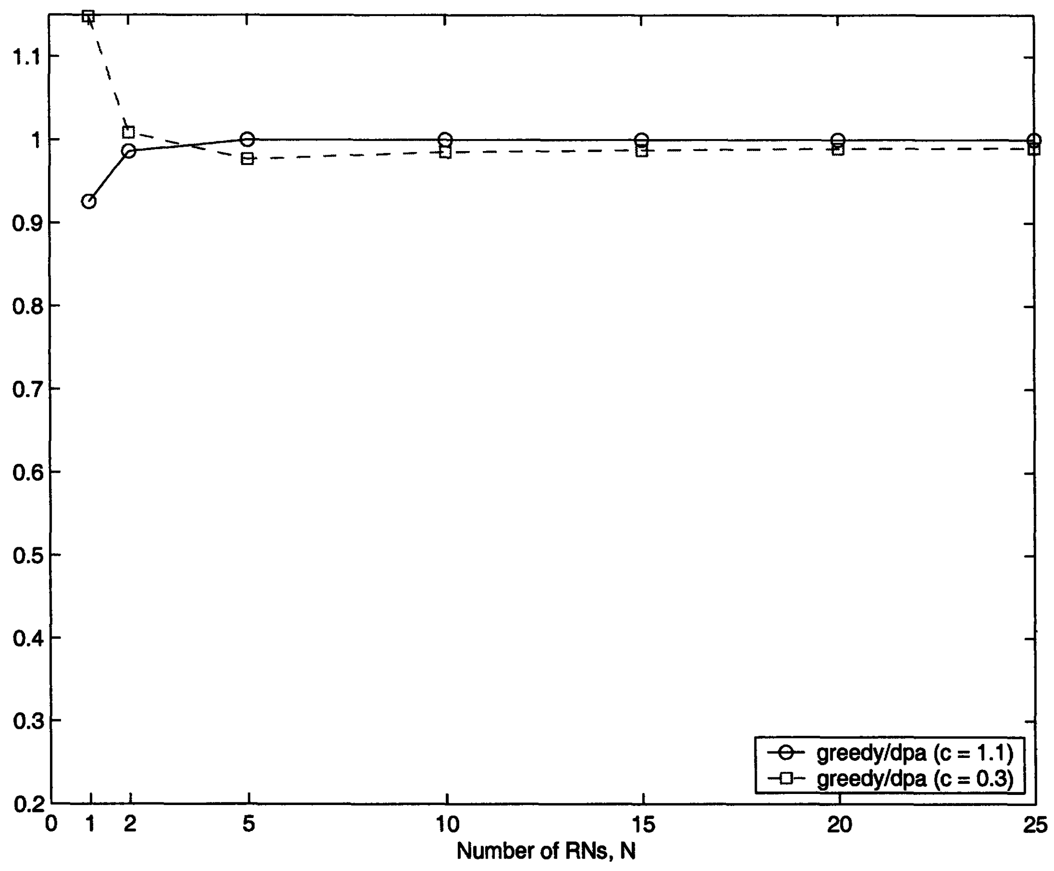

Figure 4-15: Relaxed MPP-MFPA average case plot for varying numbers of RNs, over a time horizon $t \in [0, 100]$ and $\Delta t = 1$ for both algorithms. The relaxed objective function is $\frac{1}{d_{max}(t)^2+1} + \frac{c}{d[M(t-1),M(t)]^2+1}$. The figure shows a plot for plots for $c = 0.3$ and $c = 1.1$.

$\Delta t = 1$ for both algorithms, and a grid spacing $\epsilon = 1$ for the DPA. From the plots we can see how sensitive the greedy algorithm is to the cost function. For example, in Fig. 4-14-a, for a value of $c = 1$, the relaxed greedy algorithm achieves 95% of the objective of the relaxed DPA algorithm. By contrast, in Fig. 4-14-b, when $c = 1.1$ the greedy algorithm achieves only 68% of the DPA's objective. The reason for this disparity can be seen by examining the decision of the greedy algorithm at time step $t = 1$. When $c = 1$, the tradeoff makes it worth it to move and thereby obtain a better value of $H()$. This decision propagates forward as the greedy algorithm continues to make correct decisions. However, when $c = 1.1$ the optimal 1-step tradeoff at time step $t = 1$, is to *not* move. This decision also propagates forward since for all time steps $t > 1$ the RN is even farther away and it is too late to move (i.e. the movement required is too large to justify the 1-step gain in $H()$ value).

Finally, Fig. 4-15 illustrates an average case plot for the relaxed MPP-MFPA for varying numbers of RNs. We assume the RNs move according to a random waypoint model with $[V_{min}, V_{max}] = [4, 8]$ over a 20×20 plane, over the time horizon $t \in [0, 100]$. We assume $\Delta t = 1$ for both algorithms and $\epsilon = 1$ for the DPA. We assume the same form for the penalty function $G()$ as for 4-14, with $c = 0.3$ and $c = 1.1$. As can be seen from the figure, for a single RNs the greedy algorithm performs better when $c = 0.3$ (i.e. more emphasis on optimizing $H()$), and the DPA performs better when $c = 1.1$ (i.e. more emphasis on simultaneously optimizing $H()$ and $G()$). However, it is interesting that as soon the number of RNs is increased above 1, the algorithms obtain more or less the same objective value for both values of c. The reason for this would seem that once $N > 1$ it is always likely that the farthest RN will be quite far away, and therefore given the form of the cost functions it almost never advantageous to move. Indeed, the solution of both algorithms involve the MBN being static for most of the time horizon.

4.10 Conclusion

Previous formulations of the Mobile Network construction problem have been based on placing and mobilizing the MBNs *reactively* based on the position of the RNs at the "current" time. Yet, in many practical scenarios entire RN trajectories are known a-priori, and if this is the case, then placing the MBNs by solving a placement problem independently at each time step is in general suboptimal.

To address this, in this chapter we considered the path planning of a single MBN with the goal of maximizing the time-average system throughput. We have assumed that the velocity of the MBN factors into the performance objective as either a constraint or a penalty. We first considered a discrete approach, for which our main result is a dynamic programming based approximation algorithm for the path planning problem. We provided worst case analysis of the performance of the algorithm. Additionally, we develop an optimal algorithm for the 1-step velocity constrained path planning problem. Using this as a sub-routine, we develop a greedy heuristic algorithm for the overall path planning problem. We also considered the path-planning problem from a continuous perspective. We formulated the problem as an optimal control problem, and developed interesting insights into the structure of the optimal solution. Finally, we discussed extensions of the base discrete and continuous formulations and compared the various approaches via simulation.

Future work includes extending the single MBN formulation to multiple MBNs and incorporating the assignment subproblem from chapter 3. General solutions for the continuous formulations would also be highly desirable.

Chapter 5

Conclusions

The architecture of a hierarchical Mobile Backbone Network has been presented recently. Such an architecture can significantly improve the performance, lifetime, and reliability of MANETs and Wireless Sensor Networks. In this Thesis, we have considered the problem of how to (i) Construct such Mobile Backbone Networks so as to optimize a network performance metric, and (ii) Maintain such networks under node mobility.

In the first part of this thesis, we have concentrated on placing and mobilizing backbone nodes, dedicated to maintaining connectivity of the regular nodes. We have formulated the Mobile Backbone Nodes placement problem as the Connected Disk Cover (CDC) problem and shown that it can be decomposed into two subproblems. We have proposed a number of *distributed* algorithms for the first subproblem (Geometric Disk Cover), bounded their worst and average case performance, and studied their performance under mobility via simulation. A new approach for the solution of the second subproblem (STP-MSP) and of the joint problem (CDC) has also been proposed. We have demonstrated via simulation that when it is used to solve the CDC problem in a centralized manner, the number of the required MBNs is significantly reduced.

In the second part of this thesis, we have focused on the problem of how to jointly place the Mobile Backbone Nodes (MBNs), and assign every Regular Node to exactly one MBN. In particular, we considered this problem under the assumption that the number of available MBNs is fixed a-priori, and formulated two problems under a general communications model. The first is the Maximum Fair Placement and Assignment (MFPA) problem in which the objective is to maximize the throughput of the minimum throughput RN. The second is the Maximum Throughput Placement and Assignment (MTPA) problem, in which the objective is to maximize the aggregate throughput of the RNs. Our main result is a novel optimal polynomial time algorithm for the MFPA problem for fixed K. We have also provided an optimal solution for a restricted version of the MTPA problem for $K \leq 2$. We have developed two heuristic algorithms for both problems, including an approximation algorithm with bounded worst case performance loss. Finally, we have presented simulation results to evaluate the performance of the various algorithms developed in the paper.

In the third part of the thesis, we have considered the problem of placing the Mobile Backbone Nodes over a finite time horizon. We assume complete a-priori knowledge of each of the RNs' trajectories over a finite time interval, and formulated the problem of determining the optimal MBN *path* over that time interval. We considered an objective function that maximizes the time-average system throughput, subject to a constraint or penalty on the MBN velocity. For a discrete formulation of the problem our result is a dynamic programming based approximation algorithm, for which we have provided worst case performance analysis. Additionally, we have developed an optimal algorithm for the 1-step velocity constrained path planning problem. Using this as a sub-routine, we have develop a greedy heuristic algorithm for the overall path planning problem. We also formulated the path planning problem from a continuous perspective as an optimal control problem. We developed several interesting insights into the structure of the optimal solution. Finally, we have discussed extensions of the base discrete and continuous formulations and compared the various developed approaches via simulation.

Overall, our goal in this thesis has been to study and provide a basic theoretical framework for the fundamental problems associated with the Mobile Backbone Architecture for Wireless Networks. To this end, we have formulated several interesting problems, proposed various combinatorial algorithms and have provided both theoretical as well as simulation-based performance analysis. Several open problems have emerged from this work. Three such problems are (i) Developing distributed algorithms for the CDC problem capable of dealing with the mobility of cover MBNs, (ii) Developing more efficient, distributed and mobility-handling algorithms for the MFPA and MTPA problems and (iii) Extending the path planning framework to multiple MBNs and in doing so incorporating the RN assignment subproblem.

Appendix A

Placing the Cover MBNs - Extensions to Non-Strip Based Algorithms

A.1 Introduction

Recall the architecture of a Mobile Backbone Network, shown back in Fig. 1-1 in chapter 1. The set of MBNs has to be placed and *mobilized* such that (i) every RN can directly communicate with at least one MBN, and (ii) the network formed by the MBNs is connected. We assume a *disk* connectivity model, whereby two nodes can communicate if they are within a certain range. We also assume that the communication range of the MBNs is significantly larger than the communication range of the RNs. The problem of placing the minimum number of MBNs was termed in chapter 2 as the Connected Disk Cover (CDC) problem. A similar problem has been recently also formulated in [87].

The algorithms in chapter 2 focus on *controlling the mobility* of the MBNs in order to provide a backbone for reliable communication. These algorithms are based on the fact that the CDC problem can be decomposed into the Geometric Disk Cover (GDC) problem and the and the Steiner Tree Problem with Minimum Number of

Steiner Points (STP-MSP). It was shown in chapter 2 that if the GDC and STP-MSP subproblems are solved separately by γ and δ-approximation algorithms, the approximation ratio of the joint solution is $\gamma+\delta$.

Motivated by this decomposition result, in this chapter, we focus on the GDC subproblem. This problem can be stated as: given a set of points in the plane, place the minimum number of disks such that all points are covered. Due to the our focus on decentralized operation in a mobile environment, we aim to develop *distributed* algorithms that maintain a disk cover under *mobility*. It follows from the decomposition result that any improvement in the approximation ratio of the GDC problem (γ) immediately improves the approximation ratio of the overall CDC solution. Hence, the developed algorithms are an important building block for any decomposition-based Mobile Backbone Network algorithm.

The Mobile GDC problem also stands alone as an important problem and has several applications in MANETs [34],[54], and in WSNs. For example, a possible application is in the area of point coverage in sensor networks (e.g. [65]), where sensors have to track or follow a set of moving targets. Hershberger [47] points out applications in databases, where clustering can support queries regarding time-varying data. Finally, in the context of Mobile Backbone Networks, assuming that MBNs can communicate with each other over long distances ensures that the MBNs' network is always connected and reduces the CDC problem to GDC problem.

The *static* GDC problem has been extensively studied in the past. Hochbaum and Maass [52] provided a Polynomial Time Approximation Scheme (PTAS) for the problem. However, their algorithm is impractical for our purposes, since it is centralized and has a high running time for reasonable approximation ratios. Several other *centralized* algorithms have been proposed. For example, Gonzalez [38] presented an algorithm based on dividing the plane into strips, whose approximation ratio has been recently shown to be 6 [83]. Franceschetti et al. [32] developed an algorithm that places disks only on vertices of a mesh. A table comparing the various centralized GDC algorithms can be found in [32].

As mentioned above, the properties of wireless networks call for *distributed* disk cover algorithms that deal with RNs *mobility*. However, only a few recent works have focused on algorithms that maintain coverage under mobility (i.e. solve the Mobile GDC problem) and even fewer proposed distributed algorithms. We note that *clustering* given nodes to form a hierarchical architecture has been extensively studied in the context of wireless networks (e.g. [9],[11],[37]). However, the idea of deliberately controlling the motion of *specific* nodes in order to maintain some desirable network property has been introduced only recently (e.g. [58],[83]).

