# Cooperative Exploration under Communication Constraints

by

Emily M. Craparo

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Aeronautics and Astronautics
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2008





Author .....
Department of Aeronautics and Astronautics
August 8, 2008

Certified by .....
Jonathan P. How
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by ..
Eytan Modiano
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .
Nicholas Roy
Assistant Professor of Aeronautics and Astronautics

Accepted by.....
David Darmofal
Chairman, Department Committee on Graduate Students

# Cooperative Exploration under Communication Constraints

by

Emily M. Craparo

Submitted to the Department of Aeronautics and Astronautics
on August 8, 2008, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Aeronautics and Astronautics

## Abstract

The cooperative exploration problem necessarily involves communication among agents, while the spatial separation inherent in this task places fundamental limits on the amount of data that can be transmitted. However, the impact of limited communication on the exploration process has not been fully characterized. Existing exploration algorithms do not realistically model the tradeoff between expansion, which allows more rapid exploration of the area of interest, and maintenance of close relative proximity among agents, which facilitates communication.

This thesis develops new algorithms applicable to the problem of cooperative exploration under communication constraints. The exploration problem is decomposed into two parts. In the first part, cooperative exploration is considered in the context of a hierarchical communication framework known as a mobile backbone network. In such a network, mobile backbone nodes, which have good mobility and communication capabilities, provide communication support for regular nodes, which are constrained in movement and communication capabilities but which can sense the environment. New exact and approximation algorithms are developed for throughput optimization in networks composed of stationary regular nodes, and new extensions are formulated to take advantage of regular node mobility. These algorithms are then applied to a cooperative coverage problem.

In the second part of this work, techniques are developed for utilizing a given level of throughput in the context of cooperative estimation. The mathematical properties of the information form of the Kalman filter are leveraged in the development of two algorithms for selecting highly informative portions of the information matrix for transmission. One algorithm, a fully polynomial time approximation scheme, provides provably good results in computationally tractable time for problem instances of a particular structure. The other, a heuristic method applicable to instances of arbitrary matrix structure, performs very well in simulation for randomly-generated problems of realistic dimension.

Thesis Supervisor: Jonathan P. How
Title: Professor of Aeronautics and Astronautics

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

# Acknowledgments

As I look back on a decade spent at MIT, I am amazed at the number of wonderful and fascinating people I have met. I am able to acknowledge only a small fraction in this document, but I am indebted to many more.

My first words of gratitude go to my thesis committee: Professors Jonathan How, Eytan Modiano, and Nicholas Roy. At the top of that list is my advisor, Jon How. I've always thought of myself as a straight shooter, but Jon gives new meaning to the words. His forthrightness, tempered by his humor and patience (but not *too much* patience), were instrumental in moving my work forward. I was very fortunate to have Eytan Modiano on my committee. Eytan's level of involvement in my work gradually grew to that of a co-advisor, and I thoroughly enjoyed working with him at every stage. The fact that I was able to make contributions in the area of network optimization despite coming from a controls background is evidence of Eytan's clear communication style and excellent pedagogical skills. I have learned a great deal from him, and I hope he has learned a bit from me as well. Nick Roy rounded out my my thesis committee and provided a robotics perspective for my work. I enjoyed Nick's high-energy meetings and thank him for his words of motivation during the occasional bout of pre-generals navel gazing.

I would also like to thank Daniela Pucci de Farias for serving on my committee before deciding to leave MIT. Although she was not around long enough to see my work completed, Daniela provided critical early support and ideas for problem formulations. She was a fantastic researcher, teacher and mentor. The engineering community's loss is the tango community's gain.

My thesis readers, John Deyst and Emilio Frazzoli, both provided thoughtful and insightful comments that improved my thesis document. However, their overall contribution to my career is more significant: John was one of my first instructors at MIT and gave me a solid foundation for my future work. Emilio was the teaching assistant in my first controls course and inspired me to continue working in the area. I thank John and Emilio both for their recent efforts and for all of their contributions along the way.

Special thanks go to Karen Willcox and Andreas Schulz, who very generously shared

their time with me despite the fact that I was not their student. Thanks to Karen for several career mentoring conversations and for being a friendly face in the audience of my talks. Thanks to Andreas for allowing me to bounce research ideas off him, for providing useful and insightful feedback on these ideas, and for his general collegiality.

I suspect that very little research would be done without the skills of such people as Barbara Lechner and Kathryn Fischer. Barbara is a true advocate of students and does what it takes to give each student both logistical support and moral support. Kathryn is an administrative assistant extraordinaire who goes above and beyond her duties to make sure the students of ACL have everything they need.

My final year at MIT was generously funded by the Department of Aeronautics and Astronautics, and I am extremely grateful for their support. My research was also supported by fellowships from the American Society for Engineering Education, the National Science Foundation, and Zonta International; and by NSF grant CCR-0325401 and AFOSR grant FA9550-04-1-0458.

I have greatly enjoyed interacting with a number of labmates over the years. Anand Srinivas was kind enough to take the time to explain his work to me so that I could build upon it. Animesh Chakravarthy was one of my favorite colleagues in LIDS, and he continues to be a friend from his new location in my hometown. Louis Breger is possibly my oldest acquaintance at MIT and helped facilitate my transition to ACL, for which I am extremely grateful. Luca Bertuccelli and Han-Lim Choi have provided valuable feedback on my work, even as they were completing their own work. And my MIT experience would not have been the same without great labmates such as Masha Ishutkina, Brett Bethke, Jan de Mot, Danielle Tarraf, Mehdi Alighanbari, Yoshi Kuwata, Justin Teo, Sommer Gentry, Ji Hyun Yang, Tom Shouwenaars, Gaston Fiore, and Georges Aoude.

When not in the lab, I have greatly benefited from interacting with a number of fascinating individuals from other MIT departments and from around Boston, most of whom I met through Union Boat Club and the MIT Rowing Club. There are far too many to list, but some of the people I've most enjoyed meeting are Stephanie Torta, Helyn Wynyard, Ulrike Baigorria, Pam Chang, Meghan Backhouse, Alison Skelley, Kaity Ryan, Michelle Miilu, Yuhua Hu, Carroll Williamson, Cara LaPointe, Anna Custo, AliciA Hardy, Gene Clapp,

Ryan Tam, Tom Bohrer, Laura Donohue, Aleks Zosuls, and Tony Philippi.

Finally, I would like to thank my parents, who gave me the foundation I needed to succeed (as well as plenty of support along the way), and Jon Moskaitis, my best friend of the past five years, who links my time spent at MIT to my future. I dedicate this thesis to them.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

A fundamental issue facing developers of autonomous mobile robots is the necessity of operating in *a priori* unknown environments. The ability to autonomously explore an unknown environment, build a map of this environment and, if necessary, estimate the robot's own location within that map is regarded as primary prerequisites for fully autonomous robots. Two aspects of this challenge are captured by the *exploration* and *coverage* problems [13], and by the well-known *simultaneous localization and mapping* (SLAM) problem [55].

In the coverage problem, stationary or mobile robots are deployed with the goal of maximizing the area covered by the robots' sensors [7]. A closely related problem is that of exploration, in which mobile robots seek to completely cover a given environment [7], generally in a minimum amount of time or with a minimal amount of sensor overlap. The exploration problem can also involve *discovering* areas that have not been covered (as opposed to such areas being given as input).

In the SLAM problem, the goal is not only to cover an unexplored area, but also to estimate the locations of objects within this area and if necessary, the robot's own location. Objects are generally represented either as *features* in the environment, or are described implicitly through a *grid* representation of the world, in which each grid space can either be occupied by an object, or unoccupied.

Both the exploration and mapping problems have have been examined in the context of cooperative multi-robot teams. Such teams offer many potential benefits in terms of effi-

ciency and robustness. However, interference between robots can sometimes cause robotic teams to perform their collective tasks *more slowly* than an individual robot could [46], while the computation time required to coordinate and assign tasks to multiple robots is often quite high. (Interference within a team of robots has been studied formally by Sharma et al. in [45].) Thus, scalability is an important characteristic in algorithms for cooperative mobile robot teams [52]. The magnitude of problem parameters and the number of robots available to perform tasks are both unknown *a priori* in many problems of interest [31]. Without the ability to adapt a solution structure to problem instances of practical scale, the full benefit of using multiple robots is not realized. The need to create cooperative algorithms that scale well with the size of a robot team and with the magnitude of problem parameters (e.g. the number of features in a mapping problem) has led to a widespread interest in decentralized and approximate algorithms for many problems [10, 14, 33].

In feature-based exploration and mapping problems, true scalability requires that a large number of robotic agents be able to cooperatively explore and estimate the location of a large number of features spread over a large area. Large numbers of agents rely on efficient, polynomial-time algorithms in which communication among agents facilitates the coordination of planning [4]. However, when the number of features is very large, it is impossible for all agents to communicate all information to all other agents. Furthermore, because of the spatially distributed nature of the exploration task, communication is hindered by large separations between agents. The effects of path loss, which increase with separation distance [40], result in decreasing data rate for a fixed transmission power as distance between agents increases. Thus, it is necessary that algorithms for cooperative autonomous mapping account for the fact that not all information can be transmitted between agents, and that agents must be able to act in situations where only partial information is available.

This thesis examines two distinct facets of the problem of cooperative exploration under communication constraints: the problem of controlling mobile sensing agents such that a balance is achieved between exploration and communication efficiency, and the problem of efficiently utilizing the available communication channel. These problems are related to both the exploration problem and the mapping problem. To advance the state of the art in

each of these areas, this thesis develops communication-aware, computationally efficient algorithms for mobile backbone network optimization and for cooperative estimation under communication constraints.

## 1.1 Thesis Outline

The chapters of this thesis are organized as follows:

Chapter 2 provides background information and context for the problem of cooperative exploration under communication constraints. Technical details of some key ideas and technologies used in Chapters 3-5 are given.

Chapter 3 develops new algorithms for optimization of a particular type of network known as a *mobile backbone network*. Networks using this hierarchical communication framework are composed of regular nodes (sensing agents) and mobile backbone nodes (which serve as relocatable communication hubs). Two objectives are considered: one in which the goal is to maximize the number of regular nodes that achieve a desired minimum throughput level $\tau_{min}$, and one in which the goal is to maximize the total throughput achieved by all regular nodes.

For the case in which the objective is to maximize the number of regular nodes that achieve throughput at least $\tau_{min}$, both an exact MILP-based technique and the first known approximation algorithm with computation time polynomial in the number of regular nodes and the number of mobile backbone nodes are described. Both approaches are validated through simulation and are shown to be computationally tractable for problems of interest. In addition, the approximation algorithm is proved to have a theoretical performance guarantee.

Chapter 3 also develops algorithms for maximizing the total throughput achieved by all regular nodes. Two formulations of this problem are considered: one in which all regular nodes transmitting to the same mobile backbone node adjust their transmission power in order to combat the near-far effect, and one in which all regular nodes transmit at full power, but schedule their transmissions such that each mobile backbone node receives from at most one regular node at a time.

For the case in which regular nodes are able to adjust their transmission power, an exact algorithm with computation time polynomial in the number of regular nodes is developed. This algorithm is compared with a second, MILP-based exact algorithm, as well as with a greedy heuristic algorithm, both of which are also developed in Chapter 3.

When regular nodes transmit at a fixed power level, reduction to an open shop scheduling problem facilitates the formulation of linear programming-based algorithms for optimally allocating transmission times for all regular nodes and mobile backbone nodes, under the assumption that preemption of transmissions carries no penalty. The linear program is then modified to account for time delays caused by preemption in the resulting schedule.

Chapter 4 extends the mobile backbone network optimization techniques developed in Chapter 3 to the case of mobile, controlled sensor platforms. A MILP-based exact algorithm for the problem of placing both regular nodes and mobile backbone nodes, while simultaneously assigning mobile backbone nodes to regular nodes, is shown to perform well in practice for problems of moderate size. A polynomial-time approximation algorithm is also developed, and a performance guarantee is established. The algorithm is also modified to accommodate variable distribution of information in the environment, with the objective of placing regular nodes in the most valuable locations such that they are able to communicate with mobile backbone nodes.

These network optimization techniques are then applied to a cooperative exploration problem using a limited lookahead strategy. The performance of this limited lookahead strategy is examined in the context of a cooperative exploration problem in which sensing locations are given *a priori*, and a performance bound is established for a special case of this exploration problem. Finally, the network optimization algorithms developed in this chapter are integrated into an exploration technique that takes a given area to be explored, divides it into convex subregions, places sensing locations within these subregions such that the entire area is covered while ensuring that all sensing locations are connected.

Chapter 5 focuses on efficient use of available communication bandwidth and provides insight into the way to best utilize limited communication capabilities in information-sensitive tasks, as well as the performance one can hope to achieve with a limited amount of communication capability. In the context of cooperative estimation, the problem of select-

ing information for transmission between two agents is considered. Two novel algorithms are presented: an approximation algorithm that provides a provably good solution to the transmission selection problem in computationally tractable time for a particular class of problem, and a heuristic algorithm that is applicable to all problem instances and that is shown through simulations to have good empirical performance.

For the case of a block-diagonal information matrix, a polynomial-time approximation algorithm that carries a theoretical performance guarantee is developed. This algorithm, which is based on reduction to a multiple-choice knapsack problem, provides provably good performance in computationally tractable time. Since block-diagonal information matrices arise quite naturally in certain sensing models, this algorithm has important practical applicability.

For the case of a fully-populated information matrix, Chapter 5 develops a heuristic algorithm that demonstrates good performance in simulation for problems of realistic complexity. Although it carries no performance guarantee, this algorithm can be applied to problems in which correlations exist between all pairs of feature locations being estimated; thus, this algorithm has very general applicability.

Finally, Chapter 6 summarizes the work done in this thesis and provides future directions for research.

# Chapter 2

# Background

This chapter provides background information and context for the problem of cooperative exploration under communication constraints. Technical details of some key ideas and technologies used in Chapters 3-5 are also given.

## 2.1 Coverage, Exploration and Mapping

Three related issues in mobile robotics are the problems of coverage, exploration and mapping. In coverage and exploration, the objective is to maneuver mobile robots in order to optimally cover all areas in the environment with the robots' sensor footprints [13]. In mapping, the objective is to build a probabilistic representation of the world based on noisy sensor measurements.

### 2.1.1 Coverage and Exploration

Most coverage and exploration algorithms focus on minimizing, either directly or indirectly, the time or energy required to explore an area. Some algorithms guarantee complete coverage, while others (notably, nondeterministic algorithms) do not offer such a guarantee but may entail a decreased computational burden. The earliest work in coverage focused on planning the motion of a single robot so as to completely explore an area. Most of this work was based on the idea of decomposing the environment into a set of cells, each of

which could be easily covered using a back-and-forth motion on the part of the robot [12]. The idea of cellular decomposition quickly became a mainstay in the coverage literature and still appears in algorithms for multiple robots [11].

Ge and Fua [20] focused on limiting repeated coverage of previously-explored areas in multi-robot teams by maintaining a small amount of unexplored area around all explored areas and obstacles. This procedure enables the total time required to explore the area to be bounded, although in the worst case, the upper bound on total time is equivalent to the single-robot upper bound.

Kong, Peng and Rekleitis [11] described an algorithm in which multiple agents dynamically explore an environment in the presence of unknown objects. The robots' cellular decomposition of the environment is updated when a new obstacle is discovered, and information about the accessibility of these cells is communicated via an adjacency graph that is shared among all agents. However, the algorithm in Ref. [11] assumes unlimited communication capabilities.

Batalin and Sukhatme [7] addressed the problem of covering an environment while simultaneously building a communication network. In their algorithm, a single robot carries and dispenses a limited number of "network nodes" in the environment. The nodes, in turn, provide the robot with instructions for traveling in the least recently visited direction, thus eliminating the need for explicit localization on the part of the robot.

Rekleitis, Lee-Shue, New and Choset [43] extended a decomposition-based coverage algorithm for a single robot to the case of multiple robots, with the added restriction that two robots can communicate if they have a line-of-sight connection; however, the impacts of interference and separation distance were not modeled. The work presented in this thesis takes the important step of modeling these two factors.

Also of note are techniques in which it is assumed that not all areas of the environment can be covered, and in which the goal is to choose sensing locations that, for example, minimize the entropy of the resulting estimate of some quantity in the environment. Much work in this area has been done by Guestrin and Krause (e.g., [29] and [30]). A hallmark of the work of Guestrin and Krause is the exploitation of submodularity, which will also play an important role in this thesis.

### 2.1.2 Mapping

In many applications, it is desirable to map the locations of objects or features within an environment. Some early work in mapping was done by Smith, Self and Cheeseman [47]. Leonard [34], Durrant-Whyte and Thrun [57] continue to be major contributors to the field.

Early work in mapping focused on algorithms for a single robot. Thrun [53] presented a machine learning-based algorithm for single-agent exploration, as well as an algorithm combining both grid-based and topological mapping paradigms, aimed at increasing computational efficiency while maintaining accuracy and consistency [54].

Algorithms for multiple robots have generally focused rapid dispersion of agents at the expense of communication capability. For example, Simmons et al. [46] presented a greedy heuristic algorithm for robot motion planning. They noted that their algorithm "tends to keep the robots well separated." This is a desirable characteristic in the case in which communication among agents is unnecessary; however, such separation can cause great degradation in performance of any decentralized or cooperative algorithm in which communication is necessary.

Indeed, while the mapping problem has been studied in great detail, and in fact decentralized algorithms exist for performing mapping [63, 64], none of these techniques adequately address the issue of communication among agents or capture the tradeoff between agents expanding their separation (in order to cover the area of interest more quickly) and staying close to one another (in order to maintain adequate communication links). Some approaches in this direction have involved, for example, constraining the amount of information that can be sent between agents in any one time step. This approach does not allow agents to take advantage of close proximity to send large amounts of data, nor does it realistically model the cost of communicating over very large distances. Other approaches have involved setting a maximum inter-agent distance in order to maintain connectivity, but this approach does not balance the reward of information gained by increasing inter-agent separation with the increased cost of communicating: the inter-agent separation is set *a priori* and does not depend on specific characteristics of the map, or the status of the exploration process. Empirical work has been done on balancing the dispersion of robots

with maintenance of a communication network [35], but a full analysis was not carried out.

## 2.2 Mobile Backbone Network Optimization

The first part of thesis will model the cooperative exploration problem using a particular type of network known as a *mobile backbone network*. In this type of network, *mobile backbone nodes*, which have superior mobility and communication capabilities, provide communication support for *regular nodes*, which are limited in mobility and communication capability but which can sense the environment.

Mobile backbone networks were described by Rubin et al. [44] and Xu et al. [62]. Xu et al. discussed the mobile backbone network architecture as a solution to the scalability issues inherent in ad hoc networks. Noting that most communication bandwidth in large-scale networks is dedicated to packet-forwarding and routing overhead, they proposed a multi-layer hierarchical network architecture, as is already used in the Internet [62]. The authors found empirically that most mobile ad hoc networks could operate efficiently with only two layers of hierarchy, and so they introduced algorithms for dynamic selection of mobile backbone nodes from among candidate nodes in order to address challenges arising from the mobility of these networks, which are not seen in the Internet application. Additionally, they provided a theoretical basis for finding the optimal number of mobile backbone nodes given the overall network size and communication capabilities of the mobile backbone nodes and the regular nodes. Rubin et al. [44] also focused on selecting a subset of candidate mobile backbone nodes and described protocols for coordination of network resources through these mobile backbone nodes.

Srinivas et al. [51] formulated the *connected disk cover* (CDC) problem, in which many mobile backbone nodes with fixed communication ranges are deployed to provide communication support for a set of fixed regular nodes. The goal of the CDC problem is to place a minimum number of mobile backbone nodes such that each regular node is covered by at least one mobile backbone node and all mobile backbone nodes are connected to each other. Thus, the CDC problem takes a discrete approach to modeling communication, in that two nodes can communicate if they are within communication range of each other, and

otherwise cannot.

Srinivas and Modiano [50] used a more sophisticated model of communication, in which the throughput (data rate) achieved between a regular node and a mobile backbone node was modeled as a decreasing function of both the distance between these nodes, and the number of other nodes that are operating at the same frequency. A second key difference between the work of Srinivas and Modiano and previous work lies in their assumptions about the availability of mobile backbone nodes: while previous formulations have assumed that a large or unlimited number of mobile backbone nodes are available, and a minimum number are to be selected such that a specific goal is accomplished, Srinivas and Modiano assumed that a fixed number of mobile backbone nodes are available, and are to be utilized in such a way as to optimize performance. Building upon this throughput model and assumption about mobile backbone node availability, Srinivas and Modiano formulated the *maximum fair placement and assignment* problem (MFPA) and the *maximum throughput placement and assignment* problem (MTPA). In the MFPA formulation, mobile backbone nodes are placed and regular nodes are assigned to communicate with them, such that all regular nodes are assigned and the throughput achieved by the minimum-throughput regular nodes it maximized. In the MTPA formulation, the objective is to maximize the total throughput achieved by all regular nodes. Srinivas and Modiano presented exact algorithms for the MFPA problem and for a restricted version of the MTPA algorithm with at most two mobile backbone nodes, as well as heuristic algorithms for these problems.

## 2.3 Estimation under Communication Constraints

The second part of this thesis examines the problem of selecting information for transmission among agents with the objective of reducing the agents' uncertainty about a set of parameters being estimated; in the case of cooperative exploration, the locations of features in the environment.