In the specific context of the Mobile GDC problem, [54] present a 7-approximation distributed algorithm. Hershberger [47] presents a *centralized* 9-approximation algorithm for a slightly different problem: the mobile geometric *square* cover problem. Gao et al. [34] study a closely related problem in which the centers have to be selected from the set of points (i.e. RNs). Finally, in chapter 2 we presented a number of distributed approximation algorithms for the Mobile GDC problem.

Similarly to the formulation in chapter 2, we assume that all nodes can detect their position via GPS or a localization mechanism. This assumption allows to take advantage of location information in designing distributed algorithms. However, the algorithms in chapter 2 solve the Mobile GDC problem by dividing the plane into strips, solving the GDC problem locally within strips, and finally combining these solutions to form an overall solution. One of the advantages of this type of a *strip-based* algorithm is that the optimization is easier within a narrow strip, as opposed the whole plane. Another advantage is that the computation is localized to within strips, yielding a sort of spatial decentralization of both computation and communication.

However, a drawback of this approach is the fact that cross-strip optimization cannot be exploited. A typical example of the resulting inefficiency is depicted in Fig. A-1, in which a strip-based algorithm uses two MBNs to cover two RNs that could obviously be covered by a single MBN. In this chapter we present and analyze a number of new *planar-based* distributed algorithms that do not use strips. Yet, we show they are still able to distributedly solve the Mobile GDC problem while providing good performance guarantees.

We start by presenting a novel family of algorithms that periodically merge neigh-

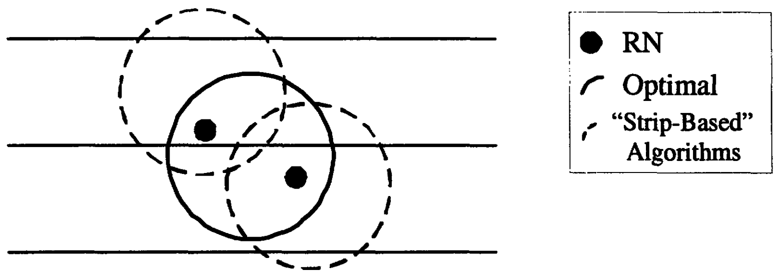

Figure A-1: Example of basic inefficiency of strip-based algorithms.

boring MBNs (if possible) and spatially separate groups of neighboring MBNs (if required). Analyzing the worst case performance of these algorithms requires developing a novel graph-based technique. We use this techniques to obtain the approximation ratios of the algorithms. We later show via simulation that on average the algorithms perform better than the strip-based algorithms.

We then present a very simple 5-approximation algorithm that is based on an *overlooked* observation regarding the relation between the GDC problem and the maximal independent set problem. We show that placing the MBNs (i.e. the disk centers) on top of some of the RNs (points) yields a restricted GDC problem, which is equivalent to a minimum dominating set problem in a unit disk graph. We show that we can find an approximate solution to the unrestricted problem by finding a maximal independent set in the unit disk graph. This simple observation is important, since it immediately provides a 5-approximation distributed algorithm for the static and mobile GDC problems, whereas in the past much effort has been dedicated to developing centralized algorithms with higher complexities and approximation ratios (see the table in [32]).

Then, we evaluate the performance of the algorithms via simulation. We start by studying the performance under mobility and by comparing the performance of the planar algorithms, presented in this chapter, to a number of previously presented Mobile GDC algorithms. Then we compare average case and simulation results of the different algorithms.

To summarize, our main contribution is the development and analysis of distributed algorithms for the Geometric Disk Cover problem in a mobile environment.

These algorithms may operate on a stand-alone basis or provide an important building block for the Mobile Backbone Network algorithms.

This chapter is organized as follows. In Section A.2 we formulate the problem. The new distributed planar algorithms are presented and analyzed in Sections A.3 and A.4. In Section A.5 we evaluate and the performance of the algorithms via simulation. We summarize the results and discuss future research directions in Section A.6.

A.2 Problem Formulation

We consider a set of *Regular Nodes* (RNs) distributed in the plane and assume that a set of *Mobile Backbone Nodes* (MBNs) has to be deployed to cover them. We denote by N the collection of *Regular Nodes* $\{1, 2, \ldots, n\}$ and by $M = \{d_1, d_2, \ldots, d_m\}$ the collection of MBNs. The locations of the RNs are denoted by the $x - y$ tuples (i_x, i_y) $\forall i$ and d_{ij} denotes the distance between nodes i and j.

We assume that the RNs and MBNs have both a communication channel (e.g. for data) and a low-rate control channel. For the communication channel, we assume a disk connectivity model. Namely, an RN i can communicate bi-directionally with another node j (e.g. an MBN) if the distance between i and j, $d_{ij} \leq r$. We denote by $D = 2r$ the diameter of the disk covered by an MBN communicating with RNs. For the control channel, we assume that both RNs and MBNs can communicate over a much longer range than their respective data channels. Since given a fixed transmission power, the communication range is inversely related to data rate, this is a valid assumption.

For this work, we assume that the number of available MBNs is not bounded (e.g. if necessary, additional MBNs can be dispatched). Yet in our analysis, we will try to minimize the number of MBNs that are actually deployed. We formulate the Geometric Disk Cover (GDC) problem [52], as follows:

Problem GDC: Given a set of RNs (N) distributed in the plane, place the smallest set of MBNs (M) such that for every RN $i \in N$, there exists at least one MBN $j \in M$ such that $d_{ij} \leq r$.

The Mobile GDC problem is implicit in the above formulation, as the goal is to maintain a valid GDC under RN mobility. We assume there exists some sort of MBN routing algorithm, which routes specific MBNs to their new locations. The actual development of such an algorithm is beyond the scope of this chapter.

Before proceeding, we introduce additional notation required for the presentation and analysis of the algorithms. Note that in the formulation of the Mobile GDC problem it is required that every RN is connected to at least one MBN. We assume that even if an RN can connect to multiple MBNs, it is actually assigned to exactly one MBN. Thus, we denote by P_{d_i} the set of RNs connected to MBN d_i. We denote by d_i^L, d_i^R, d_i^B and d_i^T the leftmost, rightmost, bottommost, and topmost RNs connected to MBN d_i. Their (x, y) co-ordinates are denoted with $x-y$ subscripts, e.g. $(d_i^L)_x, (d_i^L)_y$.

A.3 Planar Merge-And-Separate Algorithms

In this section we present and analyze a family of distributed algorithms for the Mobile GDC problem. We refer to these algorithms as the Planar Merge-And-Separate (PMAS) algorithms. These algorithms build upon the ideas presented in the development of the in-strip Merge-And-Separate (MAS) algorithm in chapter 2 section 2.4.3. However, as mentioned in Section A.1, the PMAS algorithms are planar-based as opposed to the strip-based algorithms of chapter 2. The advantage of this approach is that it avoids inherent inefficiencies resulting from dividing the plane into strips and takes advantage of possible cross-strip optimizations.

A.3.1 Distributed Algorithms

Our presentation is in the form of a generic algorithm, with three versions[1]: (i) Square-Cover with Rectangular Separation (SC) (ii) Disk-Cover with Rectangular Separation (DCR), and (iii) Disk-Cover with Circular Separation (DCC). The two disk-cover versions, i.e. DCR-PMAS and DCC-PMAS, constitute *distributed* algorithms for the

[1]In the description of the algorithm, it should be clear which procedure applies to which algorithm version.

Mobile GDC problem. The Square Cover Planar MAS (SC-PMAS) is a distributed algorithm that places the minimum number of $D \times D$ *squares* to cover the RNs. Note that the SC-PMAS algorithm is *not* applicable to the Mobile GDC problem. It is presented here solely to serve as a simple demonstration of the analysis technique that is developed for analyzing the DCR-PMAS and DCC-PMAS algorithms.

Algorithm 1/2/3 SC-PMAS, DCR-PMAS, DCC-PMAS algorithms (at MBN d_i, RN q)

```
Disconnection Rule (at RN $q$)
 1: if $q$ uncovered then
 2:     place MBN $d_i$, set $P_{d_i} \leftarrow q$
Merge Rule (at MBN $d_i$)
 3: call Chk-Sqr-Merge($d_i$), or Chk-Dsk-Merge($d_i$)
Separate Rule (at MBN $d_i$)
 4: call SC-Separate(), DCR-Separate(), or DCC-Separate()
Procedure Chk-Sqr-Merge($d_i$)
 5: for all MBNs $d_j$ within $3\sqrt{2}D$ of $d_i$ do
 6:     if $P_{d_i} \cup P_{d_j}$ coverable by a single $D \times D$ square then
 7:         merge $d_i$ and $d_j$
Procedure Chk-Dsk-Merge($d_i$)
 8: for all MBNs $d_j$ within $2D$ of $d_i$ do
 9:     if $P_{d_i} \cup P_{d_j}$ coverable by a single disk then
10:         merge $d_i$ and $d_j$
Procedure SC-Separate() (see Fig. A-2(a))
11: if $\exists$ 9 MBNs $a_1, \dots, a_9$ (including $d_i$) such that all RNs $q \in \cup_{j=1}^{9} P_{a_j}$ lie within a
    $3D \times 3D$ area then
12:     separate and reorganize $a_1, \dots, a_9$
Procedure DCR-Separate() (see Fig. A-2(b))
13: if $\exists$ 17 MBNs $a_1, \dots, a_{17}$ such that all RNs $q \in \cup_{j=1}^{17} P_{a_j}$ lie within a $3D \times 3D$
    area then
14:     separate and reorganize $a_1, \dots, a_{17}$
Procedure DCC-Separate() (see Fig. A-2(c))
15: if $\exists$ 14 MBNs $a_1, \dots, a_{14}$ such that all RNs $q \in \cup_{j=1}^{14} P_{a_j}$ lie in a circular area of
    diameter $3D$ then
16:     separate and reorganize $a_1, \dots, a_{14}$
```

The generic PMAS algorithm is simple, and the basic idea is that we periodically enforce a *merge rule* and a *separate rule* at each MBN d_i. Additionally, a *disconnection rule* is enforced at each RN q. Namely, if at any time q is not covered by any MBN, assign a new MBN to cover q.

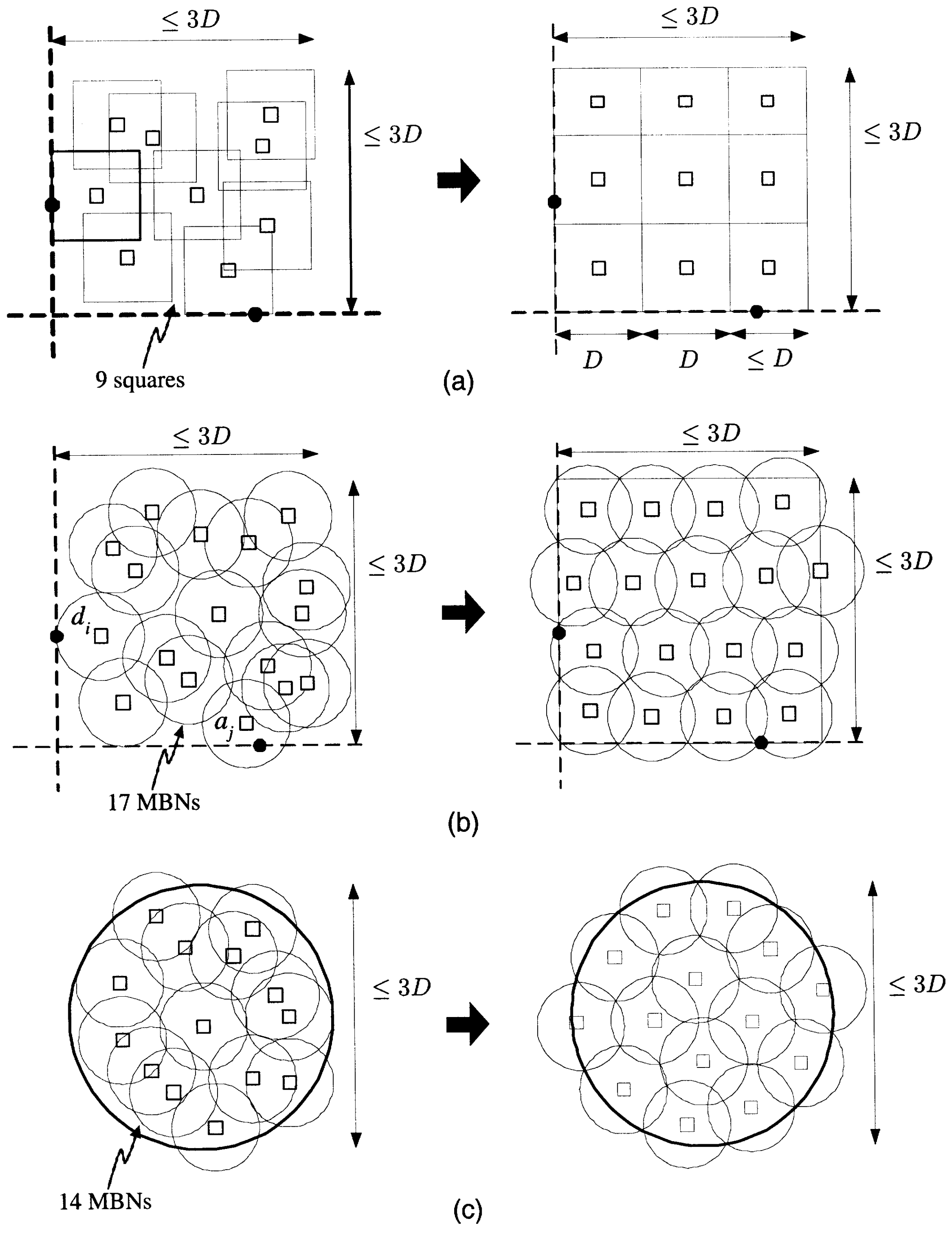

Figure A-2: Planar MAS separation rules: (a) SC-PMAS, (b) DCR-PMAS, and (c) DCC-PMAS.