### 2.3.1 The Information Filter

The work presented in Chapter 5 leverages the properties of the information filter, which is mathematically equivalent to the traditional Kalman filter but which possesses properties of value for the cooperative exploration problem [36, 22]. The information matrix and information vector are defined as $Y = P^{-1}$ and $y = P^{-1}\hat{x}$, where $P$ is the covariance matrix in the traditional Kalman filter, and $\hat{x}$ is the state vector. Compared to the covariance filter, the information filter essentially exchanges complexity in the measurement update step for complexity in the prediction step.

Denoting the state transition matrix by $F$, the observation matrix by $H$, the process noise covariance matrix by $Q$, and the observation noise covariance matrix by $R$, the prediction step of the information filter can be written as

$$\begin{aligned} Y(k|k-1) &= [F(k)Y^{-1}(k-1|k-1)F^T(k) + Q(k)]^{-1} \\ y(k|k-1) &= Y(k|k-1)F(k)Y^{-1}(k-1|k-1)y(k-1|k-1) \end{aligned}$$

and the measurement update step is:

$$\begin{aligned} Y(k|k) &= Y(k|k-1) + H^T(k)R^{-1}(k)H(k) \\ y(k|k) &= y(k|k-1) + H^T(k)R^{-1}(k)z(k) \end{aligned}$$

The information filter is particularly well suited for cooperative sensor fusion problems in which many measurement updates may take place in a single time step [36, 22]. This additive structure also facilitates the development of the algorithms described in Chapter 5.

### 2.3.2 Covariance Intersection

*Covariance intersection*, introduced by Julier and Uhlmann [23, 59], is a very widely-used technique for creating consistent estimates in the presence of unknown correlations. A consistent estimate is one that does not underestimate the true uncertainty in the system. That is, if the fusion of two estimates of *known* correlation produces covariance matrix $C$, then an estimate with covariance matrix $\tilde{C}$ is consistent if $\tilde{C} - C$ is positive semidefinite.

The covariance intersection technique simply involves taking convex combinations of the information matrices being fused. For the case of two estimates, $y_1$ and $y_2$, with information matrices $Y_1$ and $Y_2$, a consistent estimate $y_3$, $Y_3$ is simply

$$Y_3 = \omega Y_1 + (1-\omega)Y_2$$
$$y_3 = \omega y_1 + (1-\omega)y_2$$

for any $\omega \in [0,1]$.

Julier and Uhlmann proved that this simple, easily implemented technique is guaranteed to produce a consistent estimate, and they noted that the particular value of $\omega$ used can be selected in order to optimize some quantity of interest, such as the trace or the determinant of the resulting covariance matrix. Such optimization is particularly efficient in the case that the objective function is convex in $\omega$, a property that will be leveraged in Chapter 5.

### 2.3.3 The Channel Filter

Work has been done by Nettleton et al. [38] in the area of decentralized estimation under communication constraints. In Ref. [38], the problem of exchanging information among agents was considered in the context of the information filter. In the algorithm presented in Ref. [38], the information to be sent from the sender's information matrix $Y$ is selected as the single submatrix containing the features about which the sender has learned the most since the previous transmission. This is a reasonable approach, but it can be improved upon. Improved selection techniques will be discussed further in Chapter 5. These techniques are incorporated into the overall communication architecture given in in [38], which is described here for convenience.

Once a submap is selected for transmission, it is extracted from the information matrix using two projection matrices, $G_m$ and $G_v$, where $G_m y(k|k)$ contains the features selected for transmission and $G_v y(k|k)$ contains all unselected features. Then, the information vector and matrix to be sent are:

$$
\begin{aligned}
y^*(k|k) &= G_m[y(k|k) - Y(k|k)G_v^T \times (G_v Y(k|k)G_v^T)^{-1} G_v y(k|k)] \\
Y^*(k|k) &= G_m[Y(k|k) - Y(k|k)G_v^T \times (G_v Y(k|k)G_v^T)^{-1} G_v Y(k|k)] G_m^T.
\end{aligned}
$$

This submap is then sent to the *channel filter*, which maintains a record of the common information between two agents. The channel filter is updated as follows:

$$
\begin{aligned}
y_{chan}(k|k) &= \omega y_{chan}(k|k-1) + (1-\omega) G_m^T y^*(k|k) \\
Y_{chan}(k|k) &= \omega Y_{chan}(k|k-1) + (1-\omega) G_m^T Y^*(k|k) G_m
\end{aligned}
$$

where $\omega$ is the covariance intersection weighting parameter. The receiving agent updates its own channel filter in the same way, and updates its estimate as

$$
\begin{aligned}
y(k|k) &= y(k|k-1) + G_s[y_{chan}(k|k) - y_{chan}(k|k-1)] \\
Y(k|k) &= Y(k|k-1) + G_s[Y_{chan}(k|k) - Y_{chan}(k|k-1)] G_s^T
\end{aligned}
$$

where $G_s$ is a projection matrix that "inflates" the map to the appropriate size in the case that the channel filter does not contain information on all states; for instance, vehicle states that are only maintained locally and are not transmitted among agents.

This chapter has provided background information and context for the problem of co-operative exploration under communication constraints. The following three chapters will utilize the ideas and technologies described here in the development of new algorithms.

# Chapter 3

# Static Network Optimization

## 3.1 Introduction

As described in Chapter 2, previous work in mobile backbone network optimization has focused on optimal placement of mobile backbone nodes in networks of fixed regular nodes (referred to as *static network optimization* in this work) [50], or on optimal mobile backbone node trajectory design in networks of uncontrolled regular nodes [49]. This chapter extends previous work in static network optimization and lays the foundation for new *mobile network optimization* techniques, or techniques for optimizing networks of controlled regular nodes and mobile backbone nodes, which will be developed in the next chapter.

Two types of objective function are considered this this chapter: utility is treated as a piecewise constant (step) function of the data rate achievable between a regular node and a mobile backbone node, and also as a monotonically increasing function of this data rate.

In many control and sensing applications, there exists a clear threshold between an acceptable or useful data rate between agents and an unacceptable data rate. For example, a given level of throughput $\tau_{min}$ may be necessary in order to receive adequate feedback in a control system, or transmission of a series of sensor measurements may require a given data rate. This type of situation can be modeled by a step utility function: throughput below $\tau_{min}$ has zero utility, while throughput above $\tau_{min}$ has unit utility. Optimization using a step utility function is described in Section 3.2.

Other applications are more suitably modeled using a monotonically increasing objec-

tive function; that is, an objective function that seeks to maximize the aggregate throughput achieved by all regular nodes. Optimization using an increasing utility function is described in Section 3.3.

## 3.2 Step Utility

This section describes network optimization under a piecewise constant utility function (i.e., a step). In this model, data rate below a given threshold $\tau_{min}$ has zero utility, while throughput above $\tau_{min}$ has unit utility. This utility function can correspond, for example, to the case in which regular nodes are taking sensor measurements, the data from each measurement has a given size, and the goal is to transmit packets of measurement data from as many regular nodes as possible within a given timeframe. Thus, the objective of this section is to maximize the number of regular nodes that achieve throughput at least $\tau_{min}$.

### 3.2.1 Problem Statement

Although many factors can affect the throughput that a regular node can achieve to its assigned mobile backbone node, there are two major factors that are common to nearly all communication architectures: the *distance* between the regular node and the mobile backbone node, and the *number* of other regular nodes that are also communicating with that particular mobile backbone node and thus causing interference. This work models the throughput that can be achieved between a regular node and its assigned mobile backbone node as an arbitrary decreasing function of both of these quantities. For example, as described in Ref. [50], the throughput $\tau$ between regular node $i$ and mobile backbone node $j$ when using a Slotted Aloha communication protocol can be approximated by

$$\tau(i,j) \approx \frac{1}{e \cdot |A(j)| \cdot d(i,j)^{\alpha}}, \tag{3.1}$$

where $e$ is the base of the natural logarithm, $|A(j)|$ is the number of regular nodes assigned to mobile backbone node $j$, $d(i,j)$ is the distance between regular node $i$ and mobile back-

bone node $j$, and $\alpha$ is the path loss exponent.

This section assumes that each regular node is assigned to a single mobile backbone node. An implicit assumption is made that regular nodes assigned to one mobile backbone node encounter no interference from regular nodes assigned to other mobile backbone nodes (for example, because each "cluster" composed of a mobile backbone node and its assigned regular nodes operates at a different frequency than other clusters). This section also assumes that the mobile backbone nodes can communicate effectively with each other over the entire problem domain, so that there is no additional constraint that the mobile backbone nodes need to be "connected" to one another.

Building upon this throughput model, we pose the mobile backbone network optimization problem as follows: our goal is to *place* a number of mobile backbone nodes, which can occupy arbitrary locations in the plane, while simultaneously *assigning* the regular nodes to the mobile backbone nodes, such that the number of regular nodes that achieve a given minimum throughput level $\tau_{min}$ is maximized.

Denoting the problem data as $\mathbf{L_r}$ (the locations of the regular nodes) and the decision variables as $\mathbf{L_m}$ (the selected locations of the mobile backbone nodes) and $\mathbf{A}$ (the assignment of regular nodes to mobile backbone nodes), this optimization problem can be more formally stated as:

$$\max_{\mathbf{L_m},\mathbf{A}} \; F_\tau(\mathbf{L_r}, \mathbf{L_m}, \mathbf{A}, \tau_{min}) \tag{3.2a}$$

$$\text{subject to } \mathbf{A} \in \mathscr{A} \tag{3.2b}$$

where $F_\tau(\mathbf{L_r}, \mathbf{L_m}, \mathbf{A}, \tau_{min})$ is the number of regular nodes that achieve throughput level $\tau_{min}$, given node placements $\mathbf{L_r}$ and $\mathbf{L_m}$, assignment $\mathbf{A}$, and throughput function $\tau$; and $\mathscr{A}$ is the set of feasible assignments, i.e., the assignments in which each regular node is assigned to at most one mobile backbone node.

A typical example of a solution to this problem is shown in Figure 3.2.1 for a group of regular nodes denoted by $\circ$. The mobile backbone nodes, denoted by $\star$, have been placed such that they maximize the number of regular nodes that achieve the given minimum throughput level. This example is typical in that the clusters of regular nodes and

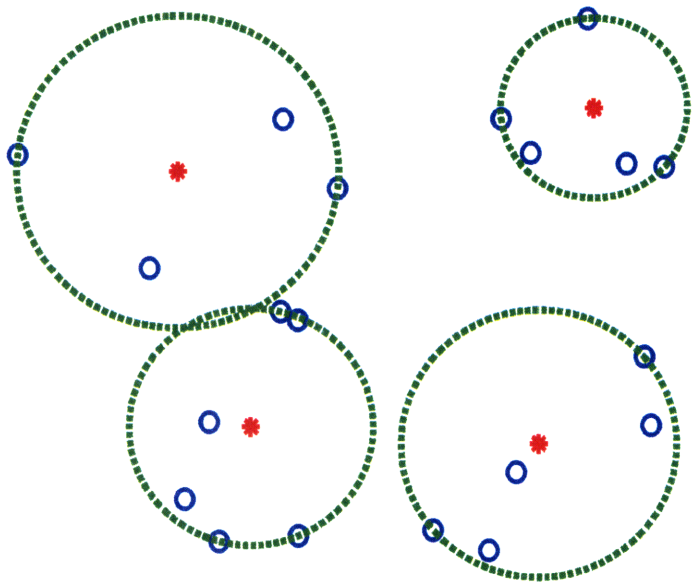


Figure 3-1: A typical example of an optimal mobile backbone network. Mobile backbone nodes, indicated by $*$, are placed such that they provide communication support for regular nodes, shown as $\circ$. Each regular node is assigned to one mobile backbone node. Dashed lines indicate the radius of each cluster of nodes.

mobile backbone nodes are relatively small, and the regular nodes are distributed intelligently among the mobile backbone nodes, with fewer regular nodes being allocated to the mobile backbone nodes with larger cluster radii. In this case, all regular nodes have been successfully assigned to mobile backbone nodes.

We note that Problem 3.2 is a sub-problem of the *maximum fair placement and assignment* (MFPA) problem considered in Ref. [50], in which the objective is to maximize the minimum throughput achieved by any regular node, such that all regular nodes are assigned. An algorithm for solving the MFPA problem takes advantage of the fact that there are only $O(N^4)$ possible values for the minimum throughput level achieved by any regular

node by performing a search over these possible values of $\tau_{min}$ and solving Problem 3.2 for each value of $\tau_{min}$. Thus, an improved algorithm for solving Problem 3.2 also yields an improved algorithm for the MFPA problem.

### 3.2.2 Solution Strategy

Problem 3.2 is, unfortunately, a nonconvex optimization problem and thus is not amenable to naive solution techniques. However, Problem 3.2 has special structure that can be leveraged.

Although the mobile backbone nodes are able to occupy arbitrary locations, they can be restricted to a relatively small number of locations without sacrificing optimality. Specifically, each mobile backbone node can be placed at the 1-center of its set of assigned regular nodes in an optimal solution. The 1-center location of a set of regular nodes is the location that minimizes the maximum distance from the mobile backbone node to any regular node, and it is easily computed [1]. Clearly, if a mobile backbone node that is not located at the 1-center of its assigned regular nodes is receiving data from each regular node at throughput at least $\tau_{min}$, then the mobile backbone node can be moved to the 1-center of these regular nodes, and all regular nodes will still achieve throughput at least $\tau_{min}$.

Fortunately, although there are $2^N$ possible subsets of $N$ regular nodes, there are only $O(N^3)$ distinct 1-center locations. This is due to the fact that each 1-center location either coincides with a regular node location, is located at the middle of the diameter of two regular nodes, or is located at the circumcenter of three regular nodes [39]. Thus, there are $\binom{N}{1} + \binom{N}{2} + \binom{N}{3}$ distinct 1-center locations.

Leveraging the fact that there are a limited number of possible mobile backbone node locations (polynomially many in $N$), Ref. [50] solves the MFPA problem by performing an exhaustive search over all possible placements of $K$ mobile backbone nodes for each possible value of the minimum throughput, determining the optimal assignment for each placement by solving an integer network flow problem. The computation time of this search-based algorithm is thus polynomial in the number of regular nodes, but exponential in the number of mobile backbone nodes.

### 3.2.3 MILP approach

A primary contribution of this work is the development of a single optimization problem that simultaneously solves the mobile backbone node placement and regular node assignment problems, thus eliminating the need for an exhaustive search over possible mobile backbone node placements. This is accomplished through the formulation of a *network design* problem. In network design problems, a given network (represented by a directed graph) can be augmented with additional arcs for a given cost, and the goal is to achieve some desired flow characteristics at a minimum cost by intelligently "purchasing" a subset of these arcs [2].

The network design problem that produces an optimal placement of mobile backbone nodes and assignment of regular nodes is constructed as follows. A source node, $s$, is connected to each node $i$ in the set of nodes N= $\{1,\ldots,N\}$ (see Figure 3.2.3). These nodes represent regular node locations. The arcs connecting $s$ to $i \in$ N are of unit capacity. Each node $i \in$ N is in turn connected to a subset of the nodes in M= $\{N+1,\ldots,N+M\}$, where $M$ is $O(N^3)$. Set M represents the set of candidate locations for mobile backbone nodes, and there are $O(N^3)$ such locations because mobile backbone nodes can be placed at 1-center locations of subsets of regular nodes without sacrificing optimality. Node $i \in$ N is connected to node $N+j \in$ M iff regular node $i$ is contained in one of the sets defining 1-center location $j$. The arc connecting $i$ to $N+j$ is of unit capacity. Finally, each node in M is connected to the sink, $t$. The capacity of the arc connecting node $N+i \in$ M to the sink is the product of a binary variable $y_i$, which represents the decision of whether to place a mobile backbone node at location $i$, and a constant $c_i$, which is the floor of the inverse with respect to cluster size of the throughput function, evaluated at the desired minimum throughput level. In other words, $c_i$ is the maximum number of regular nodes that can be assigned to a mobile backbone node at location $i$ at the desired throughput level $\tau_{min}$. For example, for the approximate Slotted Aloha throughput function described by Eq. (3.1),

$$c_i = \left\lfloor \frac{1}{e \cdot \tau \cdot r_i^{\alpha}} \right\rfloor, \tag{3.3}$$

where $\tau$ is the desired minimum throughput and $r_i$ is the radius associated with 1-center

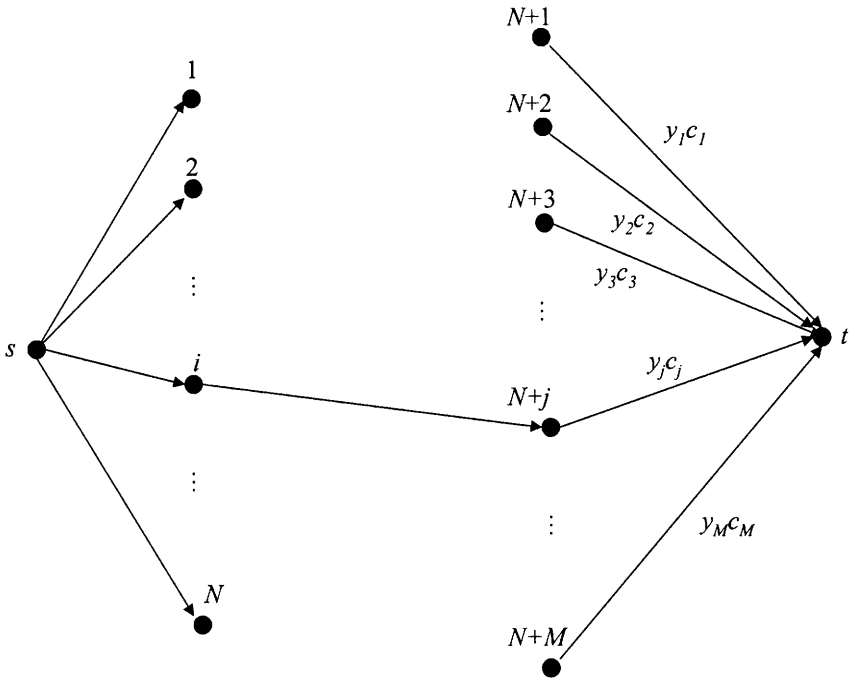



Figure 3-2: The network design problem corresponding to the joint placement and assignment problem for mobile backbone networks. Unlabeled arc capacities are equal to one.

location $i$. This means that if at most $c_i$ regular nodes are assigned to the mobile backbone node at location $i$, each of these regular nodes will achieve throughput at least $\tau$. Denote the set of nodes for this network design problem by $\mathcal{N}$ and the set of arcs by $\mathcal{A}$.

If $K$ mobile backbone nodes are available to provide communication support for $N$ regular nodes at given locations and a throughput level is specified, the goal of the network design problem is to select $K$ arcs incident to the sink and a feasible flow $\mathbf{x}$ such that the net flow through the graph is maximized. This network design problem can be solved via the following mixed-integer linear program (MILP):

$$\max_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{N} x_{si} \tag{3.4a}$$

$$\text{subject to } \sum_{i=1}^{M} y_i \leq K \tag{3.4b}$$

$$\sum_{j:(i,j)\in\mathscr{A}} x_{ij} = \sum_{l:(l,i)\in\mathscr{A}} x_{li} \qquad i \in \mathscr{N} \setminus \{s,t\} \tag{3.4c}$$

$$x_{ij} \geq 0 \qquad \forall\, (i,j) \in \mathscr{A} \tag{3.4d}$$

$$x_{ij} \leq 1 \qquad \forall\, (i,j) \in \mathscr{A} : j \in \mathscr{N} \setminus \{t\} \tag{3.4e}$$

$$x_{(N+i)t} \leq y_i c_i \qquad \forall\ i \in \{1,\ldots,M\} \tag{3.4f}$$

$$y_i \in \{0,1\} \qquad \forall\ i \in \{1,\ldots,M\}. \tag{3.4g}$$

The objective of this problem is to maximize the flow $\mathbf{x}$ through the graph (Eq. (3.4a)). The constraints state that at most $K$ arcs (mobile backbone node locations) can be selected (3.4b), flow through all internal nodes must be conserved (3.4c), arc capacities must be observed (3.4d - 3.4f), and $y_i$ is binary for all $i$ (3.4g). Note that, for a given specification of the $\mathbf{y}$ vector, all flows $\mathbf{x}$ are integer in all basic feasible solutions of the resulting (linear) maximum flow problem.

A solution to problem (3.4) provides both a placement of mobile backbone nodes and an assignment of regular nodes to mobile backbone nodes. Mobile backbone nodes are placed at locations for which $y_i = 1$, and regular node $i$ is assigned to the mobile backbone node at location $j$ if and only if $x_{i(N+j)} = 1$. The number of regular nodes assigned is equal to the volume of flow through the graph.

We make the following observations about this algorithm:

**Remark 3.2.1** *If $K$ mobile backbone nodes are available and the goal is to assign as many regular nodes as possible such that a desired minimum throughput is achieved for each assigned regular node, the above MILP problem needs only to be solved once for the desired throughput value and with a fixed value of $K$. To solve the MFPA problem, which is*

*the primary problem of interest in Ref. [50], it is necessary to solve the above MILP problem multiple times for different throughput values in order to find the maximum throughput value such that all regular nodes can be assigned. There are at most $O(N^4)$ possible minimum throughput values; searching among these values using a binary search would require $O(\log(N))$ solutions of the MILP problem, with $O(N^3)$ binary variables in each problem.*

**Remark 3.2.2** *It should be noted that the worst-case complexity of mixed-integer linear programming is exponential in the number of binary variables. However, this approach performs well in practice, and simulation results indicate that it compares very favorably with the approach developed in Ref. [50] for cases of interest (See Table 3.1). Note that while the computation time of the search-based algorithm increases very rapidly with the problem size, the MILP-based algorithm remains computationally tractable for problems of practical scale.*

**Remark 3.2.3** *If arbitrarily many mobile backbone nodes are available and the goal is simply to achieve a desired minimum throughput while utilizing a minimal number of mobile backbone nodes, then a MILP problem similar to the one above needs only to be solved once, for the values of $c_i$ corresponding to the desired throughput. The problem should be modified such that it is a minimization problem, with $\sum_{i=1}^{M} y_i$ as the objective and $\sum_{i=1}^{N} x_{si} = N$ as a constraint.*

### 3.2.4 Hardness of Network Design

Although the MILP-based algorithm performs well in practice, it is not a computationally tractable approach from a theoretical perspective. This fact motivates consideration of the fundamental tractability of the network design problem itself; if the network design problem is NP-hard, it may be difficult or impossible to find an exact algorithm that is significantly more efficient than the MILP algorithm. This section proves the hardness of network design on a network of the form shown in Figure 3.2.3, which in this thesis is called a two-layered graph. This type of graph is more formally defined as follows:

The node set of a two-layered graph is the union of node sets $\{s\}$, N, M and $\{t\}$.

Table 3.1: Average computation times for the MILP-based and search-based algorithms, for various numbers of regular ($N$) and mobile backbone nodes ($K$) in the maximum fair placement and assignment (MFPA) problem, using ILOG CPLEX 9.020.

| $N$ | $K$ | MILP Algorithm | Search-based Approach |
|---|---|---|---|
| 3 | 2 | 3 sec | 20 sec |
| 4 | 2 | 4 sec | 81 sec |
| 5 | 2 | 5 sec | 202 sec |
| 6 | 2 | 6 sec | 507 sec |
| 6 | 3 | 6 sec | $>$ 30 min |
| 8 | 3 | 8.5 sec | $>$ 30 min |
| 10 | 3 | 9 sec | $>$ 30 min |
| 15 | 5 | 18 sec | $>$ 30 min |
| 20 | 5 | 47 sec | $>$ 30 min |
| 25 | 5 | 196 sec | $>$ 30 min |

Arcs originating at $s$ must terminate at nodes in N; arcs originating in N must terminate at nodes in M; and arcs originating in M must terminate at $\{t\}$. All arcs have unit capacity except those terminating at $\{t\}$, which have integer capacity. All arcs are fixed except those terminating at $\{t\}$, which are the arcs to be selected in the network design problem.

The hardness proof reduces an instance of the $K$-vertex cover problem to an instance of the network design problem on a two-layered graph. The $K$-vertex cover problem is defined as follows:

INPUT: Graph $G = (V,E)$; $K$.

QUESTION: Is there a vertex cover $S$ for $G$ of size at most $K$?

where a vertex cover $S$ is a set of vertices of $G$ such that each edge in $G$ is covered by at least one vertex in $S$.

**Theorem 3.2.4** *Network design on two-layered graphs is NP-hard.*

**Proof 3.2.1** *An instance of the K-vertex cover problem can be reduced to a network design problem on a two-layered graph as follows. For each edge in G there is a node in N; likewise, for each vertex in G, there is a node in M. Node s is connected to each node in N. An arc is present between node $n \in N$ and node $m \in M$ if and only if edge m is incident to vertex n in G. Finally, each node in M is connected to t, and the capacity of the arc connecting $m \in M$ to t is at least the degree of vertex m in G. An example of this*

*transformation is shown in Figure 3-3.*

*Then, given an instance of the K-vertex cover problem, a solution can be obtained by solving the network design problem on the corresponding two-layered graph with cardinality constraint K. If the maximum flow achieved in an optimal solution to the network design problem is equal to $|E|$, then the answer to the K-vertex cover problem is YES. If the maximum flow is less than $|E|$, the answer is NO.*

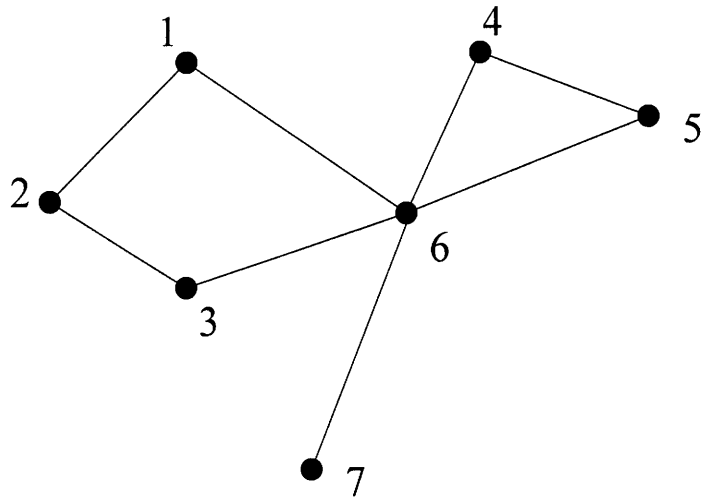
*The time required for this reduction is $O(|E|+|V|)$; therefore, the network design problem is NP-hard.*

It should be noted that this type of graph not only models all mobile backbone network optimization problems of the type described Section 3.2, but also many problems that do not arise in mobile backbone network optimization. Thus, hardness of network design on a two-layered graph does not imply hardness of mobile backbone network optimization. Nevertheless, hardness of the network design problem is evidence for hardness of mobile backbone network optimization, in the sense that any efficient exact algorithm for solving the mobile backbone network optimization problem must exploit some geometric property not captured in the network design problem, rather than exploiting the structure of the network design problem itself (unless P=NP).

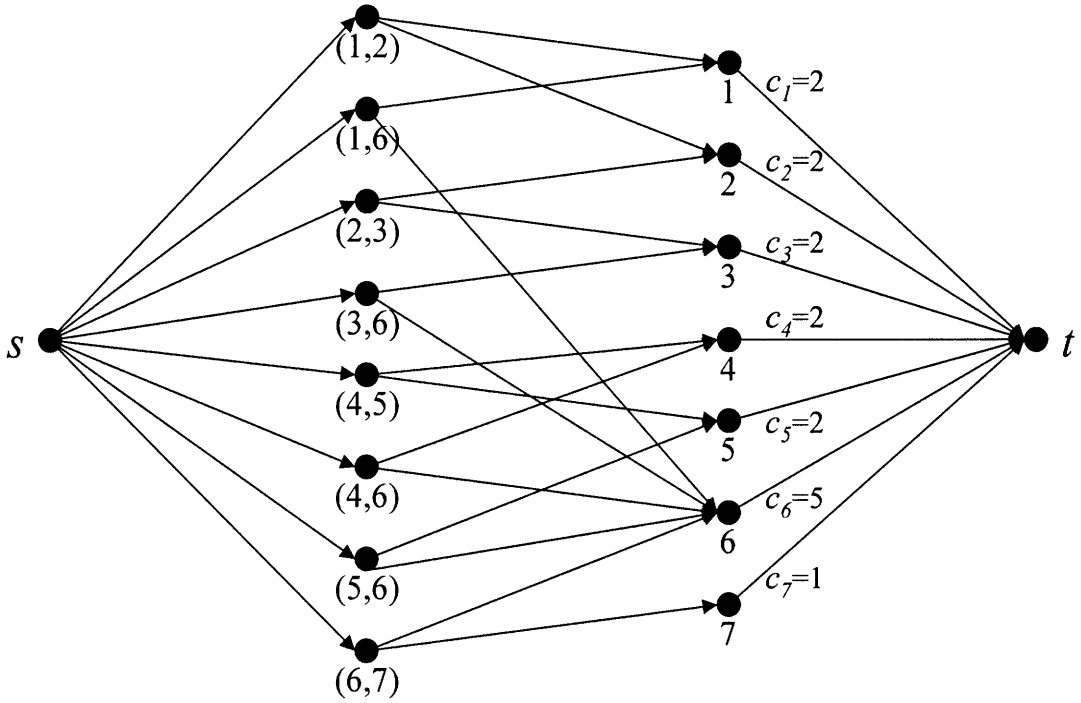
### 3.2.5 Approximation Algorithm

The MILP formulation of Section 3.2.3 provides an optimal solution to the mobile backbone network optimization problem with a step utility function in computationally tractable time for moderately-sized problems. For large-scale problems, an approximation algorithm with computation time that is polynomial in the number of regular nodes and the number of mobile backbone nodes is desirable. This section describes such an algorithm.

The primary insight that motivates the approximation algorithm is the fact that the maximum number of regular nodes that can be assigned is a *submodular* function of the set of mobile backbone node locations selected. Given a finite ground set $D = \{1,\ldots,d\}$, a set function $f(S)$ defined for all subsets $S$ of $D$ is said to be submodular if it has the property




(a) A graph given as input in the $K$-vertex cover problem.




(b) A two-layered graph in the network design problem to which the $K$-vertex cover problem is reduced.

Figure 3-3: An example of conversion from a $K$-vertex cover problem to an network design problem.

that

$$f(S \cup \{i,j\}) - f(S \cup \{i\}) \leq f(S \cup \{j\}) - f(S)$$

for all $i, j \in D$, $i \neq j$ and $S \subset D \setminus \{i,j\}$ [8]. In the context of the network design problem, this means that the maximum flow through the network is a submodular function of the set of arcs incident to the sink that are selected.

Submodular functions in discrete optimization are analogous to convex functions in continuous optimization [19]. Both can be efficiently minimized; however, maximization is more difficult. Fortunately, it has been shown that for maximization of a nondecreasing submodular set function $f$, where $f(\emptyset) = 0$, greedy selection of elements yields a performance guarantee of $1 - (1 - \frac{1}{P})^P > 1 - \frac{1}{e}$, where $P$ is the number of elements to be selected from the ground set and $e$ is the base of the natural logarithm [37]. This means that if an exact algorithm selects $P$ elements from the ground set and produces a solution of objective value $OPT$, greedy selection of $P$ elements (i.e., selection via a process in which element $i$ is selected if it is the element that maximizes $f(S \cup \{i\})$, where $S$ is the set of elements already selected) produces a solution of value at least $(1 - (1 - \frac{1}{P})^P) \cdot OPT$[37]. For the network design problem considered in this section, $P = K$ (the number of mobile backbone nodes that are to be placed), and $OPT$ is the number of regular nodes that are assigned in an optimal solution. Note that greedy selection of $K$ arcs amounts to solving at most $O(N^3K)$ maximum flow problems on graphs with at most $N + K + 2$ nodes. Thus, the computation time of the greedy algorithm is polynomial in the number of regular nodes and the number of mobile backbone nodes.

We now prove that the objective function in the problem under consideration is submodular.

**Theorem 3.2.5** *If G is a graph in the network design problem described in Section 3.2.3, the maximum flow that can be routed through G is a submodular function of the set of arcs selected.*

**Proof 3.2.2** *We begin by restating the submodularity condition as follows:*

$$f^*(S) + f^*(S \cup \{i,j\}) \leq f^*(S \cup \{i\}) + f^*(S \cup \{j\}) \tag{3.5}$$

*where $f^*$ is the maximum flow through G, as a function of the set of selected arcs. Next, we note that for a fixed selection of arcs S, the problem of finding the maximum flow through G can be expressed as an equivalent matching problem on a bipartite graph with node sets L and R*[1]*. This is accomplished as follows: node set L in the bipartite matching problem is simply node set N in the maximum flow problem. Node set R is derived from node set M in the maximum flow problem, with one modification: if the arc from node $N+i \in M$ to t has outgoing capacity $c_i$, then R contains $c_i$ copies of node $N+i$, each of which is connected to the same nodes in L as the original node $N+i$. Thus, each node $N+i$ in the maximum flow problem becomes a* set *of nodes $N+i$ in the bipartite matching problem, and the cardinality of this set is equal to $c_i$. An example of this reformulation is shown in Figure 3-4.*

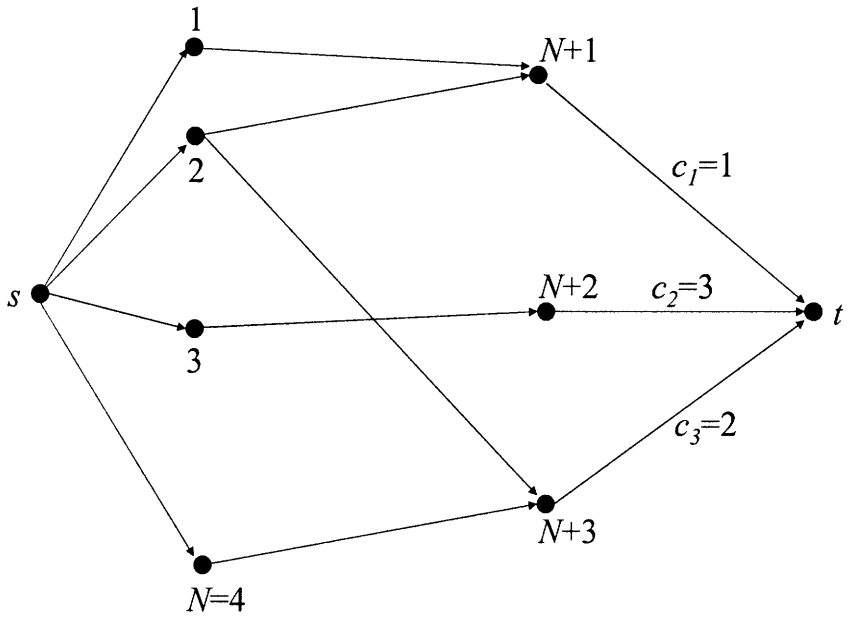
*For any feasible flow in the original graph, there is a corresponding matching in the bipartite graph with cardinality equal to the volume of flow; likewise, for any feasible matching in the bipartite graph, there is a corresponding flow of volume equal to the cardinality of the matching. Therefore, the maximum flow through the original graph is equal to the cardinality of a maximum matching in the bipartite graph.*

*The graphs expressing the relation in Eq. (3.5) are shown in the top row of Figure 3-5: the sum of the maximum flows through the left two graphs must be less than or equal to the sum of the maximum flows through the right two graphs.*

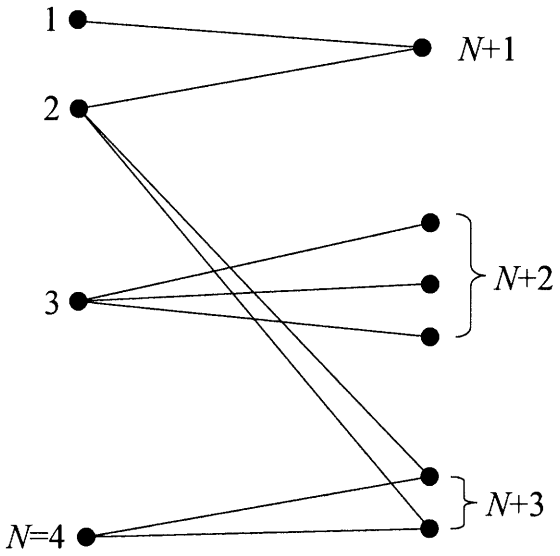
*Converting these maximum flow problems into their equivalent bipartite matching problems, we obtain the condition that the sum of the cardinalities of maximum matchings in bipartite graphs $G_1$ and $G_2$ in Figure 3-5 is at most the sum of the cardinalities of maximum matchings in $G_3$ and $G_4$.*

*Consider a maximum matching $M_1$ in graph $G_1$, and denote its cardinality by $N_s$. This means that $N_s$ nodes from set S are covered by matching $M_1$. Note that $M_1$ is a feasible matching for $G_2$ as well, since all arcs in $G_1$ are also present in $G_2$.*

[1] A set of edges in a graph is a *matching* if no two edges share a common end node. A *maximum matching* is a matching of maximum cardinality [5].




(a) A graph over which a maximum flow problem can be formulated. Unlabeled arc capacities are equal to one.




(b) A bipartite matching problem that is equivalent to the maximum flow problem above.

Figure 3-4: An example of conversion from a maximum flow problem to an equivalent bipartite matching problem, for $N = 4$, $M = 3$.

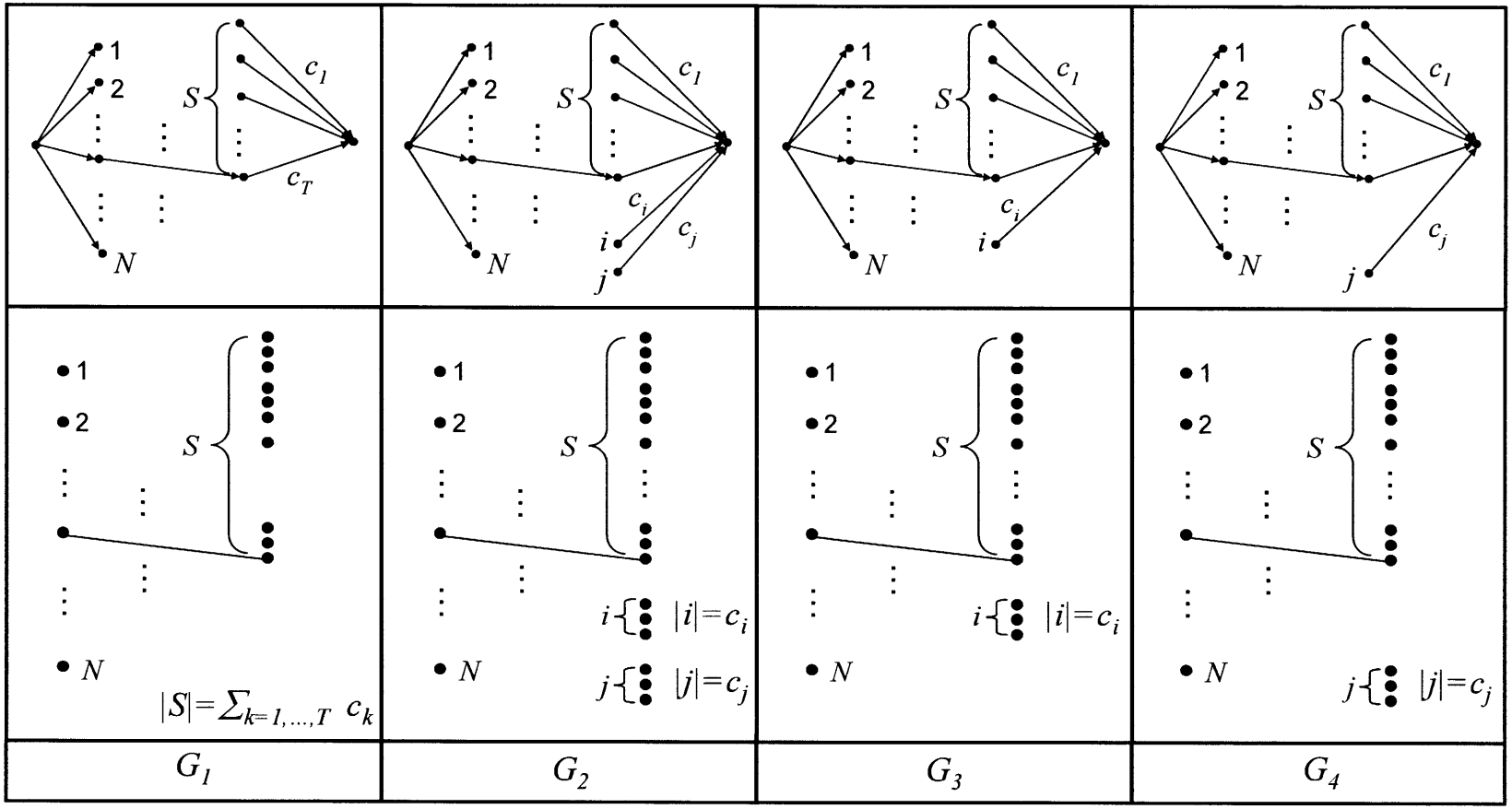



Figure 3-5: Schematic representation of the graphs involved in the proof of the submodularity condition. The top graphs relate to the original maximum flow problem, while the bottom graphs are their equivalent reformulations in the bipartite matching problem. For clarity, not all arcs are shown.