Initially, we assume that there is an MBN covering each individual RN (i.e. as per the disconnection rule). The *merge rule* states that if there exists another MBN d_j that can be merged with d_i (i.e. $P_{d_i} \cup P_{d_j}$ coverable by a single MBN), then *merge* d_i and d_j. The *separate rule* states that if the point-sets of too many mutually non-mergeable MBNs simultaneously converge on a sufficiently small area, then these MBNs should be *separated* (i.e. the MBNs relocated and their point-sets reassigned), as illustrated in Fig. A-2. The reasoning behind the choice of the numbers defining *too many* and *sufficiently small* area (e.g. 17 and a $3D \times 3D$ square for DCR-PMAS) will become clear in the next section, when we bound the worst case performance of the algorithms.

For correctness of the algorithm, we assume that both the merge and separate operations can be executed atomically (i.e. without any interrupting operation). We also use the convention that an MBN can be placed arbitrarily within its coverage disk, as long as it is within distance r from all the RNs it is covering. For square MBNs (i.e. for the SC-PMAS), we assume simply that the MBNs are placed somewhere in the $D \times D$ coverage square. Finally, we assume that if at any time an MBN does not cover any RNs (e.g. after a separation operation), it is released.

Note that in the description of the PMAS algorithm, the separate rules are described in general terms, as opposed to an explicit implementation. The reason for this is that there are several possible ways to implement the algorithm, and our goal is to convey the general idea. An example of a distributed implementation of the DCR-PMAS separation rule at MBN d_i could be as follows. MBN d_i starts by detecting all the MBNs (including itself) d_j within distance $4\sqrt{2}D$, and for which d_j^L, d_j^R, d_j^B and d_j^T all lie within an $x-y$ range of $\left[(d_i^L)_x, (d_i^L)_x + 3D\right]$, $\left[(d_i^T)_y + 3D, (d_i^B)_y - 3D)\right]$. Next, these detected MBNs are sorted by ascending bottommost point y-coordinate, yielding a sorted list, denoted by $\{a_1, a_2, \ldots, a_Q\}$. Now, d_i can sequentially check whether $(a_{j+8}^T)_y - (a_j^B)_y \leq 3D$. If this condition holds, then it can conclude that all of the RNs covered by these 9 disks $a_j, \ldots, a_{j+8}$ lie in a $3D \times 3D$ area. At this point, a separate operation can be initiated by sending messages to the appropriate MBNs to move to their new coordinates, and reassign RNs as illustrated in Fig. A-2-b. Note that the reassignment of RNs would require additional messages in order to inform each RN of its new covering MBN. The points of reference for the separation are $(d_i^L)_x$ and $(a_j^B)_y$, which are shown in the figure. In particular, the left-bottommost corner of the $3D \times 3D$ area in Fig. A-2-b would be $\left[(d_i^L)_x, (a_j^B)_y\right]$.

A.3.2 Worst Case Performance

We now analyze the worst case performance of the PMAS algorithms. The induction-based methodology used in the analysis of the strip-based algorithms of chapter 2 cannot be extended to 2-dimensions, since there is no left-to-right directionality that can be exploited. Thus, we develop a novel graph-based analysis technique, which we demonstrate by first analyzing the Square Cover version of the PMAS algorithm (SC-PMAS). We then show how this can be straightforwardly applied to the Disk-Cover versions of the PMAS algorithm.

We use $OPT = \{d_1, d_2, \ldots, d_{|OPT|}\}$ to denote an optimal solution and $ALGO = \{a_1, a_2, \ldots, a_{|ALGO|}\}$ for an SC-PMAS solution. Let P_{d_i} and P_{a_i} represent the sets of RNs covered by the OPT square d_i and the ALGO square a_i, respectively. We define the notion of a_i *touches* d_i (or vice versa) as if and only if there exists at least one RN q, such that $q \in P_{a_i}$ *and* $q \in P_{d_i}$. Finally, define the notion of the PMAS algorithm being in *steady state* if there are no merge or separate actions currently pending.

Lemma A.3.1. *In steady state, no more than 8 SC-PMAS ALGO squares can touch a single OPT square d_i.*

Proof. Suppose 9 ALGO squares each covered at least one point from P_{d_i}. However, if this was the case then all of the points covered by these 9 squares must lie in a $3D \times 3D$ area, and would have been reorganized as per the separation rule illustrated in Fig. A-2(a). Once reorganized, an OPT square can clearly touch at most 4 ALGO squares, which is a contradiction. □

Lemma A.3.2. *In steady state, at most one SC-PMAS ALGO square a_i can exclusively touch a single OPT square d_j (i.e. $P_{a_i} \subseteq P_{d_j}$).*

Proof. Suppose there existed 2 ALGO squares a_1, a_2 that exclusively touched a single OPT square d_j (i.e. $P_{a_1} \cup P_{a_2} \subseteq P_{d_j}$). However, by definition this means that the set of RNs covered jointly by a_1 and a_2 could be covered by a single square. It follows that in steady state a_1 and a_2 would have been merged as per the merge rule, which is a contradiction. □

We are now ready to prove the performance guarantee of the SC-PMAS algorithm.

Theorem A.3.1. *In steady state, the SC-PMAS algorithm is a 4.5-approximation algorithm.*

Proof. We construct an undirected graph $G = (V, E)$ as follows. Define a vertex $v \in V$ for each of the *OPT* squares. For each *ALGO* square a_i, we associate *exactly one edge* according to two cases: (i) if a_i only touches a single *OPT* square d_j, define a self-loop edge (d_j, d_j), and (ii) if a_i touches multiple *OPT* squares $d_p, d_q, \ldots$, then pick two of these OPT squares (arbitrarily) and define an edge between them (e.g. (d_p, d_q)). Note that there could be both self-loops and parallel edges in the resultant graph. An example of the graph transformation is depicted in Fig. A-3.

Finally, since we have associated exactly one *ALGO* square with one edge, we have that $|V| = |OPT|$ and $|E| = |ALGO|$. Using the standard formula for counting the number edges in an undirected graph with self-loops we have that by lemmas A.3.1 and A.3.2,

$$\begin{aligned} |E| &= \sum_{v \in V} \left(\frac{d(v) - s(v)}{2} + s(v) \right) \\ &\leq \sum_{v \in V} \left(\frac{7}{2} + 1 \right) = \frac{9}{2} |V|, \end{aligned}$$

where $d(v)$ represents the degree of node v, and $s(v)$ the number of self-loop edges at v. □

At this point, the reasoning behind the exact numbers defining the PMAS separation area (denoted A), and the number of MBNs that must converge on A before

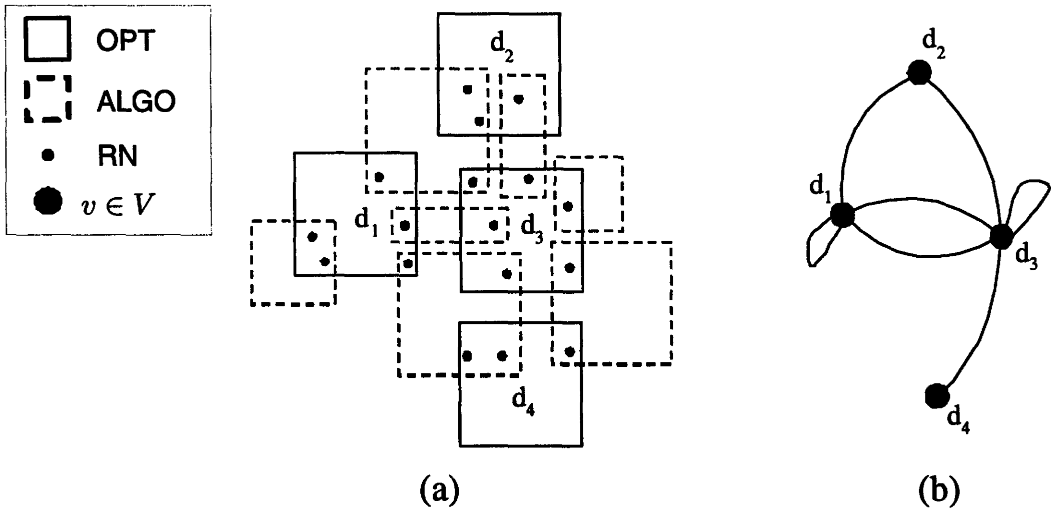

Figure A-3: Demonstration of a graph transformation: (a) original network and square cover, and (b) transformed graph.

separation (e.g. 9 and $3D \times 3D$ square for the SC-PMAS), can be more clearly understood. In turn with Lemma A.3.1, A is defined to be a minimal area satisfying the following: Consider some optimal square (disk) d. For any algorithm square a to touch d, it must only cover RNs which lie in A. Furthermore, a valid separation and reorganization can only be ensured if the squares involved can compactly cover the separation area, so as to ensure all RNs within A are still covered after the separation. Therefore, the number of separated PMAS MBNs (e.g. 9, 17 and 14 respectively) represent the minimum number of MBNs required to compactly cover their respective separation areas.

We are now ready to analyze the disk cover versions, starting with the DCR-PMAS. To do so, we can use the exact same analysis as for the square cover version. To start, we restate lemmas A.3.1 and A.3.2 (whose proofs are identical, except reapplied to disks) in the context of disks, followed by the approximation ratio theorem.

Lemma A.3.3. *In steady state, no more than 16 DCR-PMAS ALGO disks can touch a single OPT disk d_i.*

Lemma A.3.4. *In steady state, at most one DCR-PMAS ALGO disk a_i can exclusively touch a single OPT disk d_j (i.e. $P_{a_i} \subseteq P_{d_j}$).*

Theorem A.3.2. *In steady state, the DCR-PMAS algorithm is a 8.5-approximation algorithm.*

Proof. Using the same definitions and graph transformation as from the proof of Lemma A.3.1, we have that, $|E| \leq \sum_{v \in V}(15/2 + 1) = 8.5|V|$. □

For the DCC-PMAS algorithm, the proof is identical and thus we simply state the result.

Theorem A.3.3. *In steady state, the DCC-PMAS algorithm is a 7-approximation algorithm.*

A.3.3 Complexity

When discussing the complexity of the distributed algorithms presented in this chapter, we will use two standard measures, both with respect to the complexity expended in reaction to a single RN movement. The first is the *time complexity*, which we define as the number of communication rounds and the second is the *local computation complexity* at each MBN, which for a viable algorithm should be negligible compared to a communication round length.

The local computation complexity of the DCR-PMAS algorithm is a periodic $O(C(n))$ to evaluate the merge rule, where $C(n)$ is the running time of the decision 1-center subroutine used. Various efficient algorithms exist that solve the decision 1-center problem, an example being an $O(n \log n)$ algorithm in [51]. The separate rule can be evaluated in $O(1)$, since a packing argument can be used to show that at most 48 MBNs (i.e. a constant number) need be detected by an MBN d_i before there *must* exist 17 MBNs whose points all lie within a $3D \times 3D$ area. Since all point transfers are local (i.e. only take place between adjacent MBNs), the time complexity (number of rounds) is $O(1)$. Hence, this algorithm is implementable in realistic scenarios.

While the merge rule of the DCC-PMAS algorithm also entails a local, periodic $O(C(n))$ computation, implementing the separation rule is much more complex. An example implementation could be examining all circumcircles defined by pairs and

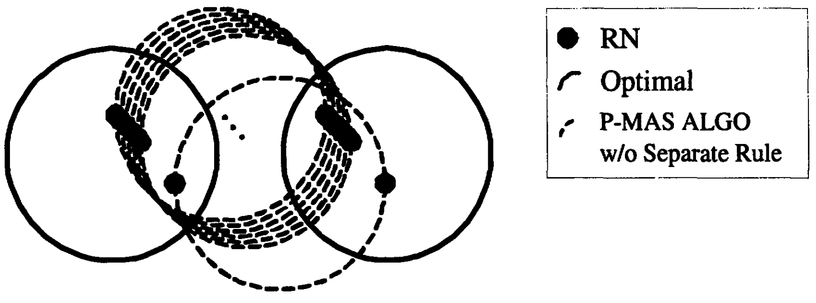

Figure A-4: A pathological example of arbitrarily bad performance of a PMAS algorithm without the separate rule.

triplets of RNs whose ensuing radii are at most $3D/2$, and testing whether the point-sets of 14 MBNs lie within. Note however, that this entails a *centralized* $O(n^3C(n))$ computation (e.g. by collecting all RN location information at some MBN), which is much too high to implement frequently.