*It is a property of bipartite graphs that if a matching $Q$ is feasible for a graph $H$, then there exists a maximum matching $Q^*$ in $H$ such that all of the nodes covered by $Q$ are also covered by $Q^*$ [5]. Denote such a maximum matching for matching $M_1$ in graph $G_2$ by $M_2$, and note that $N_s$ nodes from set $S$ are covered by $M_2$. Denote the number of nodes covered by $M_2$ in node sets $i$ and $j$ by $N_i$ and $N_j$, respectively. Then, the total cardinality of these maximum matchings for graphs $G_1$ and $G_2$ is equal to $2N_s + N_i + N_j$.*

*Now consider the matching obtained by removing the edges incident to node set $j$ from $M_2$. Note that this matching is feasible for graph $G_3$, and its cardinality is $N_s + N_i$. Likewise, the matching obtained by removing the edges incident to node set $i$ from $M_2$ is feasible for graph $G_4$, and its cardinality is $N_s + N_j$. Since these matchings are feasible (but not necessarily optimal) for $G_3$ and $G_4$, the sum of the cardinalities of maximum matchings for these graphs must be at least $2N_s + N_i + N_j$. This establishes the submodularity property for the matching problem as well as for the maximum flow problem.*

Then, given a network design graph $G$, $K$ mobile backbone nodes and $M$ possible mobile backbone node locations, and denoting by $f$ the maximum flow through $G$ as a function of the set of mobile backbone node locations selected, an approximation algorithm with performance guarantee $1 - \frac{1}{e}$ is:

**Algorithm 1** Greedy

```
S ← ∅
maxflow ← 0
for k=1 to K do
  for m=1 to M do
    if f(S∪{m}) ≥ maxflow then
      maxflow ← f(S∪{m})
      m* ← m
    end if
  end for
  S ← S∪{m*}
end for
return S
```

### 3.2.6 Computational Efficiency of the Greedy Approximation Algorithm

Greedy Algorithm 1 is computationally efficient, solving at most $O(KN^3)$ maximum flow problems with at most $N+K+2$ nodes each. Furthermore, additional computation efficiency can be achieved by exploiting the structure of the maximum flow problem solved at each iteration of the greedy algorithm. Note that in the first round of greedy selection, each maximum flow problem is of the form shown in Figure 3-6: a single node is connected to the sink, and it is in turn connected to a subset of the nodes representing regular nodes. The capacity of the minimum cut in this graph is equal to $\min\{c_j, \delta^+(j)\}$, where $\delta^+(j)$ is the incoming degree of node $j$. Then, the first round of greedy selection can be accomplished by searching over $O(N^3)$ values of $\min\{c_j, \delta^+(j)\}$, rather than formulating and solving $O(N^3)$ maximum flow problems. Furthermore, if $c_j = \delta^+(j)$, then all regular nodes covered by the mobile backbone node at location $j$ can be assigned to that mobile backbone node in any optimal assignment. Thus, if a mobile backbone node location $j$ can be found in the first round of greedy selection such that $c_j = \delta^+(j) = \max_i \min\{c_i, \delta^+(i)\}$, this location can be selected and in the second round of greedy selection mobile backbone node location $j$ and all regular nodes incident to $j$ can be removed from the graph, thereby reducing the graph over which the second round of greedy selection takes place to one of the form shown in Figure 3-6.

Thus, a more efficient implementation of the greedy selection algorithm would search over all locations $1,\ldots,M$ for a location $j$ such that

$$c_j = \delta^+(j) = \max_{k \in 1,\ldots,M} \min\{c_k, \delta^+(k)\}$$

Unfortunately, it is not always possible to select such a location. For example, consider a configuration of regular nodes as shown in Figure 3-7(a), and assume that the throughput function is such that up to five regular nodes can be assigned to a mobile backbone node with any 1-center radius. In this case, a mobile backbone node can be placed such that it covers exactly five regular nodes; see Figure 3-7(b). However, if the throughput function is such that *six* regular nodes can be assigned, it is no longer possible to place a mobile

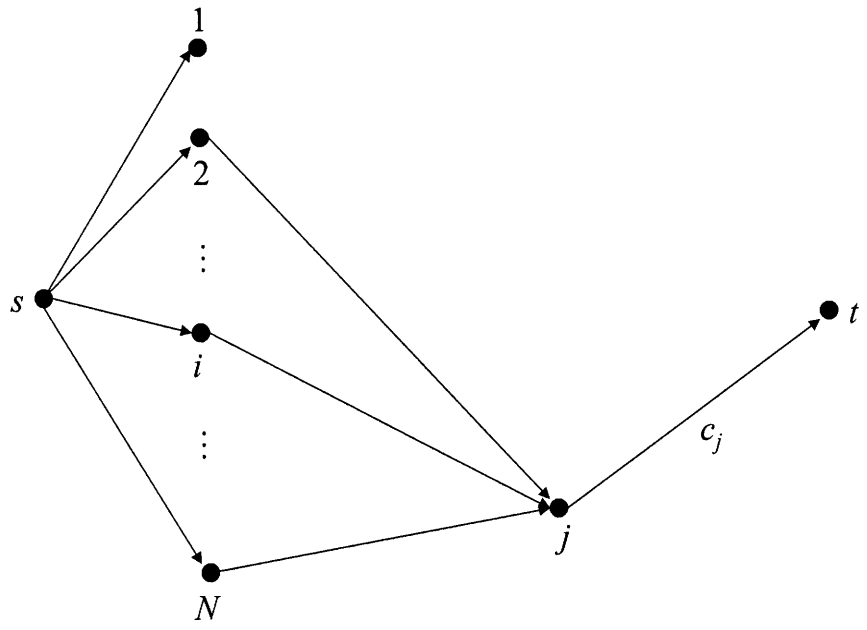
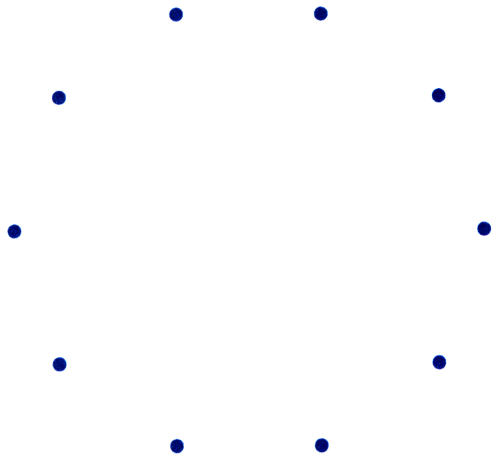


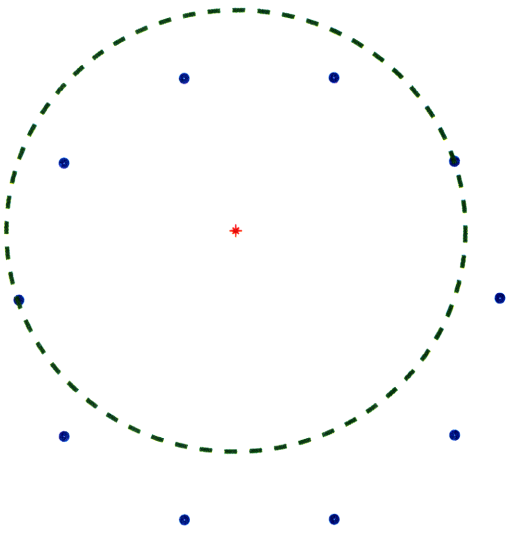

Figure 3-6: Schematic representation of the graphs over which the maximum flow problem is solved in the first round of greedy mobile backbone node location selection.

backbone node at a 1-center location such that exactly six regular nodes are covered.

However, empirical results indicate that it is nearly always possible to select such a location at each round of greedy selection for a random configuration of regular nodes. This yields a best-case computation time of $O(KN^3)$. If it is impossible to select such a location, all subsequent rounds of greedy selection will involve solution of a maximum flow problem. However, each of the maximum flow problems solved by the greedy algorithm is solved over a bipartite graph with node sets N$\cup\{t\}$ and $\{s\}\cup$K, where K is the set of nodes from M whose outgoing arcs are selected. Because maximum flow problems can be solved even more efficiently in bipartite networks than in general networks [2], the greedy algorithm is thus highly efficient even in the case in which it is impossible to take advantage of max-flow/min-cut duality. In the worst case, solution of $O(KN^3)$ maximum flow problems on bipartite graphs has overall complexity $O(K^3N^4)$ [3].



(a) A configuration of regular nodes for which exactly five regular nodes can be covered by a mobile backbone node placed at a 1-center location, but exactly six cannot.



(b) A mobile backbone node placement for which exactly five regular nodes are covered.

Figure 3-7: In this configuration of regular nodes, it is possible to cover exactly five regular nodes with a mobile backbone node placed at a 1-center location, but it is not possible to cover exactly six.

### 3.2.7 Experimental Evaluation of the Greedy Approximation Algorithm

As described in the Section 3.2.5, greedy selection of mobile backbone node locations results in assignment of at least $\lceil (1-(1-\frac{1}{K})^K) \cdot OPT \rceil \geq \lceil (1-\frac{1}{e}) \cdot OPT \rceil$ regular nodes, where $K$ is the number of mobile backbone nodes that are to be placed and $OPT$ is the number of regular nodes assigned by an exact algorithm (such as the MILP algorithm described in Section 3.2.3) [37]. However, this observation is based on a general result for nondecreasing submodular functions and not for the specific problem under consideration in this section. Therefore, it is of interest to experimentally examine the performance of the greedy algorithm for our problem of interest.

To this end, computational experiments were performed on a number of problems of various sizes. Regular node locations were generated randomly in a finite 2-dimensional area, and a moderate throughput value was specified (i.e., one high enough that there was no trivial selection of mobile backbone node locations that would result in assignment of all regular nodes). Results were averaged over a number of trials for each problem dimension.

Figure 3-8 shows the performance of the approximation algorithm relative to the exact (MILP) algorithm. In Figure 3-8(a), the average percentage of regular nodes assigned by the exact algorithm that are also assigned by the approximation algorithm is plotted, along with the theoretical lower bound of $\lceil (1-\frac{1}{e}) \cdot OPT \rceil$, for various problem sizes. In this figure, a data point at 100% would mean that, on average, the approximation algorithm assigned as many regular nodes as the exact algorithm for that particular problem size. As the graph shows, the approximation algorithm consistently exceeds the theoretical performance guarantee and achieves nearly the same level of performance as the exact algorithm for all problem sizes considered.

Figure 3-8(b) shows the computation time required for each of these algorithms, plotted on a logarithmic axis. As the figure shows, the computation time required for the approximation algorithm scales gracefully with problem size. The average computation time of the approximation algorithm was about 10 seconds for $N = 40$ and $K = 8$, whereas the MILP algorithm took nearly three hours to solve a problem of this size, on average. Both the

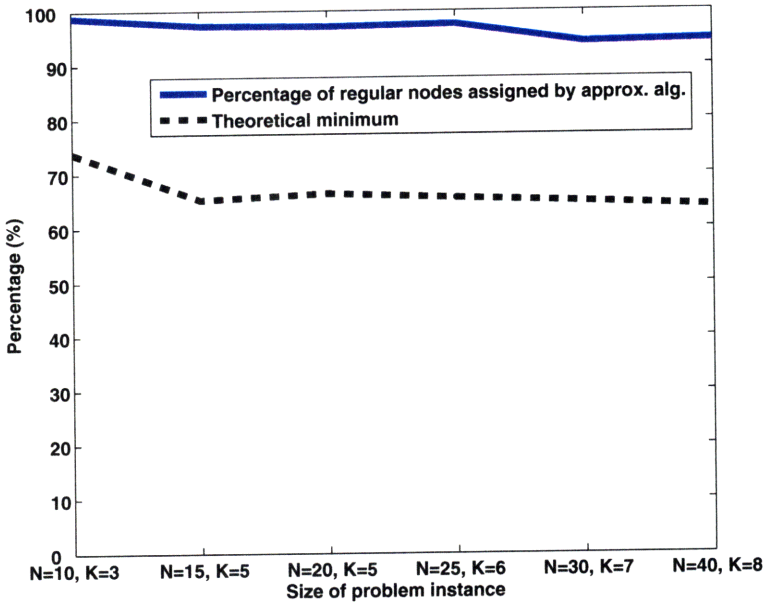
MILP algorithm and the approximation algorithm were implemented using ILOG CPLEX 9.020.

## 3.3 Increasing Utility with Data Rate

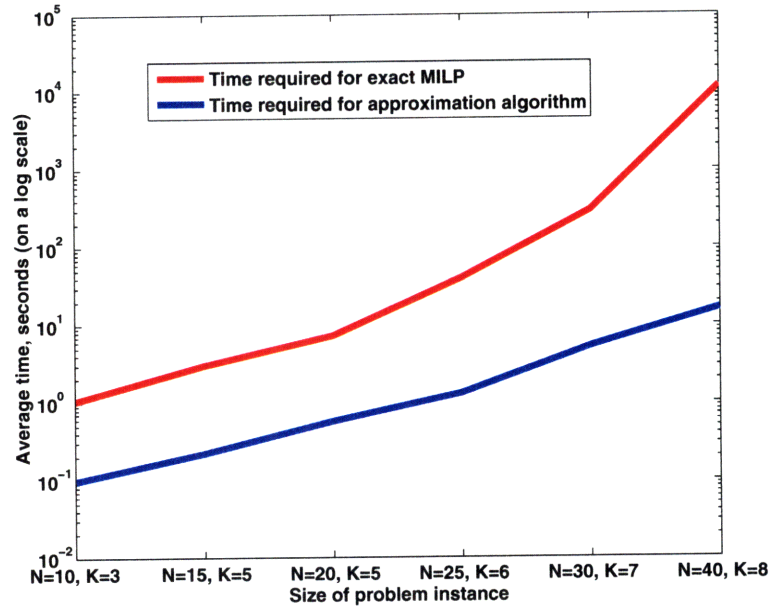
Thus far, this chapter has assumed that the utility of a communication configuration as a function of achieved data rate is a step function; that is, if a regular node achieves throughput less than $\tau_{min}$, it adds nothing to the objective function, and if a regular node achieves throughput at least $\tau_{min}$ it increases the overall objective by one. This utility model is appropriate in many cases; for instance, the case in which a prescribed amount of data must be transmitted in order to accomplish a specific task. However, other utility functions are appropriate for other tasks. This section considers an objective function for each mobile backbone node that is an increasing function of the data rates between it and its assigned regular nodes. This type of objective function is appropriate, for example, when the quality of the estimate produced through a cooperative estimation task is an increasing function of the amount of data that each sensing agent is able to transmit.

The problem of maximizing the total throughput achieved by a set of regular nodes was considered by Srinivas and Modiano [50]. They developed an exact algorithm capable of optimally placing up to two mobile backbone nodes and assigning regular nodes to these mobile backbone nodes, under the assumption that all regular nodes communicating with a particular mobile backbone node achieve the same throughput. The algorithm developed in [50] is appropriate for arbitrary throughput functions, but is only applicable when $K \leq 2$.

This section develops an algorithm capable of solving this problem for arbitrarily many mobile backbone nodes, again under the assumption that all regular nodes communicating with a particular mobile backbone node achieve the same throughput, and with the additional assumption that the total throughput (or more generally, the utility of the information) received by a mobile backbone node is a concave function of the number of regular nodes assigned. This formulation models typical power adjustment measures taken to combat the near-far effect [41] in ad hoc networks and is applicable to many common throughput functions.




(a) Performance of the approximation algorithm developed in this section, relative to an exact solution technique, in terms of number of regular nodes assigned at the given throughput level.




(b) Computation time of the approximation algorithm and the exact (MILP) algorithm for various problem sizes.

Figure 3-8: Comparison of the exact and approximation algorithms developed in this chapter. Although the MILP-based exact algorithm developed in this chapter significantly outperforms existing techniques in terms of required computation time, experiments indicate that the greedy approximation algorithm achieves nearly the same level of performance with an even greater reduction in computation time.

Additionally, this section develops and solves a second formulation in which regular nodes can communicate with multiple mobile backbone nodes, and the throughput achieved by a regular node to a particular mobile backbone node is a function only of the distance between these two nodes. This formulation allows all regular nodes to transmit at full power and so is applicable to cases in which regular nodes cannot adjust their transmission power, or in which it is undesirable for them to do so.

### 3.3.1 Equalized Received Power

In this formulation, we assume that all regular nodes communicating with a particular mobile backbone node achieve the same throughput level. This assumption has both practical justification and computational advantages.

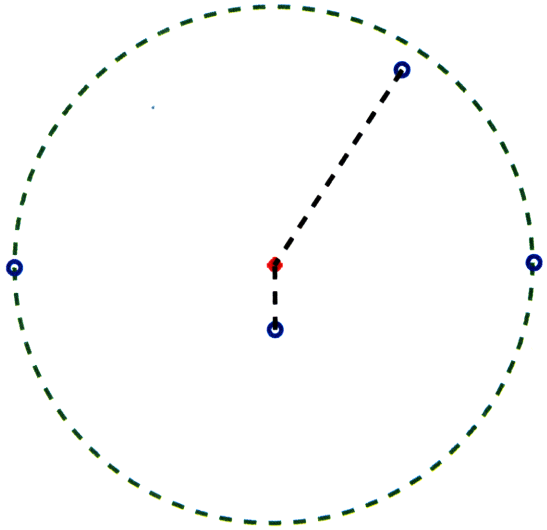
In situations in which the so-called *near-far effect* comes into play, this effect can be combated through transmission power adjustment on the part of the regular nodes [41]. The near-far effect occurs when the power received by a mobile backbone node from a nearby regular node is much greater than the power received from a distant regular node, hindering the communication capability between the mobile backbone node and the more distant regular node. This effect can be alleviated if all regular nodes communicating with a particular mobile backbone node adjust their power such that the mobile backbone node *receives* transmissions of the same power from each regular node.

This formulation has two computational advantages: first, it allows consideration of only the $O(N^3)$ 1-center locations for mobile backbone node placement, since every mobile backbone node can be placed at the 1-center of its assigned regular nodes in an optimal solution. Because all regular nodes achieve the same throughput as the most distant regular node, the total throughput of all regular nodes is clearly maximized if the distance to the most distant regular node is minimized.

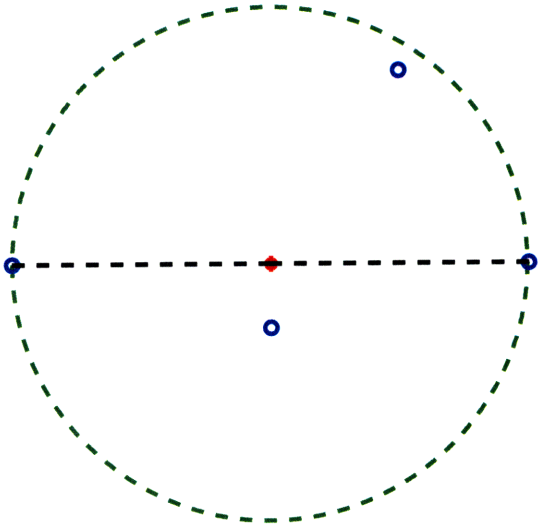
Second, this formulation simplifies the calculation of the aggregate throughput received by each mobile backbone node: the data received by a mobile backbone node can be regarded as a function of the *number* of regular nodes assigned to it, not the *particular* regular nodes that are assigned. At first glance, it would seem that the particular regular nodes

assigned to a mobile backbone node would have an impact on the aggregate throughput achieved. For instance, in Figure 3-9, the regular nodes assigned in Assignment 1 could clearly achieve a higher throughput level than those assigned in Assignment 2, even though they are assigned to a mobile backbone node at the same location, because the distance from the mobile backbone node to the most distant regular node is smaller in Assignment 1. Note, however, that Assignment 1 would never occur in an optimal solution: any exact algorithm (indeed, any algorithm that produces even a *locally optimal* solution with respect to mobile backbone node placement) would obtain a higher objective value by placing the mobile backbone node at the 1-center of the regular nodes assigned in Assignment 1 (as shown in Figure 3-10) rather than at the 1-center of a different set of regular nodes, as is the case in Assignment 1. Stated another way, at least one of the most-distant regular nodes within the 1-center radius of an optimally-placed mobile backbone node will *always* be assigned. (In fact, if more than one regular node is assigned, at least two assigned regular nodes will be most-distant; i.e., they will lie on the boundary defined by the 1-center radius of the mobile backbone node.) Therefore, it can be assumed *a priori* that all regular nodes adjust their transmission power such that the power received by the mobile backbone node is equal to the power received from the most distant regular node within the 1-center radius.

Then, the problem considered in Section 3.3.1 can be stated as follows: the objective is to *place* mobile backbone nodes, which are able to occupy any location in the plane but which can be restricted to occupy only 1-center locations without sacrificing optimality, while simultaneously *assigning* regular nodes to mobile backbone nodes, where each regular node can be assigned to at most one mobile backbone node, such that the sum of the utility of the data received by each mobile backbone node is maximized, where utility is a function of the total data received, and total data received is a function of the number of regular nodes assigned and the 1-center radius of the location where the mobile backbone node is placed. This problem formulation can be expressed as



(a) Assignment 1.



(b) Assignment 2.

Figure 3-9: Two possible assignments of regular nodes to a mobile backbone node. Black lines indicate assignment. Although Assignment 1 results in a greater aggregate throughput than Assignment 2, Assignment 1 would never occur in an optimal solution because the mobile backbone node is not placed at the 1-center of its assigned regular nodes. A better placement of the mobile backbone node is shown in Figure 3-10.

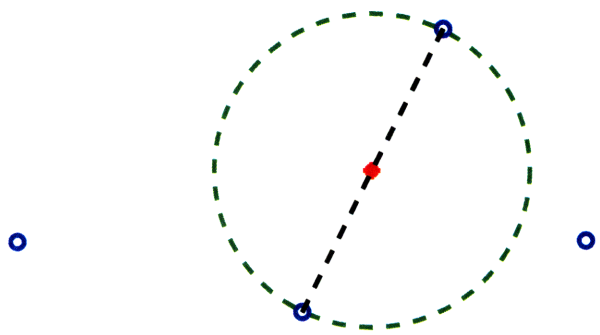


Figure 3-10: An improved assignment of the regular nodes assigned in Assignment 1. The mobile backbone node is placed at the 1-center of its assigned regular nodes.

$$\max_{L,A} \sum_{k=1}^{K} F_k(r_k(L),A) \tag{3.6a}$$

$$\text{subject to } A \in \mathscr{A} \tag{3.6b}$$

$$L \in \mathscr{L} \tag{3.6c}$$

where $L$ denotes the placement of mobile backbnode nodes, $A$ denotes the assignment of regular nodes to mobile backbone nodes, $F_k(r_k(L),A)$ is the utility of the data received by mobile backbone node $k$ as a function of the radius of the 1-center location where $k$ is placed (denoted by $r_k(L)$) and the assignment of regular nodes to mobile backbone nodes, $\mathscr{L}$ is the set of feasible placements of mobile backbone nodes, i.e., the placements such that each mobile backbone node is placed at a 1-center location and no two mobile backbone nodes are placed at the same 1-center location. $\mathscr{A}$ is the set of feasible assignments of regular nodes to mobile backbone nodes, i.e., those such that each regular node is assigned to at most one mobile backbone node.

The remainder of this section will demonstrate that Problem 3.6 can be solved exactly

via decomposition into two subproblems: placement of mobile backbone nodes (referred to as the placement subproblem), and assignment of regular nodes to mobile backbone nodes, given a placement (referred to as the assignment subproblem). We focus first on the assignment subproblem.

While it is most appropriate to maximize the sum the utility of the data received by each mobile backbone node, for simplicity, the remainder of the section will focus only on maximizing the total throughput. Note that this is a special case of the more general problem; utility is simply a linear function of throughput. However, the results obtained in this section hold for more general utility functions.

### 3.3.2 Assignment subproblem

Given a placement of mobile backbone nodes, the goal of the assignment subproblem is to assign each regular node to at most one mobile backbone node such that the total throughput is maximized. One important assumption is made in the solution of this problem: it is assumed that the total throughput received by a mobile backbone node is a *concave* function of the number of regular nodes assigned to it. In other words, for a fixed 1-center radius, the marginal increase in total throughput attained by assigning an additional regular node is a *nonincreasing* function of the number of regular nodes that have already been assigned.

Recall that in Section 3.2, the throughput achieved by a regular node was a nonincreasing function of both the *distance* between the regular node and its assigned mobile backbone node, and the *number* of other regular nodes communicating with that mobile backbone node and thus causing interference. In this section, the distance between a regular node and its assigned mobile backbone node is effectively replaced by the 1-center radius of the mobile backbone node, and again interference occurs when many regular nodes are assigned to the same mobile backbone node. Our assumption is that this interference is such that the total throughput received by a mobile backbone node is a concave function of the number of regular nodes assigned; or, equivalently, the marginal increase in total throughput achieved by assigning an additional regular node is a decreasing function of the number of regular nodes already assigned.

While this assumption may seem somewhat artificial, it is often satisfied in practice. To further explore the conditions necessary for the total throughput function to be concave in the number of regular nodes assigned, let $\tau_r(n)$ denote the *per-node* throughput achieved by *each* regular node when $n$ regular nodes are assigned to a mobile backbone node with 1-center radius $r$. The interference assumption tells us that $\tau_r(n+1) \leq \tau_r(n)$. Then, the *total* throughput achieved by *all* regular nodes is a concave function of the number of regular nodes assigned if and only if, for all $n$,

$$\begin{aligned} n\tau_r(n) - (n-1)\tau_r(n-1) &\geq (n+1)\tau_r(n+1) - n\tau_r(n) \\ 2n\tau_r(n) &\geq (n+1)\tau_r(n+1) + (n-1)\tau_r(n-1) \\ 2n\tau_r(n) &\geq n[\tau_r(n+1) + \tau_r(n-1)] + \tau_r(n+1) - \tau_r(n-1) \\ \tau_r(n) &\geq \frac{\tau_r(n+1) + \tau_r(n-1)}{2} + \frac{\tau_r(n+1) - \tau_r(n-1)}{2n}. \end{aligned}$$

This condition holds for many common and important throughput functions; for instance, the approximate Slotted Aloha function given in Section 3.2 and the CDMA-based throughput function given in Ref. [50]. Indeed, any throughput function such that $\tau_r(n)$ decreases at least as quickly as $\frac{1}{n}$ satisfies this condition.

In the case that utility is maximized rather than throughput, the equivalent condition would be concavity of utility as a function of the number of regular nodes assigned. This condition is naturally satisfied in some formulations of the exploration problem – for instance, the case in which each regular node transmits its own independent measurements of a set of features – but not necessarily in all formulations of the exploration problem.

Leveraging the concavity of the total throughput as a function of the number of regular nodes assigned, one can find an optimal assignment of regular nodes to mobile backbone nodes through the solution of a network flow problem on the multigraph shown in Figure 3-11. A *multigraph* is a graph in which multiple arcs can exist between a given pair of nodes. In the multigraph in Figure 3-11, multiple arcs connect each of the nodes representing mobile backbone nodes to the sink. Flows along these arcs represent assignment of regular

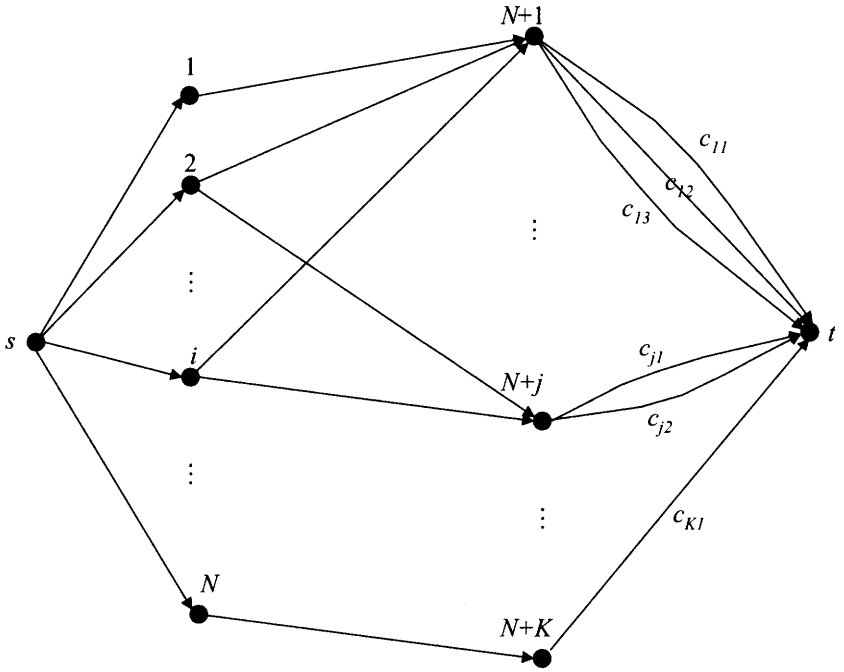



Figure 3-11: The network design problem for optimal assignment of regular nodes, given a placement of mobile backbone nodes. All arc capacities are equal to one.

nodes to the mobile backbone nodes; if $n$ of these arcs have positive flow for a particular mobile backbone node, this means that $n$ regular nodes have been assigned to that mobile backbone node. Note that each of these arcs has a cost $c_{ij}$ associated with it; this cost represents the marginal reward gained by the ability of the mobile backbone node $i$ to receive data from an additional ($j^{th}$) regular node. Assuming a concave utility function,this network flow problem will always produce a feasible assignment of regular nodes to mobile backbone nodes; moreover, an integer optimal solution always exists.

This network flow problem can be solved via the following linear program:

$$\max_{\mathbf{x}} \sum_{i=1\ldots K, j=1\ldots N_i} c_{ij} x^j_{N+i,t} \tag{3.7a}$$

$$\text{subject to} \sum_{(i,j)\in\mathscr{A}} x_{i,j} = \sum_{(j,k)\in\mathscr{A}} x_{j,k} \qquad j = 1,\ldots,N+K \tag{3.7b}$$

$$0 \le x_{i,j} \le 1 \qquad \forall\,(i,j) \in \mathscr{A} \tag{3.7c}$$

where $x_{i,j}$ is the flow from node $i$ to node $j$, $x^j_{N+i,t}$ is the flow along the $j^{th}$ arc from node

$N+i$ to node $t$, $N_i$ is the number of arcs from node $N+i$ to $t$ (the number of regular nodes covered by the mobile backbone node at location $i$), and $\mathscr{A}$ is the set of arcs.

### 3.3.3 Placement subproblem

Because the mobile backbone nodes can be restricted to the $O(N^3)$ 1-center locations, an exact algorithm for solving the overall simultaneous placement and assignment problem with computation time polynomial in $N$ consists of searching over all possible mobile backbone node placements, solving the linear program in Eq. 3.7 for each placement, and selecting the placement with the best objective value and an assignment corresponding to this value.

Denoting the $O(N^3)$ 1-center locations as $1,\ldots,M$ the selected locations of the $K$ mobile backbone nodes as $k_1,\ldots,k_K$, and the objective value in an optimal solution of problem 3.7 as a function of the mobile backbone node placement as $f(k_1 \ldots k_K)$, this search-based technique is given by Algorithm 2:

**Algorithm 2**

```
best_throughput ← 0
for k1 = 1...M − 2 do
  for k2 = k1 + 1...M − 1 do
    for k3 = k2 + 1...M do
      if f(k1,k2,k3) ≥ best_throughput then
        best_throughput ← f(k1,k2,k3)
        best_placement ← {k1,k2,k3}
      end if
    end for
  end for
end for
return best_placement
```

where $K = 3$ in this example (in general, there will be $K$ FOR loops in the algorithm), and where the optimal assignment is determined by the flows in the solution to problem 3.7 using the optimal placement.

Additionally, the simultaneous placement and assignment problem is amenable to solution using MILP, as was the placement problem in Section 3.2. The MILP formulation of

Table 3.2: Average computation times for the MILP-based and search-based algorithms, for various numbers of regular ($N$) and mobile backbone nodes ($K$) in the maximum fair placement and assignment (MFPA) problem, using ILOG CPLEX 9.020.

| $N$ | $K$ | MILP Algorithm | Search-based Approach |
|---|---|---|---|
| 3 | 2 | 5.2 sec | 1.8 sec |
| 4 | 2 | 10.8 sec | 18.5 sec |
| 5 | 2 | 21 sec | 82 sec |
| 6 | 2 | 37.8 sec | 298.6 sec |
| 6 | 3 | 55.8 sec | $>$ 30 min |
| 8 | 3 | 174.6 sec | $>$ 30 min |
| 10 | 3 | 246.0 sec | $>$ 30 min |

this problem is:

$$\max_{\mathbf{x}} \sum_{i=1\ldots K, j=1\ldots N_i} c_{ij} x^j_{N+i,t} \tag{3.8a}$$

$$\text{subject to} \sum_{(i,j)\in\mathscr{A}} x_{i,j} = \sum_{(j,k)\in\mathscr{A}} x_{j,k} \qquad j = 1,\ldots,N+M \tag{3.8b}$$

$$0 \le x_{i,j} \le 1 \qquad \forall\,(i,j)\in\mathscr{A} \tag{3.8c}$$

$$0 \le x_{N+i,t} \le y_i \qquad i = 1\ldots M \tag{3.8d}$$

$$\sum_{i=1\ldots M} y_i = K \tag{3.8e}$$

where the multigraph has been modified to include *all* possible mobile backbone node locations $i = 1\ldots M$, each of which is "activated" via a binary variable $y_i$.

Table 3.2 compares the computation time of the search-based method described by Algorithm 2 with the MILP-based method described by Eq. 3.8. As the table indicates, the MILP-based method tends to outperform the search-based method for problems of moderate size. However, if the application in which the algorithm will be used involves a fixed number of mobile backbone nodes and an unknown (and possibly very large) number of regular nodes, it may be desirable to use the search-based method due to the fact that the computation time of this method is polynomial in the number of regular nodes, unlike that of the MILP-based method. However, this benefit did not appear for the problem dimensions considered in this thesis (see Table 3.2).

Table 3.3: Relative performance of the greedy heuristic compared to an exact algorithm, in terms of the fraction of optimal total throughput achieved.

| $N$ | $K$ | Relative Performance of Greedy Approach |
|---|---|---|
| 3 | 2 | 0.98 |
| 4 | 2 | 0.98 |
| 5 | 2 | 0.97 |
| 6 | 2 | 0.96 |
| 6 | 3 | 0.96 |
| 8 | 3 | 0.99 |
| 10 | 3 | 0.97 |

Note that either the search-based method or the MILP-based method can easily be used to find the best placement and assignment such that *all* regular nodes are assigned to mobile backbone nodes. Depending on how the data rate is modeled as a function of distance, this may or may not be the configuration that results in the greatest total throughput or utility.

### 3.3.4 Greedy heuristic

As in Section 3.2, a greedy procedure for mobile backbone node location selection, where the mobile backbone node location that yields the greatest improvement in the maximum cost flow is selected in each round of selection, yields good empirical performance with improved theoretical computation time (polynomial in $N$ and $K$). However, in this case the computation time is not seen to be greatly reduced in practice. Table 3.3 summarizes the average performance of the greedy heuristic relative to that of the exact algorithms described in Section 3.3.1 for a set of randomly-generated problems of various sizes.

### 3.3.5 Time Division Multiple Access (TDMA)

In many applications, regular nodes can adjust their transmission power such that mobile backbone nodes receive transmissions at the same power level from all regular nodes. However, this is not always possible. In some applications, regular nodes simply "share" the limited resource of the channel to the mobile backbone node. One way in which multiple agents can cooperatively utilize a shared communication medium is through the use of a

*time division multiple access* (TDMA) protocol.

A TDMA protocol involves scheduling communications between regular nodes and mobile backbone nodes such that the communication channel is efficiently utilized and allocated among regular nodes. This section makes the following assumptions:

1. Each regular node can transmit to at most one mobile backbone node at a time.

2. Each mobile backbone node can receive from at most one regular node at a time.

3. Each regular node has a finite amount of data available to transmit. Otherwise, an optimal schedule would assign only the nearest regular node to communicate with a particular mobile backbone node, for any placement of mobile backbone nodes.

4. There is a finite time horizon over which all communications must take place. Otherwise, all available data could eventually be communicated for any placement of mobile backbone nodes.

The problem of interest in this section is to *place* mobile backbone nodes while simultaneously *scheduling* communications from regular nodes to mobile backbone nodes, such that the total amount of data communicated is maximized. As in Section 3.3.1, the problem can be decomposed into a placement subproblem and a scheduling subproblem. This section will focus first on the scheduling subproblem.

### 3.3.6 Scheduling subproblem

As described above, a feasible schedule of communication from regular nodes to mobile backbone nodes allows each regular node to transmit to only one mobile backbone node at a time, each mobile backbone node to receive from only one regular node at a time, and all communications to take place within a finite window of time (for convenience, this window is normalized to unit length). As a first step toward developing a feasible schedule, we first solve the simpler problem of finding an optimal *allocation* of transmission time for each regular node/mobile backbone node pair in which the only constraint is that the total time that any agent spends transmitting or receiving is at most one. This allocation can be found

via the following linear program:

$$\max_{\mathbf{t}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} t_{nk} \tag{3.9a}$$

$$\text{subject to} \sum_{n=1}^{N} t_{nk} \leq 1 \qquad k = 1, \ldots, K \tag{3.9b}$$

$$\sum_{k=1}^{K} t_{nk} \leq 1 \qquad n = 1, \ldots, N \tag{3.9c}$$

$$\sum_{k=1}^{K} \gamma_{nk} t_{nk} \leq d_n \qquad n = 1 \ldots N \tag{3.9d}$$

This linear program can be interpreted as solving a maximum flow problem on a *generalized network* such as the one shown in Figure 3-12, subject to side constraints limiting the total transmission time of each regular node to one. In a generalized network, each arc has a factor $\gamma$ associated with it. If one unit of flow enters the arc, $\gamma$ units of flow exit it. In this case, all arcs have $\gamma = 1$ with the exception of the arcs connecting the nodes representing regular nodes to the nodes representing mobile backbone nodes. The values of gamma for these nodes represent the data rates that can be achieved from the regular nodes to the mobile backbone nodes. Then, the goal of this problem is to transmit as much data as possible from the regular nodes to the mobile backbone nodes (3.9a), subject to the constraint that each regular node transmits for a total of at most one time unit (3.9b), each mobile backbone node receives for a total of at most one time unit (3.9c), and no regular node transmits more data than it has available (3.9d). The variable $\mathbf{t}$ represents flow, while $d_n$ represents the total amount of data available to transmit from regular node $n$.

The question of whether such an optimization can be used to generate a feasible communication schedule (i.e., one in which each regular node is transmitting to at most one mobile backbone node at any given time and each mobile backbone node is receiving from at most one regular node at any given time) in computationally tractable time is fundamentally related to the issue of *preemption*.

Preemption is an important notion in the machine scheduling literature. In machine scheduling problems, a set of *jobs* require processing on a set of *machines*. In a preemptive schedule, a job can be interrupted during its execution on a particular machine, another job

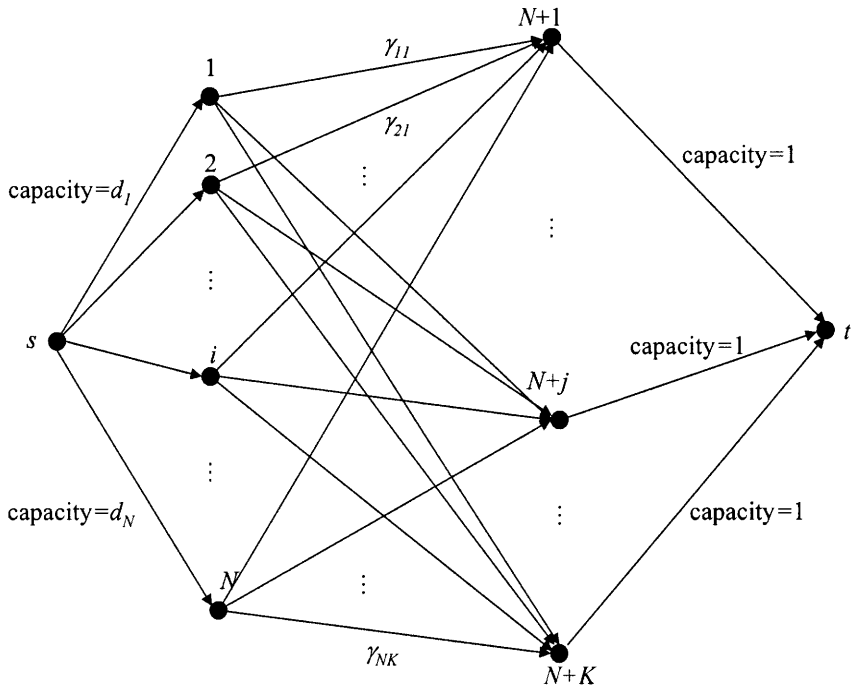



Figure 3-12: The network flow problem corresponding to the scheduling problem. Capacities of all arcs are equal to one, except the arcs originating at the source, which have capacities $d_1,\ldots,d_N$. Arcs connecting nodes $1,\ldots,N$ to nodes $N+1,\ldots,N+K$ have factors $\gamma_{nk}$ All other arcs have unit $\gamma$ (i.e., they are typical arcs).

can utilize the machine for processing, and the original job can later be re-started with no penalty (i.e., with no increase in total processing time). The relationship between communication scheduling and preemption can be illustrated through reduction of the communication scheduling problem to an *open shop scheduling problem*, a well-known variant of the traditional machine scheduling problem [21].

In the open shop scheduling problem, each job $n \in \{1,\ldots N\}$ requires processing on machine $k \in \{1,\ldots K\}$ for time $t_{nk}$. No job can be processed by more than one machine at a time, and no machine can process more than one job at once. Open shop scheduling differs from other machine scheduling problems, such as job shop and flow shop problems, in that there is no constraint on the order in which jobs must be completed, or the order in which jobs must be processed on various machines [21]. Variants of the open shop problem involve minimizing the time to complete all jobs (called the *makespan* of the schedule), completing as many jobs as possible before their respective *due dates* (possibly

with different *release dates* for each job), and maximizing the reward collected by time $T$, where each job has a given reward.

The relationship between communication scheduling to open shop scheduling is clear: regular nodes are represented by jobs, and mobile backbone nodes are represented by machines. Since there is no constraint on which mobile backbone nodes receive particular pieces of data, there is also no constraint on the order in which regular nodes transmit to mobile backbone nodes. Therefore, given an allocation of transmission times between regular nodes and mobile backbone nodes, the problem of scheduling communication between these nodes can be posed as an open shop scheduling problem.

Previous work has established that if preemption is not allowed, it is NP-hard to generate an optimal schedule for the open shop scheduling problem [21]. However, if preemption is allowed, it has been shown that there always exists a feasible schedule with a makespan that is equal to the maximum total time required by any job or machine (in the case of mobile backbone network communication scheduling, the maximum communication time of any regular node or mobile backbone node) [21]. Moreover, such a schedule can be found in polynomial time [21]. Thus, if communications can be preempted without penalty, the linear program in Eq. 3.9 can be used to generate optimal communication times for each regular node-mobile backbone node pair, such that a feasible communication schedule can be found in polynomial time.

In the communication scheduling problem, a preemption occurs when one regular node must stop its transmission to a mobile backbone node so that another regular node can transmit to that mobile backbone node. Because there is little time lost in the process of stopping transmissions or switching frequencies, is it not unreasonable to assume that preemption carries little detriment in the communication scheduling problem.

However, in order to properly interpret the data it receives from the regular nodes, it is important that each mobile backbone node is able to associate received data with the particular regular node that transmitted it. A header can be appended to the beginning of each transmission in order to identify the regular node that transmitted it; however, such a header inevitably increases the amount of time required to complete the transmission. If the number of preemptions is large, the total fraction of transmission time devoted to

transmitting headers can become significant. Fortunately, it has been shown that the number of preemptions that take place can indeed be bounded and is in fact small. Lawler and Labetoulle demonstrate that at most $4K^2 - 5K + 2$ preemptions occur in a time-optimal schedule (recall the $K$ is the number of machines/mobile backbone nodes); furthermore, they give a polynomial-time algorithm for constructing such a schedule. [2]

Thus, the linear program given by Eq. 3.9 can be modified to ensure that all communication takes place within the specified timeframe by modifying it as follows:

$$\max_{\mathbf{t}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} t_{nk} \tag{3.10a}$$

$$\text{subject to } \sum_{n=1}^{N} t_{nk} \leq 1 - (4K^2 - 5K + 2)\delta_t \qquad k = 1, \ldots, K \tag{3.10b}$$

$$\sum_{k=1}^{K} t_{nk} \leq 1 - (4K^2 - 5K + 2)\delta_t \qquad n = 1, \ldots, N \tag{3.10c}$$

$$\sum_{k=1}^{K} \gamma_{nk} t_{nk} \leq d_n \qquad n = 1 \ldots N \tag{3.10d}$$

where $\delta_t$ is the additional time needed to perform a "handoff" at each preemption, and the total duration of the timeframe has been decreased by a factor of $(4K^2 - 5K + 2)\delta_t$. This factor is constant for fixed $K$ and is therefore particularly appropriate for situations in which $K$ is known *a priori*, but the number of regular nodes and the timeframe of communication may be large.