Fortunately, an important note regarding the PMAS algorithms is that the merge rule is far more important than the separate rule. It turns out the merge rule is the one that ensures good average performance, whereas the separate rule protects against the rare, pathological yet theoretically possible cases of extreme inefficiency. An example of such a pathological situation is shown in Fig. A-4, in which an arbitrarily large number of mutually non-mergeable MBNs cover points coverable by 2 optimal MBNs. However, such situation would almost never occur in any practical scenario and thus the separate rule need only be implemented very rarely, perhaps making the DCC-PMAS also a viably implementable algorithm in certain scenarios.

A.4 Cluster Cover Algorithm

In this section we present the Cluster Cover (CC) algorithm which, like the PMAS algorithms, distributedly solves the Mobile GDC problem without the use of strips. The advantage of the CC algorithm over the PMAS algorithms is that it is simpler to implement, and has a lower computational complexity. Furthermore, we show that the approximation ratio of the CC algorithm is lower than that of the PMAS algorithms. Yet, as will be shown via simulation, on average the PMAS algorithms perform significantly better than the CC algorithm.

Before describing the algorithm we present the following definitions. Given an undirected graph $G(V, E)$, a *dominating set* is as a subset $Q \subseteq V$ such that $\forall i \in V$, either $i \in Q$ or $\exists(i, j) \in E$ for some $j \in Q$. An *independent set* is defined as a subset $Q \subseteq V$ such that $\forall i, j \in Q$, $\nexists(i, j) \in E$. Finally, given N points (RNs) distributed in the plane, a *unit disk graph* $G = (V, E)$ is defined such that $V = N$ and $(i, j) \in E \Leftrightarrow d_{ij} \leq r$.

The CC algorithm is based on an *overlooked* observation regarding the relation between the GDC problem and the Maximal Independent Set (MIS) problem. Before describing this relation, we note that restricting the locations of the MBNs (i.e. the disk centers) to the locations of the RNs (points) yields a restricted version of the GDC problem. This restricted GDC problem is equivalent to a *Minimum Dominating Set* (MDS) problem in a unit disk graph. Hence, $|GDC_{OPT}| \leq |MDS_{OPT}|$, where $|GDC_{OPT}|$ and $|MDS_{OPT}|$ are the cardinalities of the optimal solutions to the unrestricted GDC problem and to the MDS problem in a unit disk graph.

An MIS is by definition a dominating set. Therefore, finding an MIS provides an approximate solution to the MDS problem. An MIS can be found in linear time by a simple centralized algorithm that adds nodes to the set and then deletes their neighbors from the graph. It was shown in [69, Theorem 4.8] that in unit disk graphs the cardinality of an MIS is at most 5 times the cardinality of the MDS. Namely, $|MIS| \leq 5|MDS_{OPT}|$.

We now show that an MIS in the unit disk graph of the RNs is a valid solution to the unrestricted GDC problem and that its cardinality is *at most 5 times the cardinality of the optimal GDC solution.* Namely, $|MIS| \leq 5|GDC_{OPT}|$. Hence, an MIS algorithm operating on a unit disk graph provides a 5-approximation not only to the MDS problem in the unit disk graph but also to the *unrestricted* GDC problem in the plane. Notice that this relation is *not* directly implied by the above inequalities.

An MIS in the unit disk graph of the RNs is a feasible solution to the GDC problem, since all RNs are within distance r from an MBN. However, in general it is not an optimal solution. This results from the fact that for the GDC problem, MBNs can be placed anywhere in the plane. On the other hand, in the unit disk graph problem, MBNs are constrained to lie on top of RNs. As shown below the approximation ratio obtained by finding an MIS can be easily bounded.

Lemma A.4.1. *An MIS algorithm in the unit disk graph of RNs is a 5-approximation algorithm for the GDC problem.*

Proof. Let OPT and $ALGO$ represent an optimal and algorithmic GDC solutions (the algorithmic solution is an MIS). As mentioned earlier, the algorithm maintains the invariant that no two disk (MBN) centers are within distance r from each other. Similarly to [69], it can be shown that this implies that at most 5 disk *centers* can lie in a circular area of radius r. Namely, at most 5 $ALGO$ disk centers can lie inside the area covered by an OPT disk. Since all $ALGO$ disk centers are placed on top of points (RNs) that are covered by the optimal solution, all $ALGO$ disk centers are contained within *some* OPT disk. Since the number of $ALGO$ disk centers is same as the number of $ALGO$ disks, $|ALGO| \leq 5|OPT|$. □

A distributed implementation of the the Cluster Cover (CC) algorithm that finds an MIS in a unit disk graph of the RNs can be based on an algorithm developed by Baker and Ephremides [9] for clustering in a mobile wireless network. The local computation complexity of the CC algorithm is $O(1)$ since at each iteration simple decisions need to be taken. However, the time complexity (number of rounds) is $O(n)$. We note that several more efficient distributed implementations of MIS algorithms exist and can be easily adapted to our scenario.

A.5 Performance Evaluation

In this section we evaluate the performance of the algorithms via simulation. The results have been obtained by a model of our algorithms, developed in Java.

We start with the mobile RN scenario, comparing the performance of the planar GDC algorithms developed in this chapter to some of the strip-based algorithms developed in chapter 2. Figures A-5 and A-6 illustrate simulation results for a network

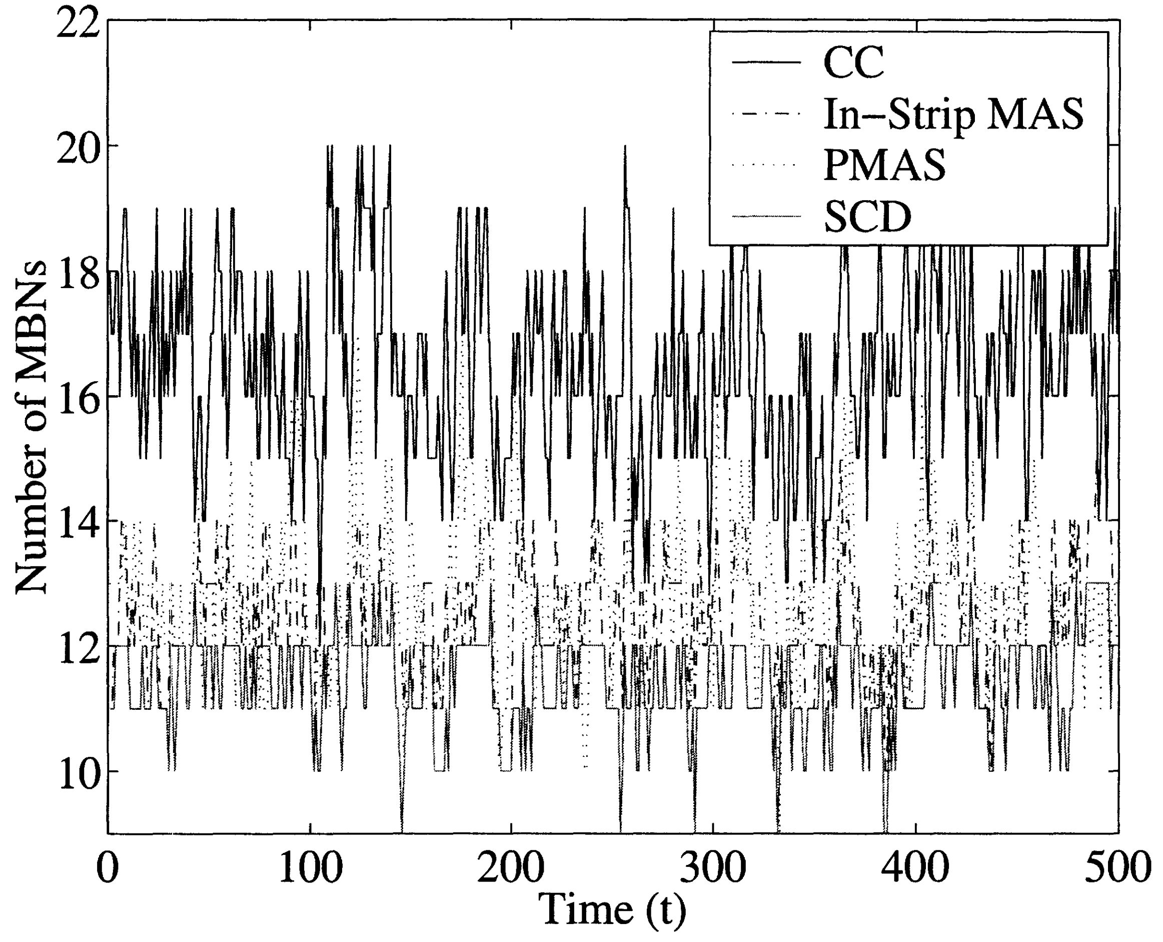

Figure A-5: The number of MBNs used by the GDC algorithms during a time period of 500s in a network of 80 RNs.

with mobile RNs. The mobility model used is the Random Waypoint Model in which RNs continually repeat the process of picking a random destination in the plane and moving there at a random speed in the range $[V_{min}, V_{max}]$. We used a plane of dimensions $600m \times 600m$, with $V_{min} = 10m/s$ and $V_{max} = 30m/s$, and set the RNs communication range as $r = 100m$. Finally, each simulation was performed for 1000s from which we discarded the first 500s.

Fig. A-5 illustrates the evolution of the algorithms over a 500s time period, with 80 RNs. It can be seen that the simplest and least computationally complex algorithm, the CC algorithm, has the poorest performance. Fig. A-6 shows the average number of MBNs used over a 500s time period as a function of the number of RNs. Each data point is averaged over 10 instances. As can be seen in the figure, when the number of RNs is low, the PMAS is the best performing algorithm. However, for a larger number of RNs, both of the strip-based algorithms perform better. The reason for this is that when the configuration of RNs is sparse, cross-strip optimization is more important, since scenarios such as those depicted in Fig. A-1 can frequently occur. By contrast, as the configuration of RNs grows more dense, MBNs will have to be used in all strips regardless. Thus, in this case, the fact that both the SCD and In-Strip MAS algorithms perform better within a strip than the PMAS explains their superior performance.

For a network with static RNs, Fig. A-7 presents the the average ratios between the solutions obtained by both the planar and strip-based algorithms, and the optimal solution. We used a plane of dimensions $1000m \times 1000m$ and set the RNs communication range as $r = 100m$. For each data point, the average was obtained over 10 different random instances in which the RNs are uniformly distributed in the plane. The optimal solutions were obtained by formulating *each instance* of the GDC problem as an Integer Program and solving it using CPLEX. From the figure, it can be seen that although the worst case performance ratios of the CC, SCR, PMAS and SCD algorithms are 5, 6, 8.5 and 4.5, their average performance ratios attained in simulation are closer to 2, 1.7, 1.5 and 1.4, respectively. Furthermore, the trend observed in the mobile scenarios, in which the PMAS outperforms the SCD for sparse RN configurations and vice versa for dense RN configurations, still holds.

Fig. A-7 also presents the upper bound on the average approximation ratios (β_{SCR} and β_{SCD}) derived in Theorem 2.4.3 in chapter 2. The large gap between the bound on the average approximation ratios and the actual ratios indicates that the bound is somewhat loose.

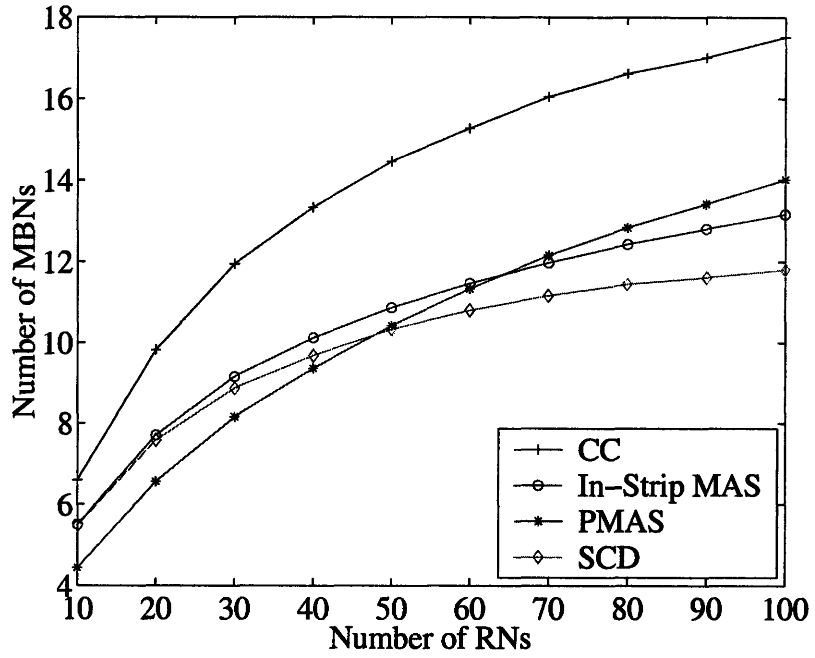

Figure A-6: The average number of MBNs used by GDC algorithms over a time period of 500s.