Note that this formulation is conservative, since in many instances, fewer than $4K^2 - 5K + 2$ preemptions will occur. In order to maximize the efficiency of the resulting schedule, one can efficiently search over the possible values for the number of preemptions (using, for example, a binary search) until the feasible schedule with the smallest number of preemptions if found.

[2] The problem addressed by Lawler and Labetoulle, in terms of the communication scheduling problem, actually minimizes the time required to collect all data from the regular nodes and could easily be used for this purpose.

### 3.3.7 Placement

Although the scheduling portion of the problem can be efficiently solved using insights gained from the open shop machine scheduling problem, the placement portion of the problem remains challenging. Because each regular node is able to transmit at full power and achieve its maximum possible data rate to the mobile backbone node to which it is transmitting, the mobile backbone nodes can no longer be restricted to 1-center locations in an optimal solution. They can instead be placed anywhere in the plane, making the overall placement and assignment problem a non-convex optimization problem.

By adopting a discretization procedure one can reduce the complexity of the problem while maintaining a bounded discretization error. Depending on the desired accuracy of the solution, an $\varepsilon$ can always be found such that a spatial discretization of possible mobile backbone node locations at intervals of $\varepsilon$, followed by a search over all possible mobile backbone node placements in which an optimal schedule is found for each placement, yields a solution in which each regular node is a distance of at most $\sqrt{2}\varepsilon$ farther from each mobile backbone node than in an optimal solution. Unfortunately, this approach does not scale well with the spatial dimensions of the problem. Future research on this problem should address intelligent selection of candidate mobile backbone node locations.

## 3.4 Summary

This chapter has introduced the network design formulation of mobile backbone optimization, a powerful and intuitive tool for solving network optimization problems under a variety of objectives. Is has then leveraged this formulation to develop new algorithms for mobile backbone network optimization for the case in which the objective is to maximize the number of regular nodes that achieve throughput at least $\tau_{min}$, as well as for situations in which the total throughput (or the utility of the total throughput) is maximized.

For the case in which the objective is to maximize the number of regular nodes that achieve throughput at least $\tau_{min}$, both an exact MILP-based technique and the first known approximation algorithm with computation time polynomial in the number of regular nodes and the number of mobile backbone nodes were described. This approximation algorithm

is based on the submodularity of the objective in the network design problem.

Based on simulation results, this chapter has shown that the MILP-based approach provides a considerable computational advantage over existing search-based techniques for mobile backbone network optimization. This approach has been successfully applied to a problem in which a maximum number of regular nodes are to be assigned to mobile backbone nodes at a given level of throughput, as well as to a related problem in which all regular nodes are to be assigned to a mobile backbone node such that the minimum throughput achieved by any regular node is maximized.

For cases in which a MILP approach is impractical due to constraints on computation time, the greedy approximation algorithm developed in this chapter presents a viable alternative. This algorithm carries the benefit of a theoretical performance guarantee, and simulation results indicate that it performs very well in practice.

This chapter has also developed algorithms for maximizing the total throughput achieved by all regular nodes. Two cases were considered: the case in which all regular nodes transmitting to the same mobile backbone node adjust their transmission power in order to combat the near-far effect, and the case in which all regular nodes transmit at full power, but schedule their transmissions such that each mobile backbone node receives data from at most one regular node at a time.

For the case in which regular nodes are able to adjust their transmission power, an exact algorithm with computation time polynomial in the number of regular nodes was developed. This algorithm was compared with a second, MILP-based exact algorithm, as well as with a greedy heuristic algorithm. Simulation results showed that the MILP-based technique tends to outperform the search-based method in terms of computation time, although both are most appropriate for problems of moderate scale. The greedy heuristic, which is most computationally efficient, was demonstrated to perform almost as well as the exact techniques in terms of total throughput achieved.

For the case in which regular nodes transmit at a fixed power level, reduction to the open shop scheduling problem facilitated the formulation of linear programming-based algorithms for optimally allocating transmission times for all regular nodes and mobile backbone nodes. The linear program was also modified to account for time delays caused

by preemption in the resulting schedule.

In the next chapter, the algorithms presented in Section 3.2 will be extended to the case in which regular nodes are capable of moving. This will facilitate integration of these algorithms into an architecture for exploration.

# Chapter 4

# Mobile Network Optimization and the Exploration Problem

The problem formulations of the previous chapter assumed that the locations of regular nodes are fixed *a priori* and that only the locations of mobile backbone nodes are variable [16, 50, 51]. This assumption is reasonable for some applications, such as scenarios that involve mobile agents extracting data from a fixed sensor network. In some previous work, mobile backbone nodes were deployed to provide communication support for mobile but uncontrolled regular nodes whose trajectories were known [49]. However, in many applications the locations of both regular nodes and mobile backbone nodes can be controlled. For example, consider a cooperative exploration mission being executed by a heterogeneous team of air and ground vehicles. The ground vehicles can move and can accurately sense phenomena at ground level, while the air vehicles are more mobile and are better equipped to communicate over long distances.

This chapter develops a modeling framework and solution technique that are appropriate for problems in which the motion of regular nodes can be controlled. In this framework, $L$ candidate regular node locations are available *a priori*, perhaps selected by heuristic means or due to logistical constraints. Each of $N$ regular nodes ($N \leq L$) must occupy one of these locations, and no two regular nodes can be assigned to the same location. Given an initial location and a mobility constraint, each regular node is capable of reaching a subset of the other locations. There are $K$ mobile backbone nodes ($K \leq N$) that can be placed

anywhere, a throughput function $\tau$ is specified, and a desired minimum throughput $\tau_{min}$ is given.

Given these assumptions, the goal of this section is to *place* both the regular nodes and mobile backbone nodes while simultaneously *assigning* regular nodes to mobile backbone nodes in order to maximize the number of regular nodes that are successfully assigned and achieve the desired minimum throughput level $\tau_{min}$, under the given throughput function $\tau$. Denoting the problem data as $L_i$ (the initial locations of the regular nodes) and L (the set of all locations); and the decision variables as $L_r$ (the selected locations of the regular nodes), $L_m$ (the selected locations of the mobile backbone nodes), and $A$ (the assignment of regular nodes to mobile backbone nodes), this optimization problem can be stated as:

$$\max_{L_r, L_m, A} \; F_\tau(L_r, L_m, A, \tau_{min})$$
$$\text{subject to } L_r \in r(L_i)$$

where $F_\tau(L_r, L_m, A, \tau_{min})$ is the number of regular nodes that achieve throughput level $\tau_{min}$, given node placements $L_r$ and $L_m$, assignment $A$, and throughput function $\tau$; and $r(L_i)$ denotes the set of regular node placements reachable from $L_i$ under the regular node mobility constraints.

In this problem formulation, regular node placement can greatly impact the quality of the network that can be achieved through mobile backbone node placement and assignment. Note that other considerations can also come into play when placing mobile backbone nodes; for instance, sensor data available at some locations can be more useful than sensor data available at other locations. Considerations such as sensor data quality can be modeled in the framework presented in this chapter. However, for clarity, this chapter will focus only on optimizing network throughput under the assumption that all candidate regular node locations are equally good in other respects.

Additionally, this chapter only considers a step utility function; i.e., regular nodes that achieve throughput at least $\tau_{min}$ have unit utility, and those that do not achieve $\tau_{min}$ have zero utility, as in Section 3.2. However, the other results of Chapter 3 also hold in the case of mobile regular nodes.

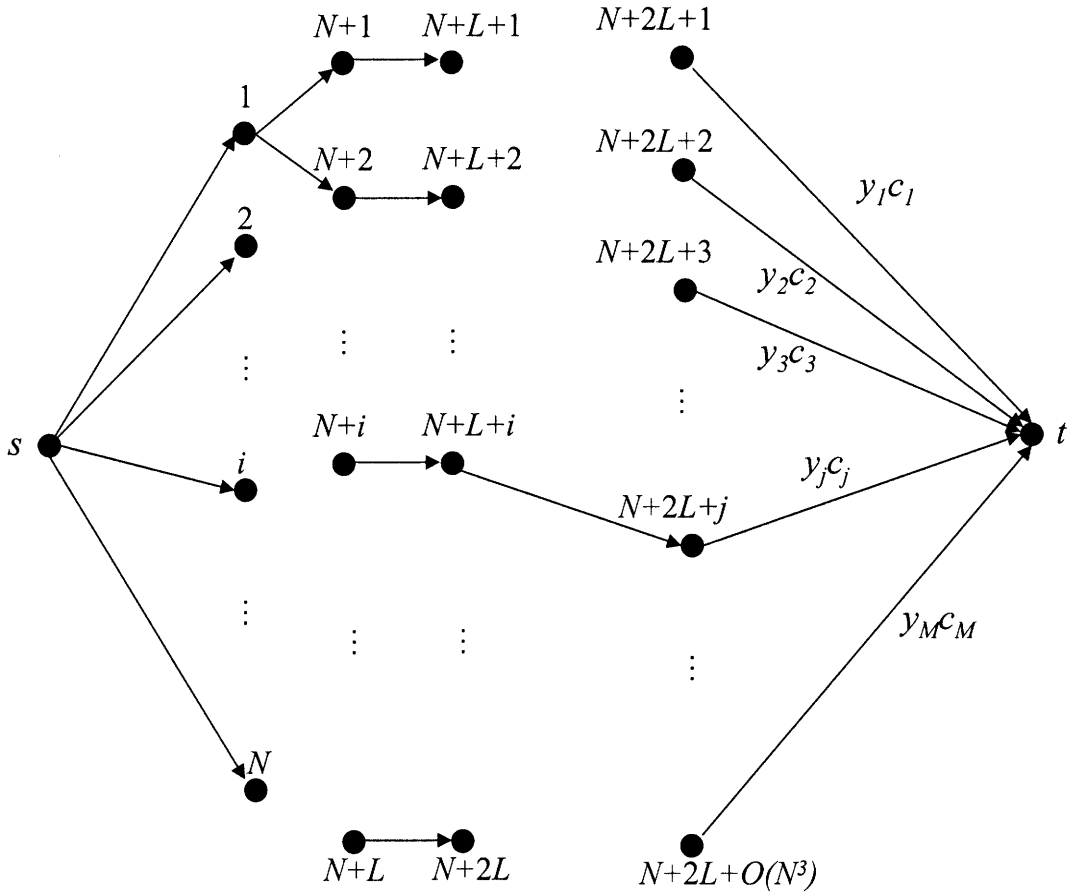



Figure 4-1: The network design problem corresponding to the joint placement and assignment problem for mobile backbone networks, with regular node mobility. Unlabeled arc capacities are equal to one. For clarity, not all arcs and nodes are shown.

## 4.1 Network design formulation

As in Section 3.2, optimal placement of mobile backbone nodes and simultaneous placement and assignment of regular nodes is achieved through the solution of a network design problem.

The network graph over which this optimization takes place is schematically represented in Figure 4-1. This graph is constructed as follows: the source node, $s$, is connected via an arc of unit capacity to each of a set of nodes N= $\{1,\ldots,N\}$, which represent the initial locations of the $N$ regular nodes. Each of these nodes, in turn, is connected via an arc of unit capacity to a subset of nodes in L= $\{N+1,\ldots,N+L\}$. Node $i$ is connected to node $N+j$ if and only if regular node $i$ can reach sensing location $j$ under its mobility constraint. Next, each of the nodes in L is connected to a copy of itself in set L$'$= $\{N+L+1,\ldots,N+2L\}$, and again these arcs are of capacity one. This duplication is done in order to enforce the constraint that only one regular node can occupy each sensing location. The portion of the graph described thus far models regular node placement.

The remainder of the graph models mobile backbone node placement, as well as assignment of regular nodes to mobile backbone nodes. Each of the nodes in L$'$ is connected via an arc of unit capacity to a subset of the nodes in M= $\{N+2L+1,\ldots,N+2L+M\}$, which represent possible mobile backbone node locations. (Recall that, although mobile backbone nodes can be placed at arbitrary locations, only $M$ 1-center locations need to be considered, where $M$ is $O(L^3)$.) Node $N+L+i$ is connected to node $N+2L+j$ if and only if sensing location $i$ is within the radius of the 1-center associated with $j$. (Recall that each 1-center location has both a center and a radius associated with it.)

Finally, each node in M is connected to the sink $t$. The capacity of the arc from node $N+2L+i$ to $t$ is the product of a binary variable $y_i$, which represents the decision of whether to "purchase" this arc, and a constant $c_i$, which is again the floor of the inverse with respect to cluster size of the throughput function, evaluated at the desired minimum throughput level, i.e., the maximum number of regular nodes that can be assigned to a mobile backbone node at location $i$ and achieve the desired throughput level.

Note that any feasible solution to this network design problem represents a feasible

placement and assignment of regular nodes and mobile backbone nodes; likewise, any feasible placement and assignment of regular nodes and mobile backbone nodes also determines a feasible flow in the graph. Therefore, an optimal solution to this network design problem yields an optimal solution to the simultaneous placement and assignment problem.

Denote the set of nodes in the network design graph by $\mathcal{N}$ and the set of arcs by $\mathcal{A}$. If $K$ mobile backbone nodes are available and a minimum throughput level is specified, the goal of the network design problem is to select $K$ arcs from $\{N+2L+1,\ldots,N+2L+M\}$ and a feasible flow $x_{ij}$, $(i,j) \in \mathcal{A}$ such that the $s-t$ flow is maximized. This problem can be solved via the following mixed-integer linear program (MILP):

$$\max_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{N} x_{si} \tag{4.2a}$$

$$\text{subject to} \sum_{i=1}^{M} y_i \leq K \tag{4.2b}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij} = \sum_{l:(l,i)\in\mathcal{A}} x_{li} \qquad i \in \mathcal{N} \setminus \{s,t\} \tag{4.2c}$$

$$x_{ij} \geq 0 \qquad \forall\, (i,j) \in \mathcal{A} \tag{4.2d}$$

$$x_{ij} \leq 1 \qquad \forall\, (i,j) \in \mathcal{A} : j \in \mathcal{N} \setminus \{t\} \tag{4.2e}$$

$$x_{(N+2L+i)t} \leq y_i c_i, \qquad i \in \{1,\ldots,M\} \tag{4.2f}$$

$$y_i \in \{0,1\} \qquad i \in \{1,\ldots,M\} \tag{4.2g}$$

where the constraints state that at most $K$ arcs (mobile backbone node locations) can be selected (4.2b), flow through all internal nodes must be conserved (4.2c), arc capacities must be observed (4.2d- 4.2f), and $y_i$ is binary for all $i$ (4.2g).

Figure 4-2 shows an example of a solution to the simultaneous placement and assignment problem with regular node movement. The regular nodes, initially in positions indicated by •, are able to move to other locations (○) within their radii of motion, indicated by shaded pink circles. This initial configuration is shown in Figure 4-2(a). In an optimal solution to this problem, shown in Figure 4-2(b), the regular nodes have moved such that they are grouped into compact clusters for which the mobile backbone nodes can provide

an effective communication infrastructure. The clusters are relatively balanced, in that the clusters with larger radii tend to have fewer regular nodes, while the more compact clusters can accommodate more regular nodes and still achieve the desired minimum throughput. In this example, all regular nodes have been successfully assigned to mobile backbone nodes.

We make the following remarks about this algorithm:

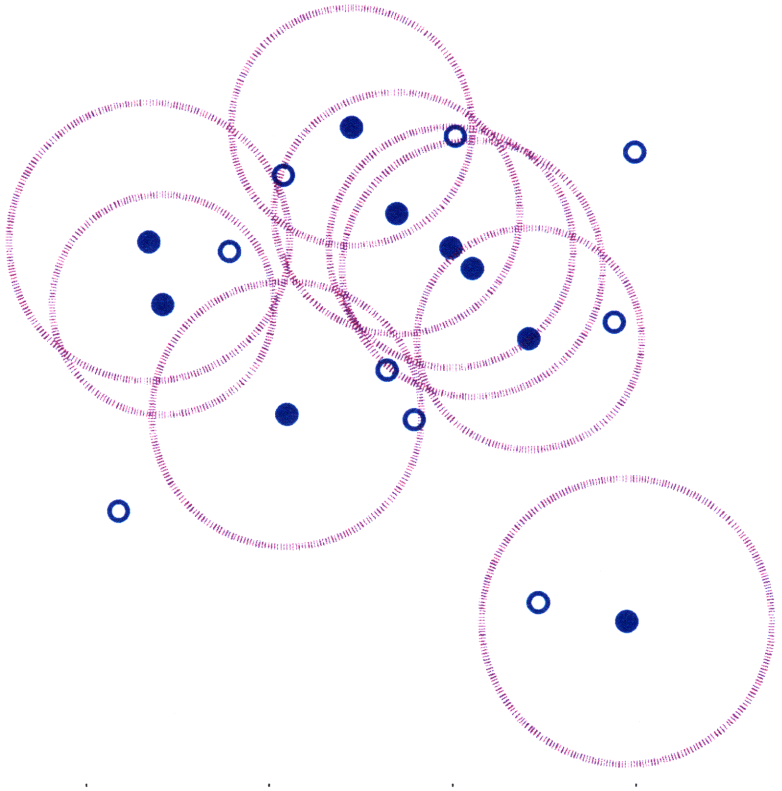
*Remark 1:* This algorithm is designed to maximize the number of regular nodes that are assigned at throughput level $\tau_{min}$. If, instead, the goal is to achieve the best possible minimum throughput such that *all* regular nodes are assigned to a mobile backbone node (i.e., to solve the MFPA problem), it is necessary to solve the MILP problem in Eq. 4.2 $O(\log(NL))$ times for different throughput values (which result in different values for the $c_i$'s in the network design problem).

*Remark 2:* If arbitrarily many mobile backbone nodes are available and the goal is to achieve a desired minimum throughput while utilizing a minimal number of mobile backbone nodes, then a MILP problem similar to the one in Eq. 4.2 needs only to be solved once for the values of $c_i$ corresponding to the desired throughput. The problem must be modified so that the number of mobile backbone nodes used is minimized, subject to the constraint that the flow through the graph is equal to the number of regular nodes.

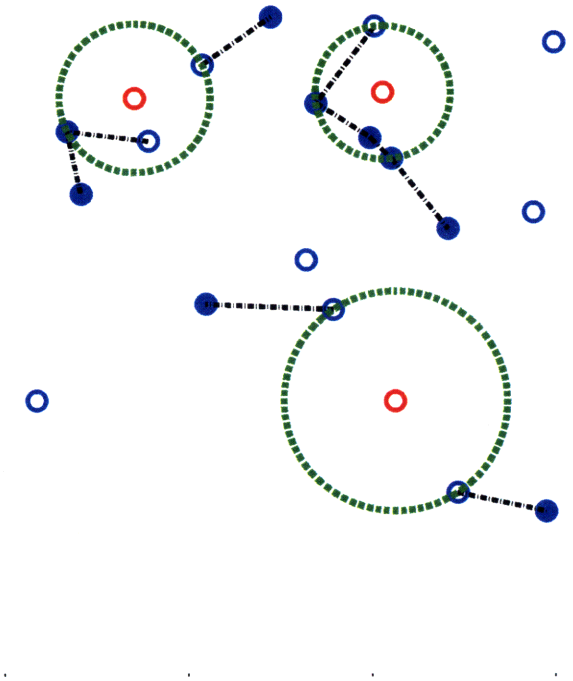
*Remark 3:* It should be noted that the worst-case complexity of mixed-integer linear programming is exponential in the number of binary variables. However, this approach performs well in practice. Table 4.1 shows the computation time of the MILP algorithm when applied to the MFPA problem described in Remark 1. Note that this problem requires repeated solution of the MILP; for problems that do not require repeated solution of the MILP, the algorithm is therefore faster. As the table indicates, this method is appropriate for problems of moderate scale.

### 4.1.1 Hardness of Network Design

As in Chapter 3, the network design problem formulated in this chapter is NP-hard in general.



(a) Initial regular node placement, with radius of motion for each regular node.



(b) An optimal placement of regular and mobile backbone nodes.

Figure 4-2: A small example of mobile backbone network optimization with limited regular node movement. Open blue circles represent possible regular node locations, and filled blue circles are the positions of the regular nodes. Shaded pink circles in the left figure indicate the possible radius of motion of each regular node. In the right figure, mobile backbone nodes, shown in red, are placed such that they provide communication support for the regular nodes. Each regular node is assigned to at most one mobile backbone node. Dotted lines indicate regular node motion in this optimal solution. Dashed circles indicate the radius of each cluster of nodes. In this example, all regular nodes have been successfully assigned to mobile backbone nodes.

Table 4.1: Average computation times for various values of $N$, $K$ and $L$, for the MFPA problem.

| $N$ | $K$ | $L$ | MILP Algorithm with Regular Node Movement |
|---|---|---|---|
| 4 | 2 | 10 | 10 sec |
| 6 | 2 | 15 | 16.5 sec |
| 8 | 3 | 20 | 80 sec |

Defining a four-layered graph analogously to the two-layered graph in Section 3.2.4, we have the following theorem:

**Theorem 4.1.1** *Network design on four-layered graphs is NP-hard.*

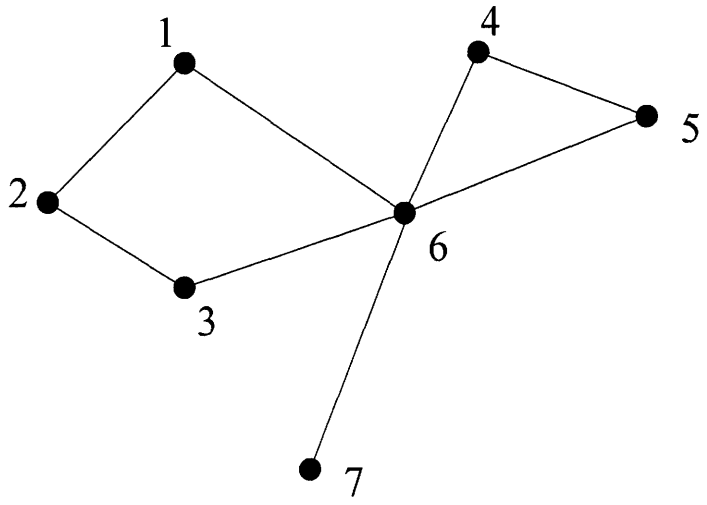
**Proof 4.1.1** *The same reduction from K-vertex cover shown in Section 3.2.4 can be made, with the addition of two layers of "dummy nodes" of cardinality equal to $|E|$, as shown in Figure 4-3.*

## 4.2 Approximation Algorithm

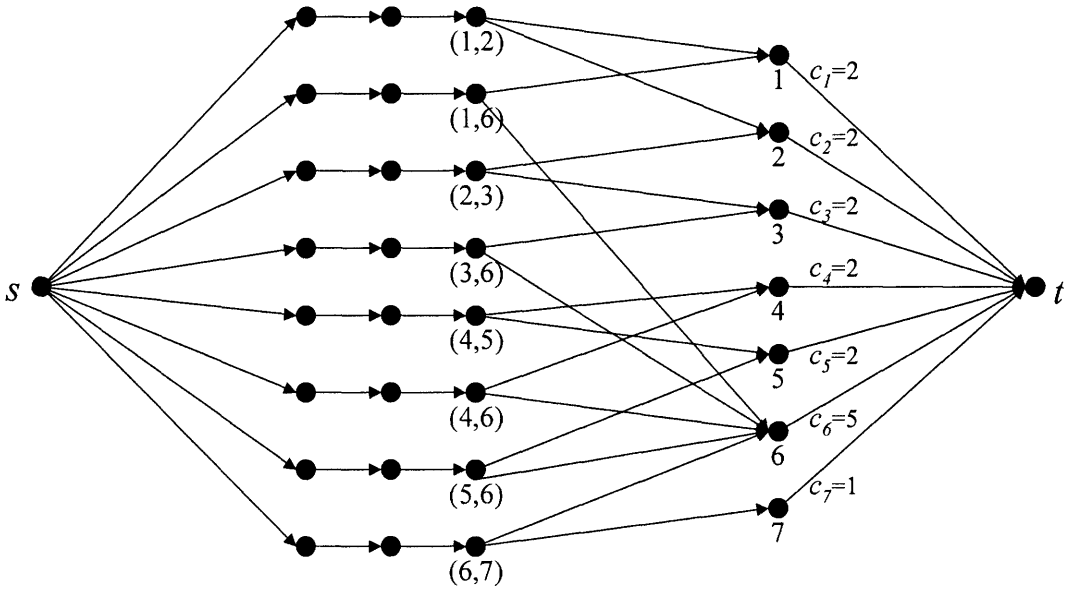
While the MILP-based algorithm described in Section 4.1 is computationally tractable for problems of moderate scale, its worst-case computation time is exponential in the number of binary variables. Therefore, this section develops an approximation algorithm for this problem that is appropriate for problems of larger scale.

This approximation algorithm is again based on the insight that the number of regular nodes that can be placed and assigned is a *submodular* function of the set of mobile backbone node locations that are selected.

This observation motivates consideration of a greedy algorithm (Algorithm 3) for the problem of maximizing the number of regular nodes that achieve throughput level $\tau_{min}$, where both regular nodes and mobile backbone nodes are mobile. Given a network design graph $G$, $K$ mobile backbone nodes and $M$ possible mobile backbone node locations, and denoting by $f$ the maximum flow through $G$ as a function of the set of mobile backbone node locations selected, this greedy algorithm is:




(a) A graph given as input in the $K$-vertex cover problem.




(b) A four-layered graph in the network design problem to which the $K$-vertex cover problem is reduced.

Figure 4-3: An example of conversion from a $K$-vertex cover problem to an network design problem.

**Algorithm 3**

$S \leftarrow \emptyset$
$maxflow \leftarrow 0$
**for** $k$=1 to $K$ **do**
  **for** $m$=1 to $M$ **do**
    **if** $f(S \cup \{m\}) \geq maxflow$ **then**
      $maxflow \leftarrow f(S \cup \{m\})$
      $m^* \leftarrow m$
    **end if**
  **end for**
  $S \leftarrow S \cup \{m^*\}$
**end for**
**return** $S$

The following theorem describes the performance of Algorithm 3:

**Theorem 4.2.1** *Algorithm 3 returns a solution S such that* $f(S) \geq \lceil (1 - \frac{1}{e}) \cdot f(S^*) \rceil$, *where* $S^*$ *is the optimal solution to the network design problem on G.*

**Proof 4.2.1** *This follows from the observation that all maximum flows through G are integer, and from the following Lemma:*

**Lemma 4.2.2** *The maximum flow that can be routed through G is a submodular function of S, the set of arcs that are selected.*

For purposes of proving the submodularity of the network design objective function, the maximum flow problem of Section 4.1 will be reformulated as a set-to-set node disjoint path problem in a modified version of the maximum flow graph. A set-to-set node disjoint path problem specification consists of a directed graph, $H$, and designations of subsets of the nodes of $H$ as the source set and the destination set. The goal of a set-to-set node disjoint path problem is to find the maximum number of paths originating in the source set and terminating in the destination set, such that no node in $H$ is traversed by more than one path.

The modification of the maximum flow graph to the graph induced by the corresponding set-to-set node disjoint path problem is accomplished as follows: the $s$ and $t$ nodes are removed, and node set N remains unchanged. Node sets L and L$'$ are compressed into a

single set L; since the problem under consideration is a node disjoint path problem, there is no need to enforce the node capacity constraint using a duplicate set of location nodes, as in the maximum flow problem. Set M is modified in the following way: if a node $m \in$M in the maximum flow problem has outgoing capacity $c$, then node set M in the modified graph itself contains a *set* of nodes $m$ consisting of $c$ copies of this node, each of which is connected to the same nodes in L as the original node $m$. An example of this reformulation is shown in Figure 4-4.

The source set in this problem is N, and the destination set is M. Note that any configuration of set-to-set node disjoint paths in this modified graph has a corresponding feasible flow in the maximum flow problem. Likewise, any feasible flow in the maximum flow problem defines a set of node disjoint paths in the modified problem. Therefore, the maximum flow in the original problem is equal to the maximum number of node disjoint paths in the modified problem.

To show that the maximum flow through $G$ is a submodular function of the set of arcs that are selected, we will prove that the maximum number of node disjoint paths in $H$ is a submodular function of the set of destination nodes. A restatement of the submodularity condition is:

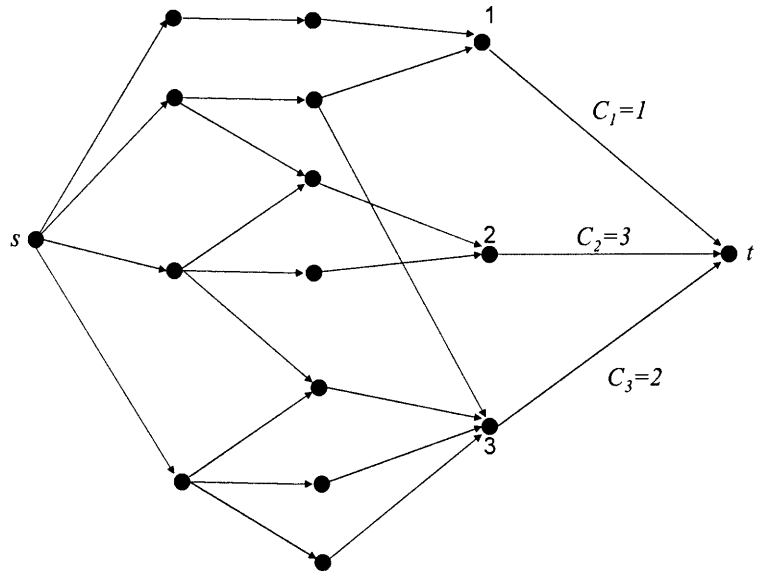
$$f(S \cup \{i,j\}) + f(S) \leq f(S \cup \{i\}) + f(S \cup \{j\}).$$

The relevant maximum flow graphs for this relation are shown at the top of Figure 4-5: the sum of the maximum flows through the left two graphs must be less than or equal to the sum of the maximum flows through the right two graphs.

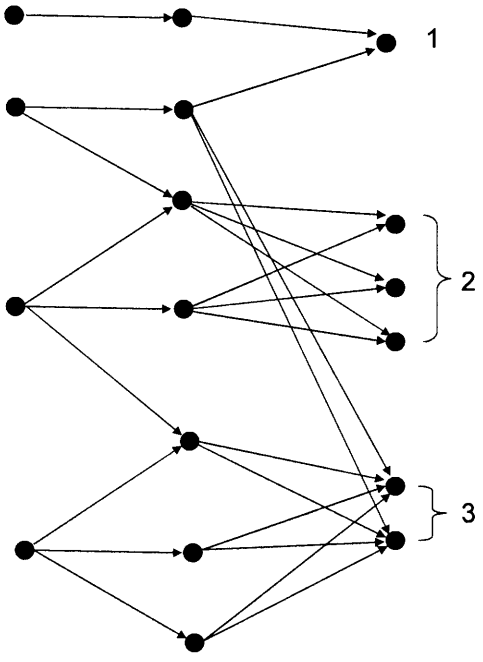
Converting these maximum flow problems into their equivalent node disjoint path problems yields the graphs shown at the bottom of Figure 4-5. The submodularity condition states that the maximum number of node disjoint paths in the left two graphs is at most the maximum number of node disjoint paths in the right two graphs. Denote these graphs from left to right by $H_1$, $H_2$, $H_3$ and $H_4$.

We will also make use of the following fact:

**Fact 4.2.1** *If $H$ is a graph of the form shown in Figure 4-4(b), with source set N and*




(a) Graph induced by a maximum flow problem. For clarity, node sets L and L′ present in Figure 3.2.3 have been replaced with a single node set, with the restriction that at most one unit of flow may traverse each of these nodes.




(b) Graph induced by a set-to-set node disjoint path problem.

Figure 4-4: An example of conversion from a maximum flow problem to an equivalent set-to-set node disjoint path problem.

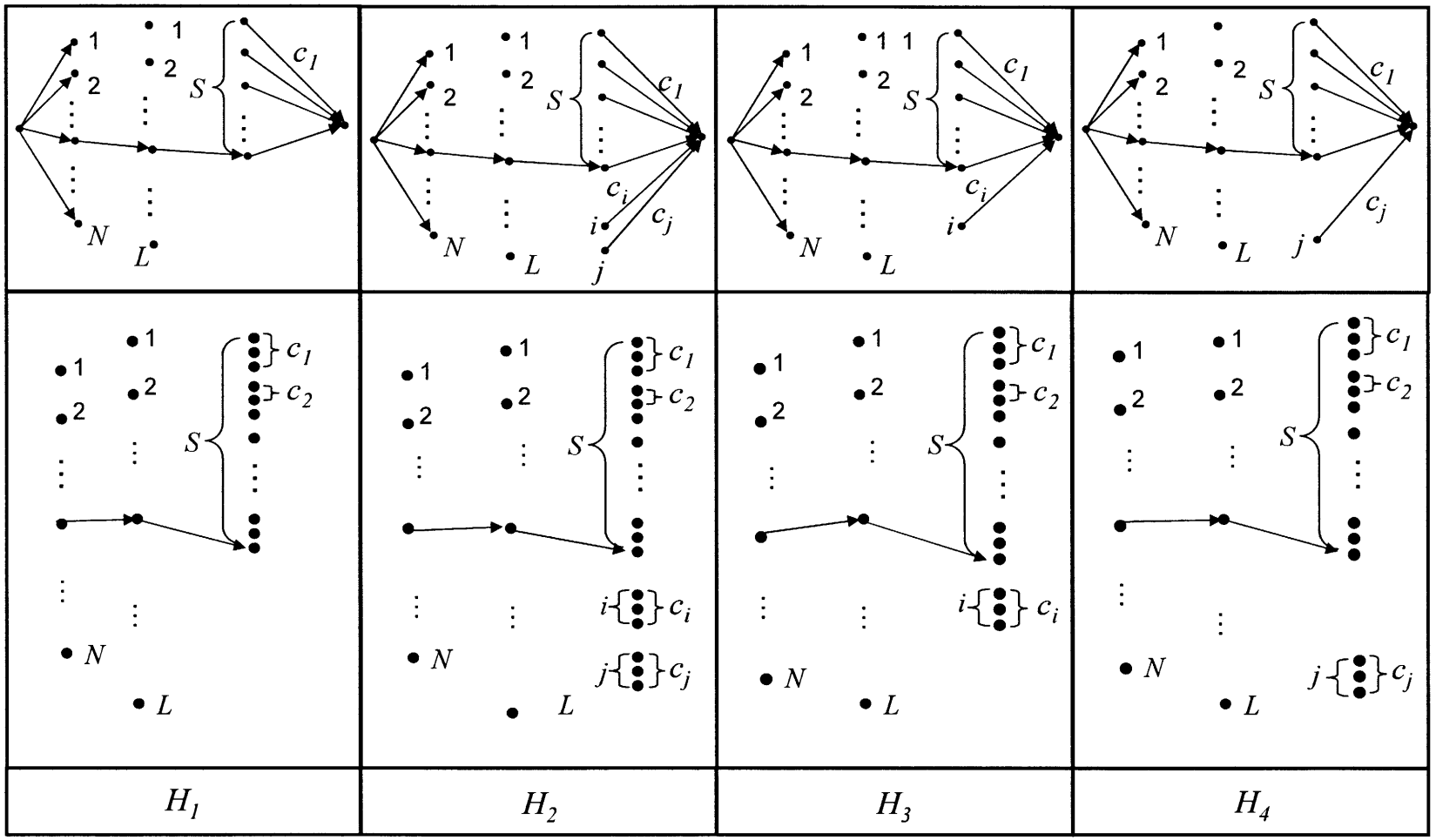



Figure 4-5: Schematic representation of the graphs involved in the proof of Lemma 4.2.2. The top four graphs are for the original maximum flow problem, while the bottom four graphs are their equivalent reformulations in the node disjoint path problem. For clarity, not all arcs are shown.

*destination set M, and P is a set of node disjoint paths in H that covers a subset D of the destination nodes, there exists a maximum set of node disjoint paths $P'$ in H that also covers node set D.*

To see that this is true, consider a maximum set of node disjoint paths $P''$ in $G$ that does not cover all of the nodes in $D$, and modify it as follows: for each node $i$ in $D$ that was covered by $P$, examine the node from the middle column that was used in $P$. If this node is used in a path leading to node $j \neq i$ in $P''$, then the second segment of its path may be redirected to node $i$. This operation results in no net gain or loss of node disjoint paths, so the current solution is still optimal. If the node that was connected to $i$ in $P$ is not used in $P''$, then the entire path beginning at the source node for the path in $P$ ending in $i$ may be redirected to $i$, again with no net gain or loss in total paths. Note that this operation might uncover a node that was covered in an iteration of the first operation, but this uncovered node can still be re-covered by a path originating at its source node from $P$. At this point it cannot be uncovered again. If this process is repeated, it eventually covers all all destination nodes that were originally in $P$, and the modified version of $P''$ remains optimal after the modification. This modified version of $P''$ is $P'$.

Lemma 4.2.2 can now be proved.

**Proof 4.2.2** *Making use of the reformulation of the maximum flow problem as a node disjoint path problem, the claim of the lemma can be restated as follows: if $P_i$ denotes a maximum set of node disjoint paths in graph $H_i$ from Figure 4-5 for $i = 1, \ldots, 4$, and $|P_i|$ denotes the cardinality of $P_i$ (i.e., the number of elements from the source or destination sets covered by $P_i$), then $|P_1| + |P_2| \leq |P_3| + |P_4|$.*

*Consider a maximum set of node disjoint paths $P_1$ in graph $H_1$, and denote its cardinality by $N_s$. Note that $P_1$ is a feasible set of node disjoint paths for graph $H_2$ as well.*

*Because $P_1$ is feasible in graph $H_2$, there is a maximum set of node disjoint paths in $H_2$ that covers the same set of destination nodes in S as $H_1$. Call this optimal solution $P_2$. Denote the number of nodes covered by $P_2$ in node sets i and j by $N_i$ and $N_j$, respectively. Then, the total number of node disjoint paths in $P_1$ and $P_2$ is equal to $2N_s + N_i + N_j$.*

*Now consider the set of node disjoint paths obtained by removing the paths ending in*

*node set $j$ from $P_2$. Note that this set of node disjoint paths is feasible for graph $H_3$, and its cardinality is $N_s + N_i$. Likewise, the set of node disjoint paths obtained by removing the paths ending in node set $i$ from $P_2$ is feasible for graph $H_4$, and its cardinality is $N_s + N_j$. Since these sets of node disjoint paths are feasible (but not necessarily optimal) for $H_3$ and $H_4$, the sum of the cardinalities of maximum node disjoint paths for these graphs must be at least $2N_s + N_i + N_j$.*

*This establishes the submodularity property for the node disjoint path problem under consideration, and by extension for the maximum flow problem.*

Thus, Algorithm 3 is an approximation algorithm with approximation guarantee $1 - \frac{1}{e}$. Additionally, because each round of greedy selection consists of solving a polynomial number of maximum flow problems, and there are $K$ rounds of selection, the running time of Algorithm 3 is polynomial in the number of regular nodes, the number of locations, and the number of mobile backbone nodes.

## 4.3 Empirical Computation Time of Exact and Approximation Algorithms

Two factors impact the complexity of the network flow problem resulting from a given mobile backbone node placement: the mobility of the regular nodes and the value of $\tau_{min}$. The mobility of the regular nodes impacts the number of arcs in the graph: increased regular node mobility increases the number of arcs connecting nodes $1,\ldots,N$ to nodes $N+1,\ldots,N+L$. The value of $\tau_{min}$ impacts the capacity of arcs in the network design graph: decreased $\tau_{min}$ increases the capacity of the arcs incident to the sink, up to a maximum value of $N$.

Computational experiments were conducted to investigate the impact of these factors on the empirical computation time of the exact and approximation algorithms. Figure 4-6 shows the average computation time of the exact (MILP) algorithm on a set of 20 randomly-generated problems, for $L = 20$, $N = 10$, $K = 3$. Regular node mobility and $\tau_{min}$ were varied for each case. Regular node mobility ranges from zero, to a value large enough

such that a regular node in the middle of the area to be explored can reach any point in the area. $\tau_{min}$ varies from a value such that most mobile backbone node locations can only accommodate a single regular node, to very small values that result in many mobile backbone node location being able to handle $N$ regular nodes. (Note that even if many regular nodes can be accommodated, only those that are covered can actually be assigned. That is, even if the capacity of an arc incident to the sink in the network design graph is very high, the actual volume of flow it can receive is limited to the number incoming of arcs incident to its origin node.)

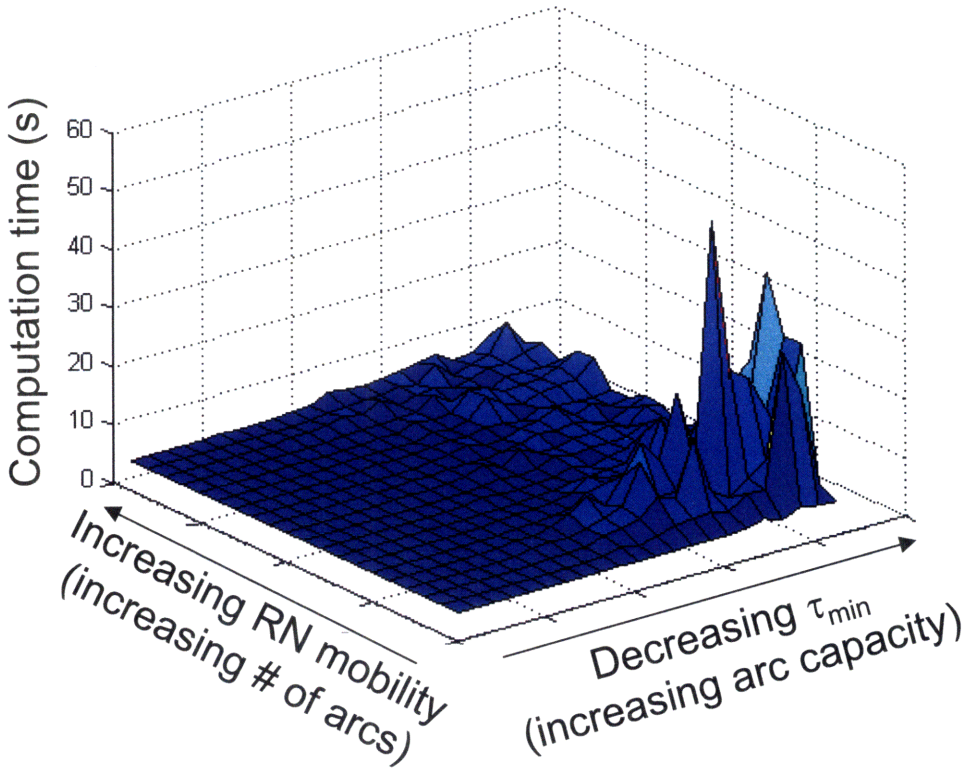



Figure 4-6: Average computation time for various values of regular node mobility radius and $\tau_{min}$.