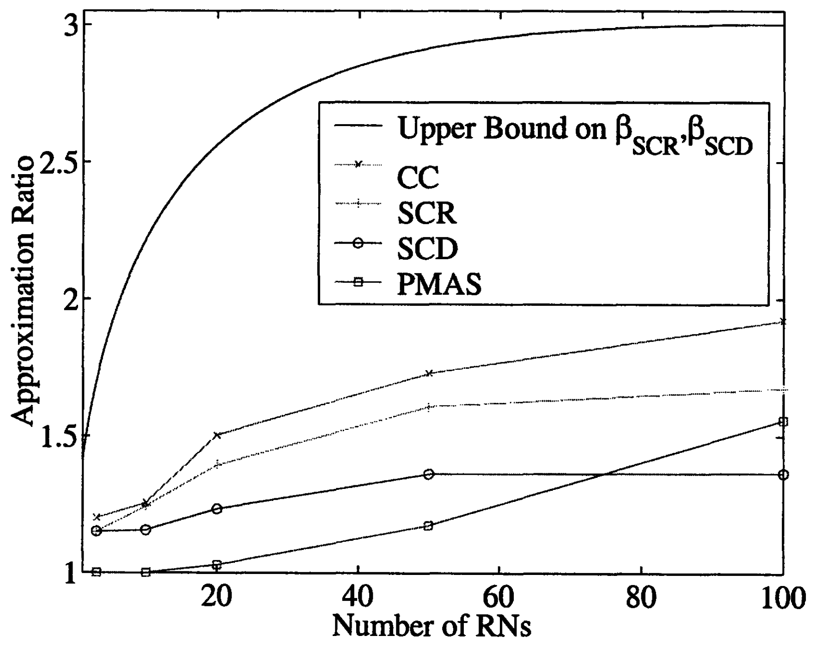

Figure A-7: Ratios between the solutions by the SCD and SCR algorithms and the optimal solution, and an upper bound on average approximation ratios.

A.6 Conclusion

The architecture of a hierarchical Mobile Backbone Network has been presented only recently. Such an architecture can significantly improve the performance, lifetime, and reliability of MANETs and WSNs. In this chapter, we concentrate on placing and mobilizing backbone nodes, dedicated to maintaining connectivity of the regular nodes. Specifically, we focus on the important subproblem of Mobile Geometric Disk Cover. We have proposed a number of *distributed* planar-based algorithms for this problem and bounded the worst case performance of two of them using a new methodology. Finally, we studied the performance under mobility via simulation.

A major future research direction is to generalize the model to other connectivity constraints and objective functions. For instance, we intend to extend the results to connectivity models that are more realistic than the disk connectivity model. Moreover, we intend to consider the energy resources and the communication requirements of the RNs when making the mobility decisions.

Appendix B

Optimal Beam Forming and Positioning for Efficient Satellite-to-Ground Broadcast

B.1 Introduction

Future satellite systems will be equipped with antenna arrays that will be capable of dynamically changing transmission beam size and position [22],[77]. This chapter addresses the problem of exploiting this beam forming and steering capability to facilitate efficient satellite-to-ground broadcast.

The satellite-to-ground broadcast problem relates to a situation where there are several users on the ground, all of whom require transmission of the same data from the satellite in a timely manner. This is a realistic model for several real-world scenarios, including commercial (e.g. TV broadcast, Teleconference) and military (e.g. aggregated intelligence data for troops) areas. In any scenario, satellite usage time is a scarce resource of prime value, governed by various factors including monetary related, political as well as logistical. Therefore, efficient management of this time is extremely important, and serves as the main motivation for the problem considered in this chapter.

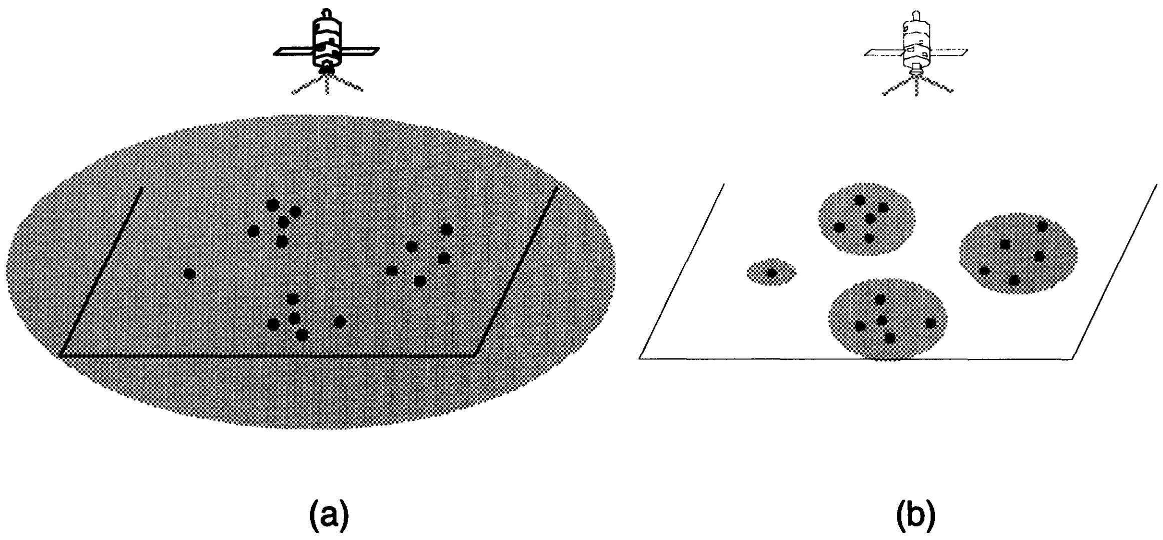
Figure B-1: Example of using beam forming and steering for satellite-to-ground broadcast. (a) Satellite using a single *low data rate* global beam to transmit to all the users. (b) Satellite using different sized *high data rate* beams in succession to transmit data to all the users.

Many current satellite systems transmit using a single *global beam* that covers all users simultaneously. For example, a GEO satellite global beam can cover a third of the globe. This is done regardless of the communication paradigm, user distribution (geographic) or required data rates. With the advent of dynamic beam forming and steering capabilities, we can significantly optimize satellite usage time and transmission capability. Figure B-1 illustrates an example of how dynamic beam forming and steering can be utilized to reduce the total time spent in transmission. Whether the scenario in B-1-a or B-1-b results in a lower total transmission time depend on two factors: (1) the data rate of a particular size (radius) transmit beam, and (2) the switching time, i.e. time it takes to change the beam size and position. We note that for an antenna-based system, the data rate increases as the size of the transmit beam decreases. This is due to the fact that the same amount of power is spread over a smaller area when the beam size is decreased, resulting in a higher signal-to-noise-ratio (SNR) at the receiver. The result of this is a higher sustainable data rate [76],[77]. Therefore, assuming the switching time is reasonably small, it is clear that depending on the users' geographic distribution, we can optimize the tradeoffs between size (data rate) of individual beams versus the number of beams, in order to minimize the total transmission time. To this end, we will formulate the Minimum Time Broadcast (MTB) problem in section B.3 and provide an approximation algorithm for it in section B.5.

The remainder of the chapter is organized as follows. We discuss previous related work in section B.2. Next, we introduce the communications model, as well as formulate the Minimum Time Broadcast (MTB) problem in section B.3. In section B.4 we present an optimal polynomial time algorithm for the 1-dimensional version of the MTB problem. Lastly, we present an approximation algorithm for the 2-dimensional MTB problem in B.5.

B.2 Related Work

The Minimum Time Broadcast (MTB) problem is related to the well studied *Geometric Disk Cover* (GDC) and *K-center* problems, but differs in a few key aspects. Below we briefly describe these two problems, and point out the subtle yet important differences between them and the MTB problem. We also describe the Conceptual Clustering problem, which more closely resembles the MTB problem than the previously mentioned two problems, yet also exhibits a key difference.

The GDC problem, discussed in great detail in Chapter 2, is also known as the *Planar Location Set Cover problem* and a variant of the *Facility Location problem* [52], [32], [88], [23], [26]. The basic problem is defined as follows: Given a set of points distributed on a plane, cover all of the points with disks of fixed radius R such that the number of disks used is minimized. The GDC is an NP-complete problem, for which several heuristic algorithms have been developed, ranging from simple greedy heuristics to more complex polynomial time approximation algorithms and schemes [52],[88],[32].

The K-center problem is also known as the K-clustering problem, the minimax radius clustering problem, and also falls under the broad umbrella of Facility Location problems [48],[93], [39],[53],[26], [19]. The basic problem is defined as follows: Given a set of demand nodes distributed on a plane, *cover* all nodes with (fixed) K disks such that the maximum radius of any of the disks is minimized. This problem is also NP-complete, and several heuristic and approximation algorithms have also been developed [38],[53].

The MTB problem differs from both the GDC and K-center problems in that variable disk radii (in contrast to the GDC) as well as a variable number of disks (in contrast to K-center) are allowed. Instead of either of these constraints the MTB aims to minimize an objective function that is defined in the next section. In this sense, the MTB can be thought of as a relaxed version of the GDC and K-center problems.

The Conceptual Clustering problem [70], [72] is more generally defined than other traditional clustering problems. In our context, i.e. where the points to be clustered are distributed on a plane, conceptual clustering methods aim to produce a clustering that is "good" based on some metric (objective function). Moreover, there exist algorithms for the conceptual clustering problem, that can find optimal solutions (in polynomial time) for a restricted class of objective functions. The most common such algorithms, the Hierarchical Agglomerative Clustering (HAC) algorithm, involves iteratively pair-wise merging the two (distance-wise) closest clusters until there remains just one cluster containing all of the points. Yet, it turns out the MTB objective as described in the next section does not fit into the class of objective functions that can be (optimally) optimized by the HAC algorithm.

B.3 Problem Formulation

We assume N users $P = \{p_1, \dots, p_N\}$ that are arbitrarily distributed on a plane, all of whom require the same data. The locations of the users are denoted by the x-y tuples (p_{i_x}, p_{i_y}) $\forall i \in P$. Transmission beams are modelled as disks. Specifically, beam

k is modelled as the center-radius tuple $[c_k, r_k]$. We say that a user j is *covered* if it is enclosed by at least one beam, i.e. $d(c_k, j) \leq r_k$ for some beam k, where $d(c_k, j)$ refers to the distance between c_k and p_j. We assume the satellite is equipped with a single transmitter. This means that transmissions to a subset of users take place via a single beam (with associated center and radius) at any time, and that this beam may need re-location and re-sizing before the beam can be trained on a different subset of users. We assume that it takes a constant amount of time, L, to re-size and re-locate the beam; we refer to L as the *beam switching time.*

The transmit data rate (in bits/sec), $b(r_k)$, of beam k is modelled as *proportional* to $\frac{1}{r_k^2}$. This is a common simplified assumption for wireless transmission. Without loss of generality, we assume the constant of proportionality to be equal to 1, i.e. $b(r_k) = \frac{1}{r_k^2}$. In this model therefore, the amount of time beam k with radius r_k needs to be held in order to transmit 1 bit, is equal to $\frac{1}{b(r_k)}$, or just r_k^2. Finally, due to physical constraints associated with any antenna system, especially satellites, the transmission beam cannot be made arbitrarily small (i.e. obtaining infinite data rate). Thus we assume transmit beams must have a minimum radius of r_0.

Based on this model, one can formulate several pertinent problems related to efficient satellite-to-ground transmission. The problem we address in this chapter is the Minimum Time Broadcast (MTB) problem, defined below. We define a feasible *beam allocation* as a set of m beams $\{[c_1, r_1], \ldots, [c_m, r_m]\}$ that cover all users, where m is variable.

Problem MTB: Given a set of users (P) distributed on a plane. Find a beam allocation $\{[c_1, r_1], \ldots, [c_m, r_m]\}$, $r_i \geq r_0$, $i = 1, \ldots, m$, such that all users are covered, and the total time required to transmit Q bits to all users,

$$T = \sum_{i=1}^{m} Q r_i^2 + (m-1)L \tag{B.1}$$

is minimized, where $Q r_i^2$ is the time for beam i to transmit Q bits.

B.4 1-Dimensional MTB Problem

The 1-dimensional MTB problem is a restricted version of the more general planar (i.e. 2-dimensional) version of the problem, where all of the points (users) are constrained to be on a line (as opposed to a plane). We assume the the users are sorted by increasing x-coordinate, e.g., $P = \{p_1, \ldots, p_N\}$, $i < j \Leftrightarrow p_{i_x} < p_{j_x}$.

We now provide an optimal algorithm for this problem. Our method is one in which we modify the Dynamic Programming approach from [17] for the 1-Dimensional K-clustering problem. To this end, we begin by defining an edgeweighted graph $G = (V, E)$, the process of which is illustrated in Fig. B-2. The vertex set V is defined as the set of tuples $\{(p_i, k)\}, i = 1, \ldots, N, k = 1, \ldots, i$. A vertex $\{(p_i, k)\}$ can be interpreted as "p_i is the leftmost user in the k^{th} beam", where the beams are also ordered from left to right. We define edges between vertices (p_i, k) and $(p_j, k+1)$, $j = 2, \ldots, N$, $i < j$, $k = 1, \ldots, i$. We interpret an edge between (p_i, k) and $(p_j, k+1)$ to indicate that "the k^{th} beam has been allocated to covers the users $\{p_i, \ldots, p_{j-1}\}$". We define the weight of an edge (p_i, k) and $(p_j, k+1)$ as the time taken to transmit to the users $\{p_i, \ldots, p_{j-1}\}$ plus a switching time, i.e.,

$$W[(p_i, k), (p_j, k+1)] = \max\{Qr_0^2, Q(\frac{p_{(j-1)_x} - p_{i_x}}{2})^2\} + L \qquad \text{(B.2)}$$

Finally, we define edges between vertices (p_i, k) and a dummy sink vertex (p_{N+1}), $i = 1, \ldots, N$,$k = 1, \ldots, i$. These edges are interpreted in a similar way as the previous edges, i.e. that the "k^{th} beam has been allocated to covers the users $\{p_i, \ldots, p_N\}$". Their weight is defined similarly as well.

The following theorem and corollary serve as the key results needed for the optimal solution.

Theorem B.4.1. *The set of paths between the vertices $(p_1, 1)$ and (p_{N+1}) enumerate all candidate optimal beam allocations.*

Proof. To prove the theorem, we first observe that an optimal beam allocation must be contiguous. To see why, consider a beam allocation that includes two beams, one that

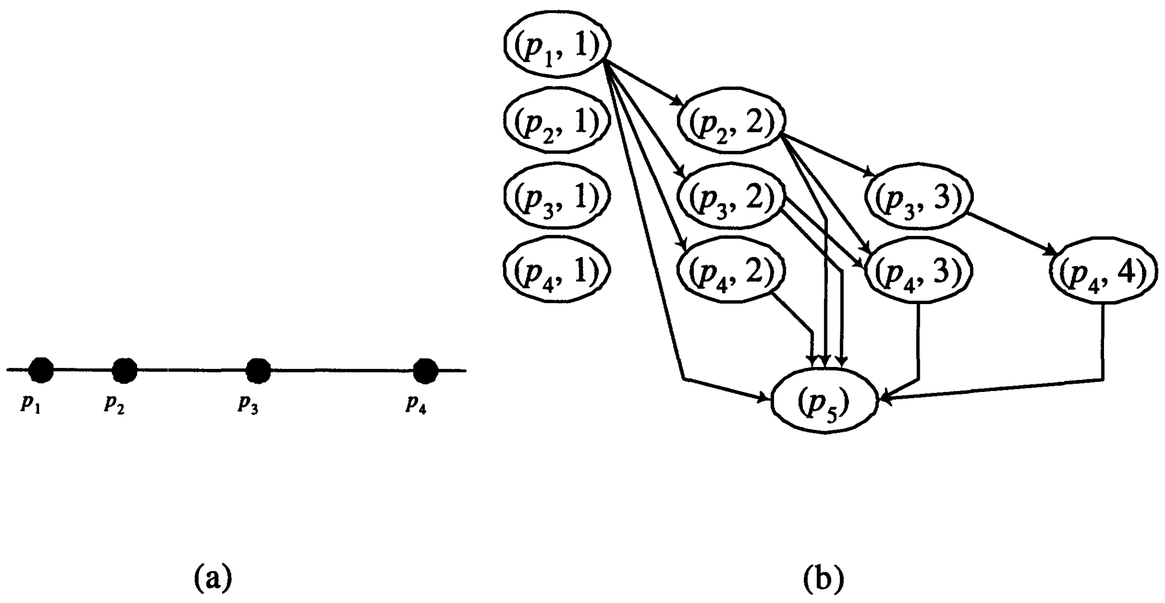

Figure B-2: Example of construction of graph G. (a) Original 1D problem instance. (b) Resultant graph G. Outgoing edges from vertices $(p_i, 1), i \geq 2$ are not shown since these would never be part of a path originating from the vertex $(p_1, 1)$.

covers users $p_a, \ldots, p_b, p_{b+c}, \ldots, p_d$ $(a \leq b < c \leq d)$, and the other that covers users $p_{b+1}, \ldots, p_{b+c-1}$. Indeed, the first beam could have covered the users covered by the second beam at no extra cost, whereas the second beam must incur some finite cost. Given this observation, consider an arbitrary optimal candidate beam allocation of m beams, covering user sets $\{p_1, \ldots, p_{k_1}\}, \{p_{(k_1+1)}, \ldots, p_{k_2}\}, \ldots, \{p_{(k_{m-1}+1)}, \ldots, p_{k_m}\}$ where p_{k_i} denotes the rightmost user covered by beam i. This can be mapped onto the path in G consisting of vertices $(p_1, 1) \rightarrow (p_{(k_1+1)}, 2) \rightarrow \ldots \rightarrow (p_{(k_{m-1}+1)}, m) \rightarrow (p_{N+1})$. Similarly, by construction of G, any path between $(p_1, 1)$ and (p_{N+1}) must correspond to a valid beam allocation. □

Corollary B.4.1. *The minimum weight path between $(p_1, 1)$ and (p_{N+1}) corresponds to the optimal MTB beam allocation.*

Proof. Consider a path consisting of m edges (i.e. corresponding to a beam allocation of m beams), $(p_1, 1) \rightarrow (p_{k_1+1}, 2) \rightarrow \ldots \rightarrow (p_{k_{m-1}+1}, m) \rightarrow (p_{N+1})$, where p_{k_i} denotes the rightmost user covered by beam i. By construction of G, the weight of this path is,

$$Weight = \sum_{i=1}^{m}(\max\{Qr_0^2, Q(\frac{p_{k_{i_x}} - p_{(k_{i-1}+1)_x}}{2})^2\} + L) \tag{B.3}$$

where $p_{(k_0+1)} \triangleq p_1$ for notational convenience. Next, note that for an arbitrary beam allocation of m beams we can re-write (B.1) as,

$$T = \sum_{i=1}^{m}(\max\{Qr_0^2, Q(\frac{p_{k_{i_x}} - p_{(k_{i-1}+1)_x}}{2})^2\} + L) - L \tag{B.4}$$

from which we can conclude the result of the corollary. □

To find the minimum weight path in G, we observe that G constitutes a Directed Acyclic Graph (DAG). Thus there exist several algorithms we can employ to obtain the minimum weight path with $O(|V| + |E|)$ computational complexity [24]. For our problem this corresponds to $O(N^3)$. Finally, the overall algorithm including the graph construction phase is given below.

Algorithm 1/2/3 1-D Beam Allocation Algorithm

```
Graph Construction:
1: Let $(G = V, E)$ represent the beam allocation graph.
2: Set $V$ as the set of tuples $\{(p_i, k)\}, i = 1, \dots, N, k = 1, \dots, i$.
3: Add to $V$ the dummy sink vertex $(p_{N+1})$
4: Define edges in $E$ between vertices $(p_i, k)$ and $(p_j, k+1)$, $j = 2, \dots, N$,
   $i < j$, $k = 1, \dots, i$. Set the weights of these edges as $W[(p_i, k), (p_j, k+1)] =$
   $\max\{Qr_0^2, Q(\frac{p_{(j-1)_x} - p_{i_x}}{2})^2\} + L$.
5: Add edges to $E$ between the vertices $(p_i, k)$ and $(p_{N+1})$, $i = 1, \dots, N, k = 1, \dots, i$.
   Set the weights of these edges as $W[(p_i, k), (p_{N+1})] = \max\{Qr_0^2, Q(\frac{p_{(N)_x} - p_{i_x}}{2})^2\} + L$.

Main Algorithm:
6: Find the minimum weight path in $G$ between $(p_1, 1)$ and $(p_{N+1})$ using an algo-
   rithm from [24]. Let $H = (p_1, 1) \to (p_{k_1+1}, 2) \to \dots \to (p_{k_{m-1}+1}, m) \to (p_{N+1})$
   denote this path of $m \geq 1$ edges.
7: Set the beam allocation $M$ as the $m$ beams covering the sets of users
   $\{p_1, \dots, p_{k_1}\}, \{p_{(k_1+1)}, \dots, p_{k_2}\}, \dots, \{p_{(k_{m-1}+1)}, \dots, p_N\}$, respectively.
8: return $M$.
```

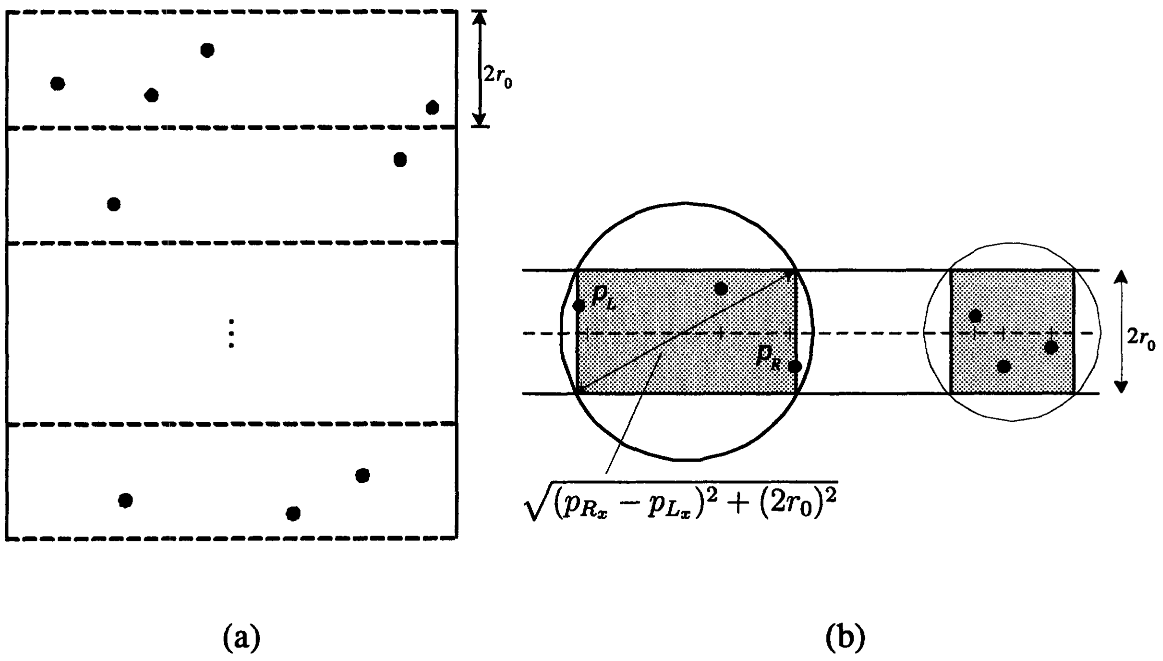

Figure B-3: Examples of dividing the plane into strips and turning the 2-D problem into a series of 1-D problems. (a) Division of the plane into strips of width $2r_0$. (b) Forcing beam centers to be located on the center line of the strip, and forcing beams to cover entire rectangular slabs of the strip.

B.5 2-Dimensional MTB Problem

The 2-dimensional MTB problem formulation was given in section B.3. To solve this problem we attempt to leverage the discussion given in the previous section. In particular, our goal is to construct an algorithm that applies the 1-dimensional optimal algorithm in 2-dimensions. We show that such an algorithm has bounded worst case performance.

To this end, we begin by dividing the plane into strips of width $D = 2r_0$, as depicted in Fig. B-3-a. The reason for this choice of strip width will become clear when we analyze the performance of the algorithm. The 2-D algorithm works by applying the 1-D algorithm to find a beam allocation for the the users in each strip independently. To facilitate this, we will force the in-strip algorithm to place beam centers on the center-line of the strip, as well as always cover full rectangular slabs of the strip, eg. as shown in Fig. B-3-b. Doing this will allow us to treat the strip-covering problem as a 1-D problem, needing only to consider the x-coordinate of each of the users within a strip. In particular, as shown in Fig. B-3-b, to cover a set of users with leftmost user p_L and rightmost user p_R, we force the 1-D algorithm to use a beam of radius $\frac{\sqrt{(p_{R_x}-p_{L_x})^2+(2r_0)^2}}{2}$. A complete description of the 2-D algorithm is given below.

Algorithm 1/2/3 2-D Beam Allocation Algorithm

```
Initialization:
1: Divide The plane into $Z$ strips $S_1, S_2, \ldots, S_Z$
2: for $i = 1$ to $Z$ do
3:    Execute the 1-D algorithm on the users in strip $S_i$ treating the users in $S_i$
      as located on their projection onto the center line. Also, in line 5 of the 1-D
      algorithm change $W[]$ to $W[(p_i, k), (p_{N+1})] = \max\{Qr_0^2, Q(\frac{(p_{(N)_x}-p_{i_x})^2+(2r_0)^2}{4})\} + L$.
4:    Let $M_{S_i}$ denote the resultant beam allocation.
5: return $M = \bigcup_{i=1}^{Z} M_{S_i}$.
```

The following theorem shows the worst case performance of the 2-D algorithm as compared to the optimal beam allocation.

Theorem B.5.1. *In the worst case, the cost of the beam allocation found by the 2-D algorithm is at most* $(8+\frac{3L}{Qr_0^2})$*-times the optimal beam allocation.*

To prove the above theorem, we will employ the following methodology. First, for any instance of users, we will construct a candidate beam allocation that allocates beams to users on different strips independently. Additionally, we will force the candidate allocation to cover entire intervals of the strip. Given these restrictions, by the discussion in section B.4 we can conclude that the 1-D algorithm must outperform this candidate allocation *within each strip*. Therefore, the solution found by the 2-D algorithm which uses the 1-D algorithm in each strip must outperform the overall candidate beam allocation solution. It follows that any performance bound we show for the candidate allocation must hold for the overall 2-D algorithm.

To further proceed, we first need the following observation, analogous to observation 2.4.1 from chapter 2.

Observation B.5.1. *An optimal beam of radius q will contain users from at most $\lceil \frac{q}{r_0} \rceil + 1$ different strips.*

Proof of theorem B.5.1. We construct a candidate beam allocation M as follows. Consider a single optimal beam OPT with center c and radius q, illustrated in Fig. B-4. Next, assume $kr_0 \leq q \leq (k+1)r_0$, for an integer $k \geq 1$. We consider two cases, the first assuming $k \geq 2$. In this case, we can upper bound the users covered the optimal beam by a square with center c and side length $2q$ as depicted in the figure. Next, we compactly cover this square area with $(\lceil \frac{q}{r_0} \rceil + 1)(\lceil \frac{q}{r_0} \rceil)$ candidate beams that cover square slabs of the strip of side length $2r_0$ (i.e. of radius $\sqrt{2}r_0$). The reason this can be done follows from observation B.5.1, and is depicted in Fig. B-4-a. Thus for such a scenario we have that,

$$\begin{aligned}
\frac{Cost(M)}{Cost(OPT)} &= \frac{(k+2)(k+1)Q(\sqrt{2}r_0)^2 + [(k+2)(k+1)-1]L}{Qq^2} \\
&\leq \frac{2(k^2+3k+2)Qr_0^2 + (k^2+3k+1)L}{k^2Qr_0^2} \\
&= 2(1+\frac{3}{k}+\frac{2}{k^2}) + (1+\frac{3}{k}+\frac{1}{k^2})\frac{L}{Qr_0^2} \\
&\leq 6+3\frac{L}{Qr_0^2} \qquad \text{(B.5)}
\end{aligned}$$

the last line follows from the assumption that $k \geq 2$, and $Cost()$ refers to the cost function in (B.1). Next, assume that $k = 1$, i.e. $r_0 \leq q \leq 2r_0$. Simply substituting $k = 1$ into B.5 while valid yields a very loose performance upper bound of $12 + 5\frac{L}{Qr_0^2}$. We can tighten this bound by noting that for $r_0 \leq q \leq 2r_0$, there are three cases for the candidate beam M, illustrated in Fig. B-4. In particular, if $q = r_0 + \epsilon$, ϵ denoting a very small value, then the optimal beam will just barely cover users from three different strips as shown in Fig B-4-b. Thus in this case, the candidate M will only need at most 4 beams of radius $\sqrt{2}r_0$ to cover these users. This yields,

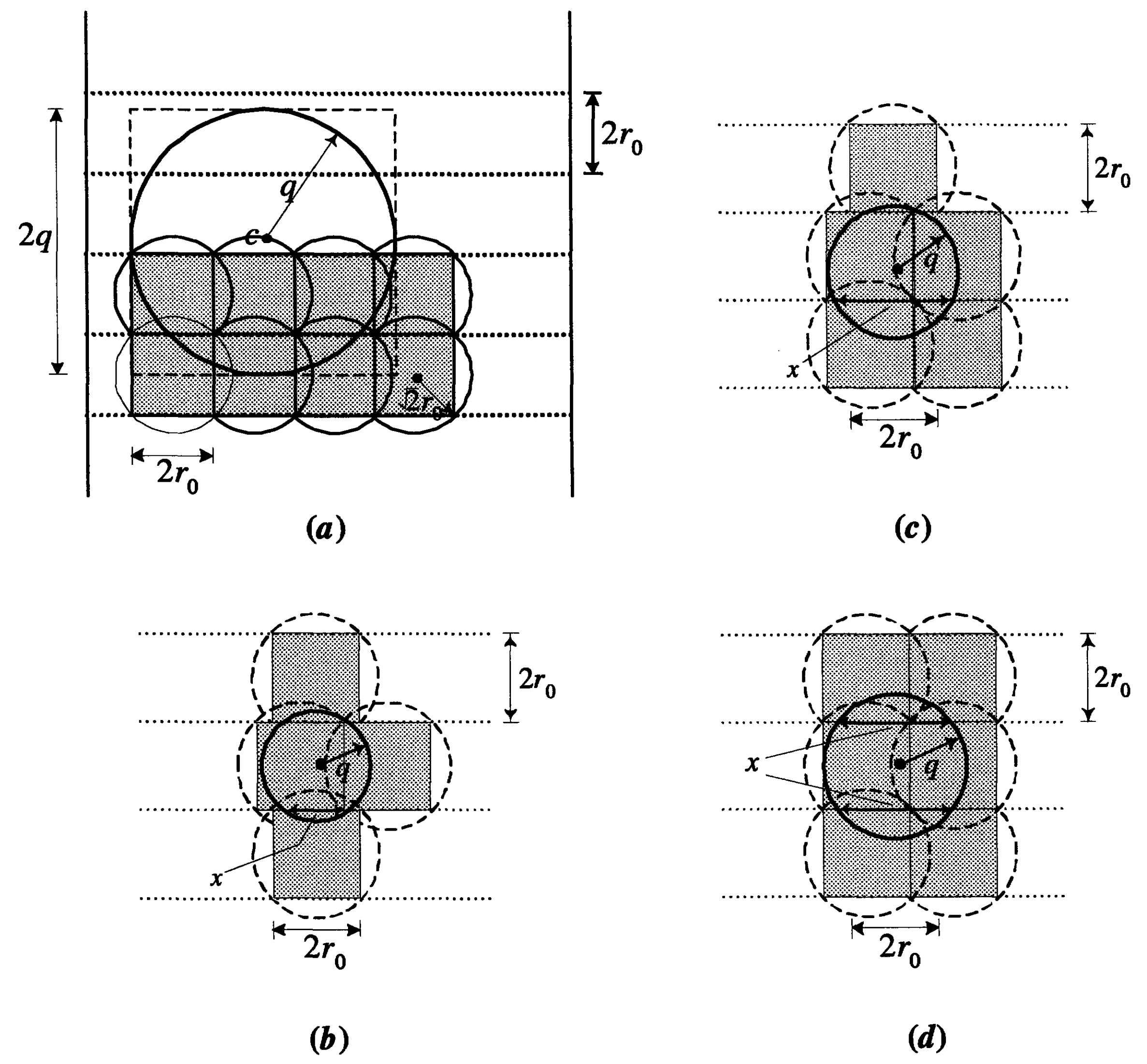

Figure B-4: Illustrations of the proof of theorem B.5.1. (a) Covering an optimal beam of radius q with candidate beams of radii $\sqrt{2}r_0$. (b) Case 1: $q = r_0 + \epsilon$. Note that the length of intersection of the optimal beam with the bottom (and top) strip is $x < 2r_0$. (c) Case 2: $q > \frac{5}{4}r_0$. The length of intersection of the optimal beam with the bottom strip is $x > 2r_0$. (d) Case 3: $q > \sqrt{2}r_0$. The length of intersection of the optimal beam with the top *and* bottom strips is $x > 2r_0$.

$$\begin{aligned}\frac{Cost(M)}{Cost(OPT)} &\leq \frac{4Q(\sqrt{2}r_0)^2+3L}{Qr_0^2} \\ &= 8+3\frac{L}{Qr_0^2}\end{aligned} \tag{B.6}$$

which will turn out to be the worst case. To see this, consider the second case wherein $q \geq \frac{5}{4}r_0$ and therefore the optimal beam again covers users from three strips but now covers users from a long enough interval (i.e. more than length $2r_0$) on a second strip to force the candidate solution to have to use 5 total beams of radius $\sqrt{2}r_0$. This is shown in Fig. B-4-c. This yields,

$$\begin{aligned}\frac{Cost(M)}{Cost(OPT)} &\leq \frac{5Q(\sqrt{2}r_0)^2+4L}{Q(\frac{5}{4}r_0)^2} \\ &\leq \frac{32}{5}+\frac{64}{25}\frac{L}{Qr_0^2}\end{aligned} \tag{B.7}$$

which is smaller than $8+3\frac{L}{Qr_0^2}$. The third case, depicted in Fig. B-4-d, is the case that $q \geq \sqrt{2}r_0$ in order to force the candidate solution to utilize 6 beams of radius $\sqrt{2}r_0$. This case yields,

$$\begin{aligned}\frac{Cost(M)}{Cost(OPT)} &\leq \frac{6Q(\sqrt{2}r_0)^2+5L}{Q(\sqrt{2}r_0)^2} \\ &\leq 6+\frac{5}{2}\frac{L}{Qr_0^2}\end{aligned} \tag{B.8}$$

which is also smaller than $8+3\frac{L}{Qr_0^2}$. Finally, since this procedure can be applied to all optimal disks, we obtain the result of the theorem. $\square$

It should be noted that for a normal problem instance, the performance bound proved in theorem B.5.1 is still quite loose. For instance, in most instances the 2-D algorithm will be able to optimize the solution within a strip much better than naively covering the area covered by the optimal beam with $2r_0 \times 2r_0$ squares. Additionally, for a given optimal beam, it may not necessarily be the case that the users it covers lie in every strip that it comes in contact with. It is also possible that dividing the strip into different sized strips could have yielded a tighter approximation ration. We have not pursued this in this chapter. We also should note that the solution found by the 2-D algorithm can be further improved by removing redundant beams from different strips that actually cover the same sets of users, as well as moving beam centers to the 1-center locations of the users that they cover as a final step.

Finally, we note that if the ratio $\frac{L}{Qr_0^2}$ is large then the 2-D algorithm presented in this section will in general have very poor performance even with the above improvements. Indeed, in this scenario any algorithm that uses more beams than the optimal solution would have poor performance. Thus in this case an alternative heuristic (e.g. non-strip-based) should be employed.

A loose bound on the computational complexity of the 2-D algorithm is $O(N^4)$. This comes from simply multiplying the total number of nodes multiplied by the computational complexity of the 1-D algorithm discussed in section B.4.

B.6 Conclusion

Future satellite systems will be equipped with antenna arrays that will be capable of dynamically changing transmission beam size and position. In this chapter we have addressed the problem of exploiting this beam forming and steering capability to facilitate efficient satellite-to-ground broadcast. To this end we have formulated the Minimum-Time Broadcast (MTB) problem which chooses the optimal transmit beam allocation so as to minimize the broadcast time to set of users on the ground. If all of the users are located on a line, we have provided an optimal polynomial time algorithm. We have used this 1-dimensional algorithm as a subroutine for an approximation algorithm for the general 2-dimensional MTB problem.

Future work includes the development of algorithms with better approximation ratios. Additionally, in practice transmit beams are not exactly circles, and thus the formulation needs to be expanded to facilitate arbitrary shaped beams.

Bibliography

[1] P. Agarwal, M. Pellegrini and M. Sharir, "Counting circular arc intersections," *Siam J. Computing*, 22, pp. 778-793, 1993.

[2] P. Aggarwal and M. Sharir, "Efficient Algorithms for Geometric Optimization," *ACM Comput. Surveys*, 30, pp. 412-458, 1998.

[3] R. Ahuja, T. Magnanti, and J. Orlin, "Network Flows," Prentice Hall, New Jersey, 1993.

[4] E. Amaldi, A. Capone, and F. Malucelli, "Discrete models and algorithms for the capacitated location problems arising in UMTS network planning," *ACM DIAL-M'01*, July 2001.

[5] A.D. Amis, R. Prakash, D. Huynh and T. Vuong, "Max-Min d-cluster formation in wireless ad hoc networks," *Proc. IEEE INFOCOM '00*, pp. 32 41, 2000.

[6] S. Arora, "Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems," *J. ACM*, 45, 5, pp. 753-782, 1998.

[7] M. Athans, P. Falb, "Optimal control: an introduction to the theory and its applications," McGraw-Hill, New York, 1966.

[8] D. J. Bell, "Singular problems in optimal control - A survey," *Int. J. Control*, 21, 2, pp. 319-331, 1975.

[9] D.J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Trans. Commun.*, 29, 11, pp. 1694-1701, 1981.

[10] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, "Localized protocols for ad hoc clustering and backbone formation: a performance comparison," *IEEE Trans. on Parallel and Dist. Systems*, 17, 4, pp. 292-306, 2006.

[11] S. Basagni, "Distributed Clustering for Ad Hoc Networks," *Proc. I-SPAN'99*, June 1999.

[12] S. Bereg, B. Bhattacharya, D. Kirkpatrick, M. Segal, "Competetive Algorithms for Maintaining a Mobile Center," *Mobile Networks and Applications*, 11, pp. 177-186, 2006.

[13] M. Bern and P. Plassmann, "The steiner problem with edge lengths 1 and 2," *Inf. Proc. Letters*, 32, pp. 171-176, 1989.

[14] L. Booth, J. Bruck, M. Franceschetti, and R. Meester, "Covering Algorithms, Continuum Percolation and the Geometry of Wireless Networks", *Ann. Appl. Probab.* 13, 2 pp.722741, 2003.

[15] P. Bose and G. Toussaint, "Computing the constrained euclidean geodesic and link centre of a simple polygon with applications," *Proc. of Pacific Graphics International*, 1996.

[16] J. Bredin, E. Demaine, M. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," In *Proc. ACM MOBIHOC'05*, 2005.

[17] P. Brucker, "On the Complexity of Clustering Problems," *Optimization and Operations Research*, Springer Verlag, pp. 45-54, 1977.

[18] A. E. Bryson Jr. and Y.C. Ho, "Applied optimal control: Optimization, Estimation, and Control," Blaisdell, Waltham, MA, 1969.

[19] V. Capoyleas, G. Rote and G. Woeginger, "Geometric Clusterings," in *J. Algorithms*, 12, pp. 341-356, 1991.

[20] D. Chen, D.-Z. Du, X.-D. Hu, G.-H. Lin, L. Wang, and G. Xue, "Approximations for steiner trees with minimum number of steiner points," *J. Global Optimization*, 18, 1, pp. 17-33, 2000.

[21] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," To appear in *ACM/Springer WINET*, 2007.

[22] J.P. Choi and V.W.S. Chan, "Optimum multibeam satellite downlink power allocation based on traffic demands," *IEEE GLOBECOMM'02*, 2002.

[23] R. Church, "The Planar Maximal Covering Location Problem," *Journal of Regional Science*, 24, 2, pp. 185-201, 1984.

[24] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "Introduction to Algorithms," McGraw Hill, 2001.

[25] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in ad-hoc networks using a virtual backbone," in *Proc. IEEE ICCCN'97*, Sept. 1997.

[26] Z. Drezner and H.W. Hamacher, "Facility Location, Applications and Theory," Springer-Verlag, Berlin, 2002.

[27] Z. Drezner and G.O. Wesolowsky, "Facility location when demand is time dependent," *Naval Res Log*, 38, pp. 263277, 1991.

[28] D-Z. Du, F. Hwang, "A Proof of Gilbert-Pollak's Conjecture on the Steiner Ratio," *Algorithmica*, 7, pp. 121-135, 1992.

[29] D. Eppstein, "Faster construction of planar two-centers," *Proc. ACM-SIAM SODA'97*, pp. 131-138, 1997.

[30] D. Erlenkotter, "A comparative study of approaches to dynamic location problems," *Eur. J. Oper. Res.*, 6, pp. 133143, 1981.

[31] Y. Fernandess and D. Malkhi, "K-clustering in wireless ad-hoc networks," *Proc. ACM POMC 2002*, pp. 3137, 2002.

[32] M. Franceschetti, M. Cook, and J. Bruck, "A geometric theorem for approximate disk covering algorithms," *Technical Report ETR035*, Caltech, 2001.

[33] H. Gabow, "Scaling algorithms for network problems," *J. Comput. Syst. Sci.*, 31, pp. 148-168, 1985.

[34] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Discrete mobile centers," *Disc. Comput. Geom.*, 30, 1, pp. 45-63, 2003.

[35] M.R. Garey and D.S. Johnson, *Computers and Intractibility.* Freeman, 1979.

[36] R.G. Gallager, P.A. Humblet, and P.M. Spira, "A distributed algorithm for minimum weight spanning trees," *ACM Trans. Prog. Lang. Syst.*, 5, 1, pp. 66-77, 1983.

[37] M. Gerla and J. T. Tsai, "Multicluster, mobile, multimedia radio network," *ACM/Kluwer Wireless Networks*, 1, 3, pp. 255-265, 1995.

[38] T. Gonzalez, "Covering a set of points in multidimensional space," *Inf. Proc. Letters*, 40, 4, pp. 181-188, 1991.

[39] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, 38, pp. 293-306, 1985.

[40] D. Gu, G. Pei, M. Gerla, and X. Hong, "Integrated hierarchical routing for heterogeneous multi-hop networks," in *Proc. IEEE MILCOM'00*, 2000.

[41] S. Guha and S. Khuller, "Improved methods for approximating node weighted steiner trees and connected dominating sets," *Inf. Comput.*, 150, 1, pp. 57-74, 1999.

[42] H. Gupta, S. Das, and Q. Gu, "Connected sensor cover: self organization of sensor networks for efficient query execution," in *Proc. ACM MOBIHOC'03*, June 2003.

[43] A. Gupta, A. Kumar, and T. Roughgarden, "Simpler and Better Approximation Algorithms for Network Design," *Proc. STOC'03*, 2003.

[44] S. L. Hakimi, M. Labbe and E. Schmeichel, "Locations on Time-Varying Networks," *Networks*, 34, 4, pp. 250-257, 1999.

[45] S. Hanly, "An Algorithm for Combined Cell-Site Selection and Power Control to Maximize Cellular Spread Spectrum Capacity," *IEEE J. Select. Areas Commun.*, 13, pp. 1332-1340, 1995.

[46] S. Har-Peled, "Clustering Motion," Discrete and Computational Geometry, 31, 4, pp. 545-565, Mar. 2004.

[47] J. Hershberger, "Smooth kinetic maintenance of clusters," in *Proc. ACM SoCG'03*, June 2003.

[48] S.L. Hakimi, "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems," *Operations Research*, 13, pp. 462-475, 1965.

[49] S. Haykin, "Communication Systems," 4th Ed., Wiley, 2001.

[50] H. Maurer, "Numerical solution of singular vontrol problems using multiple shooting techniques," *J. Opt. Theory and Appl.*, 18, 2, pp. 235-257, 1976.

[51] J. Hershberger and S. Suri, "Finding tailored partitions," *J. Algorithms*, 12, 3, pp. 431-463, 1991.

[52] D.S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *J. ACM*, 32, 1, pp. 130-136, 1985.

[53] D.S. Hochbaum and D.B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operational Research*, 10, 2, pp. 180-184, 1985.

[54] H. Huang, A. Richa, and M. Segal, "Approximation algorithms for the mobile piercing set problem with applications to clustering in ad-hoc networks," *ACM/Kluwer MONET*, 9, 2, pp. 151-161, 2004.

[55] R. Hwang, R.C.T. Lee and R.C. Chang, "The slab dividing approach to solving the euclidean p-center problem," *Algorithmica*, 9, pp. 1-22, 1993.

[56] S. K. Jacobsen, "An algorithm for the minimax weber problem," *Eur. J. Oper. Res.*, 6, pp. 144-148, 1981.

[57] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing, eds: T. Imielinski and H. Korth*, Ch. 5, pp. 153-181, Kluwer, 1996.

[58] A. Kansal, M. Rahimi, D. Estrin, W.J. Kaiser, G. J. Pottie, and M. B. Srivastava, "Controlled mobility for sustainable wireless sensor networks," in *Proc. IEEE SECON'04*, Oct. 2004.

[59] S. Khuller and A. Zhu, "The General Steiner Tree-Star problem," *Inf. Process. Lett.*, 84, 4, pp. 215-220, 2002.

[60] P. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted steiner trees," *J. Algorithms*, 19, 1, pp. 104-115, 1995.

[61] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing newly deployed ad hoc and sensor networks," in *Proc. ACM MOBICOM'04*, 2004.

[62] B. Liang and Z. J. Haas, "Virtual backbone generation and maintenance in ad hoc network mobility management," in *Proc. IEEE INFOCOM*, pp. 12931302, 2000.

[63] C. R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.*, 15, 7, pp. 1265-75, 1997.

[64] G. Lin and G. Xue, "Steiner tree problem with minimum number of steiner points and bounded edge-length," *Inf. Proc. Letters*, 69, 2, pp. 53-57, 1999.

[65] M. Lu, J. Wu, M. Cardei, and M. Li, "Energy-efficient connected coverage of discrete targets in wireless sensor networks," in *Proc. ICCNMC'05*, Aug. 2005.

[66] J. Luo and J.-P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *Proc. IEEE INFOCOM'05*, 2005.

[67] I. Mandoiu and A. Zelikovsky, "A Note on the MST Heuristic for Bounded Edge-Length Steiner Trees with Minimum Number of Steiner Points," *Inf. Proc. Letters*, 75, 4, pp. 165-167, 2000.

[68] R. Mathar and T. Niessen, "Optimum positioning of base stations for cellular radio networks," *Wireless Networks*, 6, pp. 421428, 2000.

[69] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, "Simple Heuristics for Unit Disk Graphs," *Networks*, 25, pp. 59-68, 1995.

[70] R.S. Michalski, "Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts," *International Journal of Policy Analysis and Information Systems*, 4, pp. 219-243, 1980.

[71] A. Orda and R. Rom, "Location of central nodes in time varying computer networks," *Oper Res Lett*, 10, pp. 143152, 1991.

[72] L. Pitt, R. E. Reinke, "Criteria for Polynomial Time (Conceptual) Clustering," *Machine Learning*, 2, pp. 371-396, 1988.

[73] F. Preparata and M. Shamos, "Computational Geometry: An Introduction," Spinger-Verlag, New York, 1985.

[74] J.S. Provan, "Convexity and the steiner tree problem," *Networks*, 18, 1, pp. 55-72, 1988.

[75] R. Rao and G. Kesidis, "Purposeful mobility for relaying and surveillance in mobile ad hoc sensor networks," *IEEE Trans. Mob. Comp.*, 3, 3, pp. 225-231, 2004.

[76] T.S. Rappaport, *Wireless Communications: Principles and Practices*, Prentice Hall, 1996.

[77] Rizwan Mustafa Mir, "Satellite Data Networks," *Online Report*, 1997.

[78] T.J. Van Roy and D. Erlenkotter, "A dual-based procedure for dynamic facility location, *Mgmt Sci*, 28, pp. 10911105, 1992.

[79] I. Rubin, A. Behzad, R. Zhang, H. Luo, and E. Caballero, "TBONE: a mobile-backbone protocol for ad hoc wireless networks," in *Proc. IEEE Aerospace Conf.*, Mar. 2002.

[80] A. Segev, "The node-weighted steiner tree problem," *Networks*, 17, 1, pp. 1-17, 1987.

[81] M. A. Soliman and W.H. Ray, "A computational technique for optimal control problems having singular arcs," *Int. J. Control*, 16, 2, pp. 261-271, 1972.

[82] A. Srinivas, G. Zussman, and E. Modiano, "Construction and maintenance of a mobile backbone for wireless networks," *MIT/LIDS Technical Report #2676*, Oct. 2005.

[83] A. Srinivas, G. Zussman, and E. Modiano, "Mobile Backbone Networks: Construction and Maintenance," *ACM MOBIHOC'06*, May 2006.

[84] A. Srinivas and E. Modiano, "Joint node placement and assignment for throughput optimization in mobile backbone networks," Submitted, 2007.

[85] D. Stamatelos and A. Ephremides, "Spectral Efficiency and Optimal Base Placement for Indoor Wireless Networks," *IEEE Journal on Selected Areas in Communications*, 14, 4, May 1996.

[86] C. Swamy and A. Kumar, "Primal-dual algorithms for connected facility location problems," *Algorithmica*, 40, 4, pp. 245-269, 2004.

[87] J. Tang, B. Hao, and A. Sen, "Relay node placement in large scale wireless sensor networks," *Computer Commun.*, 29, 4, pp. 490-501, 2006.

[88] C. Toregas, R. Swain. C. Revelle and L. Bergman, "The Location of Emergency Service Facilities," *Operations Research*, 19, pp. 1363-1373, 1971.

[89] K. Tutschku, "Demand-based Radio Network Planning of Cellular Mobile Communication Systems," *IEEE INFOCOM'98*, 1998.

[90] E. Weiszfeld, "Sur le Point pour Lequel la Somme des Distances de n Points Donn'ees est Minimum," *Tohoku Math J.*, 43, pp. 355-386, 1937.

[91] G.O. Wesolowsky, "Dynamic facility location," *Mgmt Sci*, 19, pp. 12411248, 1973.

[92] G.O. Wesolowsky and W.G. Truscott, The multiperiod location- allocation of facilities, *Mgmt Sci*, 22, pp. 5765, 1975.

[93] J. White and K. Case, "On Covering Problems and the Central Facilities Location Problem," *Geographical Analysis*, 6, pp. 281-293, 1974.

[94] K. Xu, X. Hong, and M. Gerla, "Landmark routing in ad hoc networks with mobile backbones," *J. Parallel Distrib. Comput.*, 63, 2, pp. 110-122, 2003.