As Figure 4-6 indicates, average computation time is highest for relatively low regular node mobility and low $\tau_{min}$. As shown in Figure 4-7, the greatest increase in computation time generally occurs when not all regular nodes can be assigned, indicating that increased regular node mobility for a given $\tau_{min}$ results in an "easier" problem.

Figure 4-6 is an average of many problem instances. Closer examination of a single

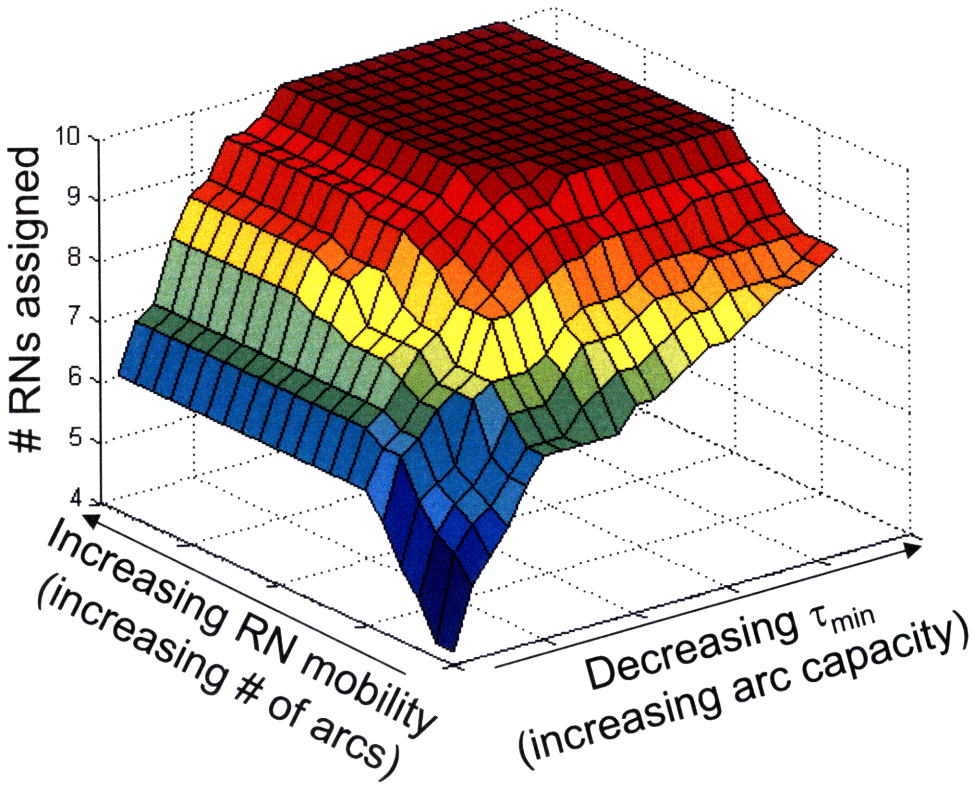



Figure 4-7: Average number of regular nodes assigned for various values of regular node mobility radius and $\tau_{min}$.

problem instance reveals that typically, only a few values of the regular node mobility radius and $\tau_{min}$ result in excessively high computation times, while computation times are quite low for most values of the regular node mobility radius and $\tau_{min}$. Figure 4-8 shows an example of this phenomenon, for a single problem instance. Fortunately, for the cases in which the computation time of the MILP algorithm was excessively high (greater than 1000 seconds), the approximation algorithm was able to assign as many regular nodes as the exact algorithm in less than 10% of the computation time, for all cases considered.

## 4.4 Variably Valuable Sensing Locations

Thus far, the objective function considered in this chapter has reflected uniformly valuable sensing locations. That is, the objective has been to maximize the number of regular nodes that achieve throughput at least $\tau_{min}$, regardless of where these regular nodes are placed.

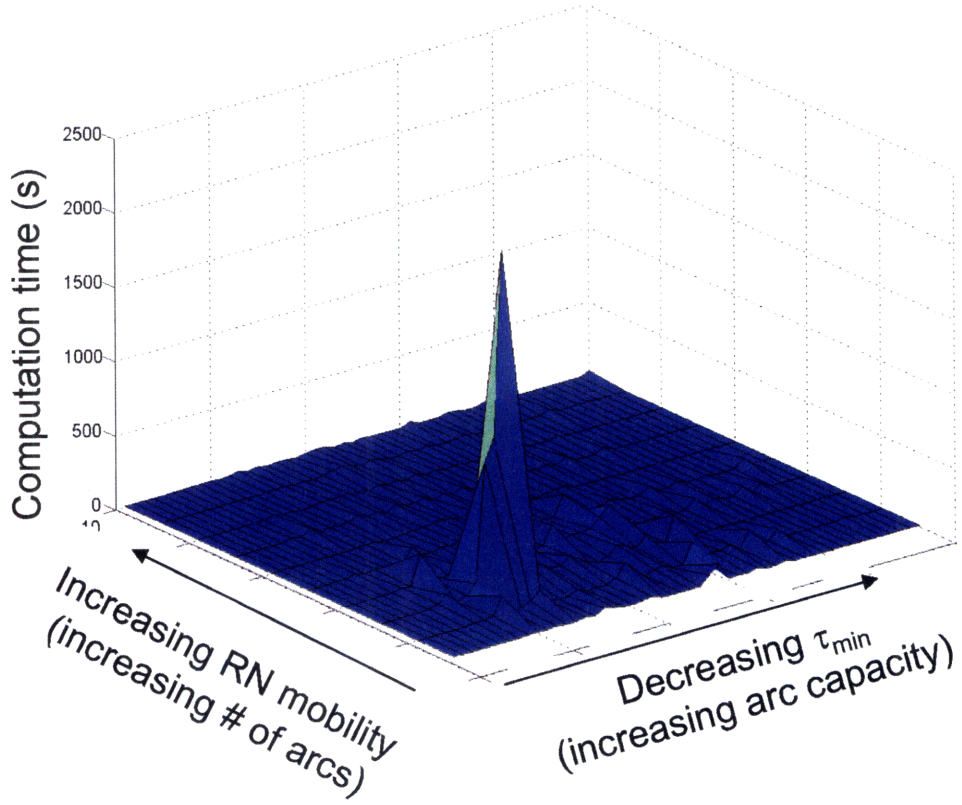



Figure 4-8: Computation time for various values of regular node mobility radius and $\tau_{min}$, for a single problem instance.

However, in many sensing applications, and in the cooperative exploration problem in particular, information generally is not distributed uniformly in the environment. The network design problem can be modified in order to model the non-uniform value of measurements taken from various locations. In particular, the objective is changed from a maximum flow objective to a maximum *cost* flow objective. For example, the graph shown in Figure 4-1 can be modified such that the arc connecting node $N+i$ to node $N+L+i$ has cost $w_i$, reflecting a value of $w_i$ for location $i$. This modified graph is shown in Figure 4-9.

The resulting network design problem is

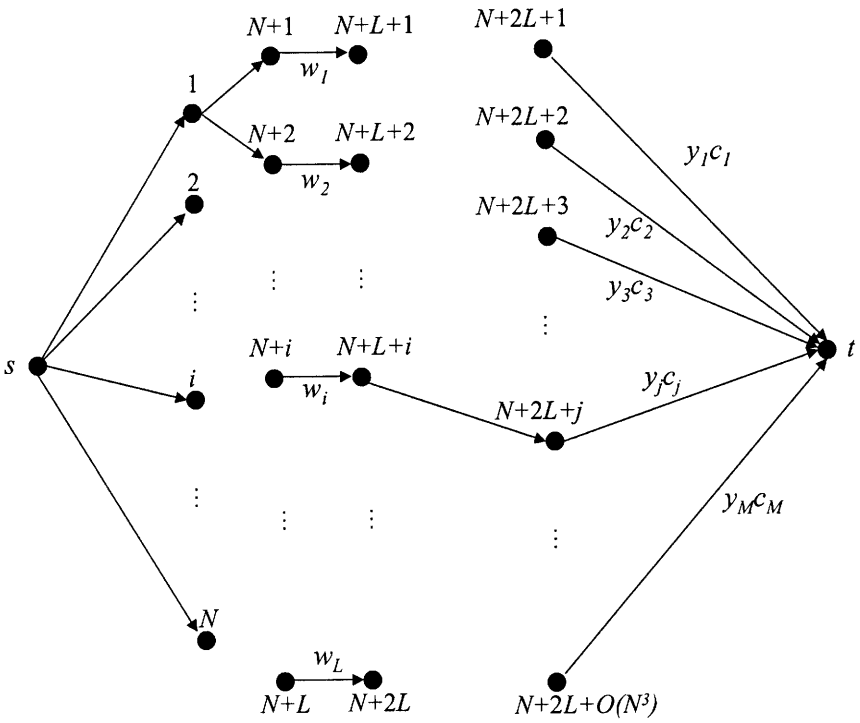



Figure 4-9: A network design graph for the case of variably valuable sensing locations. Weights $w_1, \ldots, w_L$ reflect the values of the sensing locations.

$$\max_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{L} w_i x_{N+i,N+L+i} \tag{4.3a}$$

$$\text{subject to } \sum_{i=1}^{M} y_i \leq K \tag{4.3b}$$

$$\sum_{j:(i,j)\in\mathscr{A}} x_{ij} = \sum_{l:(l,i)\in\mathscr{A}} x_{li} \qquad i \in \mathscr{N} \setminus \{s,t\} \tag{4.3c}$$

$$x_{ij} \geq 0 \qquad \forall\,(i,j) \in \mathscr{A} \tag{4.3d}$$

$$x_{ij} \leq 1 \qquad \forall\,(i,j) \in \mathscr{A} : j \in \mathscr{N} \setminus \{t\} \tag{4.3e}$$

$$x_{(N+2L+i)t} \leq y_i c_i, \qquad i \in \{1,\ldots,M\} \tag{4.3f}$$

$$y_i \in \{0,1\} \qquad i \in \{1,\ldots,M\} \tag{4.3g}$$

where the objective is to maximize the weighted flow $\mathbf{x}$ (Eq.4.3a). The constraints state that at most $K$ arcs (mobile backbone node locations) can be selected (4.3b), flow through all internal nodes must be conserved (4.3c), arc capacities must be observed (4.3d - 4.3f), and $y_i$ is binary for all $i$ (4.3g). Once again, for a given specification of the $\mathbf{y}$ vector, an integer optimal flow $\mathbf{x}$ always exists.

Computational experiments were conducted in which 25 locations were assigned random values in $[0,1]$, and five regular nodes and two mobile backbone nodes were optimally placed and assigned. On average, the algorithm that accounted for these values visited locations that were 57% more valuable than those visited by the algorithm that assigned all locations the same value.

## 4.5 Application to Cooperative Exploration

This section applies the techniques developed in the previous sections to a cooperative exploration problem. Consider a situation in which a set of $L$ locations are to be visited and sensed by regular nodes, and the sensor data taken by the regular nodes is to be transmitted to the mobile backbone nodes. A location is successfully visited at time $t$ if the following conditions are met:

- The location is occupied by a regular node $n_i$ at time $t$.

- Regular node $n_i$ is assigned to a mobile backbone node at time $t$.

Once a location has been visited, it remains visited for all future time. Our goal is to minimize the time required to visit all locations.

This problem can be formulated as a MILP; however, even for small numbers of locations and regular nodes, the problem rapidly becomes computationally intractable. Therefore, we turn our attention to heuristic and approximate algorithms.

First, consider a 1-step lookahead (i.e., greedy in time) algorithm (Algorithm A) based on a slight modification of the MILP technique described in Section 4.1. At each time step, the algorithm positions both regular nodes and mobile backbone nodes in order to maximize the number of unvisited locations that are visited. This is accomplished using the MILP described in Section 4.1. In the case that no regular node is able to reach an unvisited location in a particular iteration, a simple greedy algorithm can become "stuck" and make no further progress because no regular node has any incentive to move. Therefore, if a subset of the regular nodes is unable to reach any unvisited locations, they simply move to the locations that minimize the sum of their distances to the remaining unvisited locations, where distance is calculated as the number of steps that need to be taken in the accessibility graph. This modification guarantees that all locations will be visited in finite time.

For comparison, a second 1-step lookahead algorithm is also considered. This algorithm (Algorithm B) is based on existing techniques that cannot accommodate controlled regular node motion. In this algorithm, regular nodes are greedily positioned on unvisited locations, and mobile backbone nodes are then optimally placed in order to provide communication support for the regular nodes occupying unvisited locations. Again, regular nodes that cannot reach unvisited locations are moved to the locations that minimize the sum of their distances to the remaining unvisited locations.

The key difference between these two algorithms is that Algorithm A optimizes over both the *placement* of regular and mobile backbone nodes as well as the *assignment* of regular nodes simultaneously, while Algorithm B treats regular node placement and assignment sequentially, resulting in degraded performance.

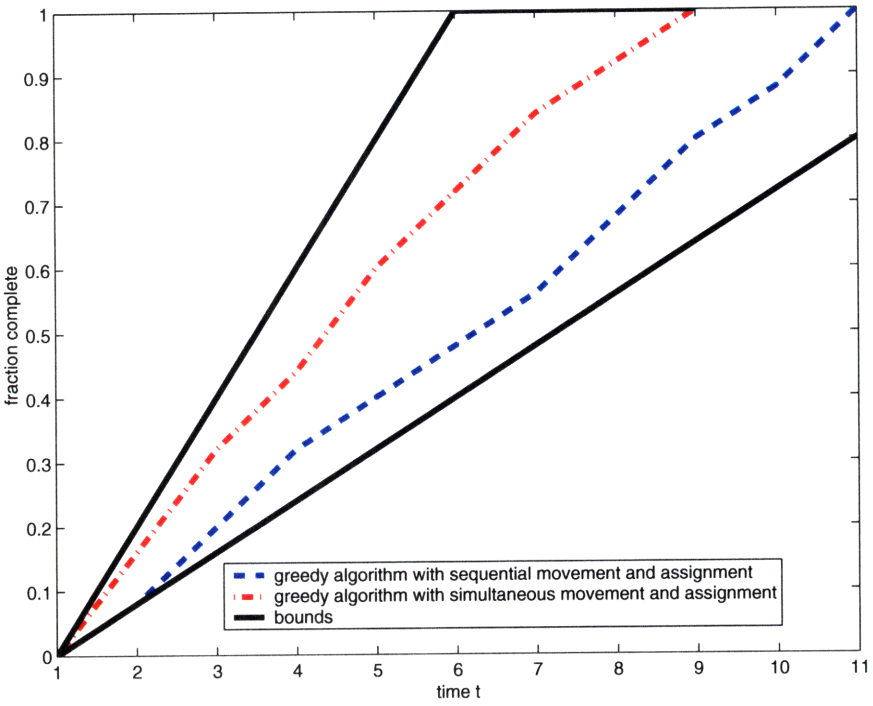



Figure 4-10: Performance of two 1-step lookahead algorithms (Algorithm A and Algorithm B) for the exploration problem, in terms of the fraction of locations visited as a function of time.

### 4.5.1 Empirical Performance Analysis

Figure 4-10 illustrates the typical performance of these 1-step lookahead algorithms on a particular example problem. A set of 25 locations were randomly generated in the plane according to a uniform distribution, and five regular nodes were randomly assigned to initial locations. Two mobile backbone nodes were available to collect data from the regular nodes. The red (dash-dot) line represents the percentage of the locations that were visited as a function of time when Algorithm A was used. The blue (dashed) line represents the same quantity when Algorithm B was used.

The upper black line in Figure 4-10 is a theoretical upper bound on performance: this line depicts the fraction of locations that would be visited if every regular node were successfully placed at an unvisited location and assigned to a mobile backbone node at every time step. In many cases this level of performance is not achievable by any algorithm; this upper bound is considered due to the intractability of solving the problem to optimality.

The lower black line represents the level of performance that would be achieved if every mobile backbone node covered only one regular node on an unvisited location at each time step. This is a lower bound on performance if the regular nodes are unconstrained in their movement (i.e., a regular node can reach any location from any other location in a single time step); otherwise, it is not a bound, but it is an interesting point of comparison by which to judge algorithms.

As shown in Figure 4-10, simultaneous placement and assignment of regular nodes and mobile backbone nodes tends to significantly outperform sequential placement of these nodes in terms of total time required to visit all locations, as well as in the percentage of locations that have been visited at times prior to the completion time.

Figure 4-11 shows the same quantities as Figure 4-10 for a different problem instance, with the addition of the approximate versions of Algorithm A and Algorithm B (shown with dotted lines). As the figure indicates, both approximation algorithms perform quite well compared to their exact counterparts. In this case, the performance of the approximate version of Algorithm B coincides with that of the exact version.

To verify that these trends hold over many problem instances, the performance of the

two 1-step lookahead algorithms was examined for 100 randomly-generated sets of initial conditions. On average, Algorithm A significantly outperformed Algorithm B, both in terms of total time to visit all locations and in terms of the percentage of locations that were visited at any particular time. At the theoretical minimum time at which exploration might have been completed by an optimal algorithm ($t = \lceil \frac{L}{N} \rceil$), Algorithm A had visited an average of 72% of the locations, while Algorithm B had only visited 57% of the locations. An approximate version of Algorithm A in which the MILP optimization was replaced with the polynomial-time approximation algorithm developed in Section 4.2 had visited 67% of the locations.

It is also of interest to examine the time-discounted performance of both algorithms, since information gathered from uncertain environments is generally more useful when it is received earlier rather than later. The average time-discounted reward earned by both 1-step lookahead algorithms was calculated for the randomly-generated instances described in the previous paragraph, where the reward at time $t$ is simply the total number of locations that have been visited at time $t$, discounted by a factor of $\alpha^t$, where $\alpha \leq 1$. Figure 4-12 shows the relative improvement in total discounted reward obtained by the exact and approximate versions of Algorithm A over Algorithm B, evaluated at $t = \lceil \frac{L}{N} \rceil$, for various values of $\alpha$. As the graph indicates, Algorithm A achieved a discounted reward that was $35 - 45\%$ greater than that of Algorithm B for values of $\alpha \geq 0.5$, and the approximate version of Algorithm A achieved a discounted reward $25 - 35\%$ greater than that of Algorithm B.

To examine the time-discounted performance of Algorithm A in the case of variably valuable sensing locations, computational experiments were performed in which locations were assigned random values in $[0, 1]$. Figure 4-13 shows the relative improvement in total time-discounted reward when values of locations are taken into account, as described in Section 4.4. As Figure 4-13 indicates, accounting for locations' values in Algorithm A results in a $15 - 30\%$ improvement in total reward accumulated by time $t = \lceil \frac{L}{N} \rceil$, depending on the value of $\alpha$ used.

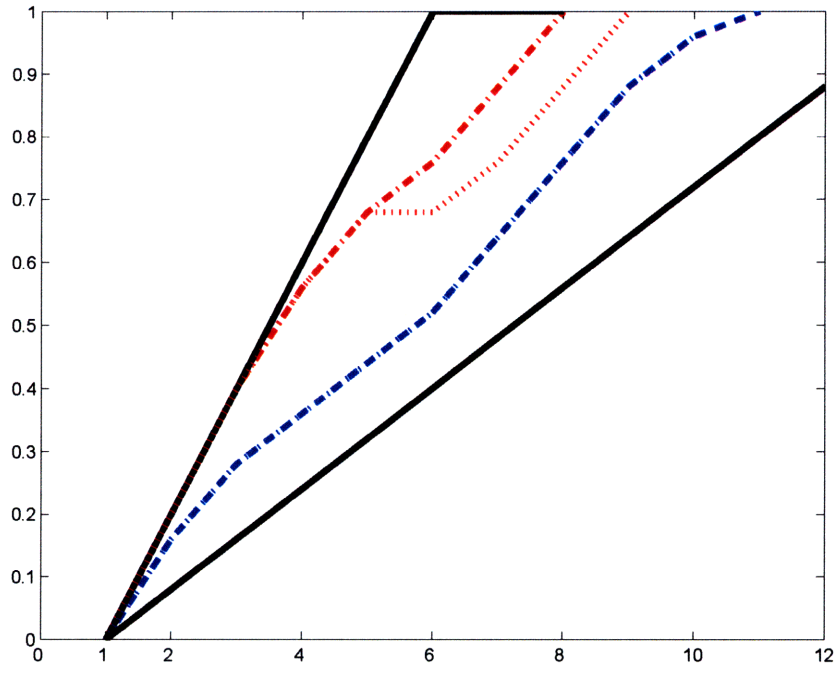



Figure 4-11: Performance of the exact and approximate versions of Algorithm A and Algorithm B, in terms of the fraction of locations visited as a function of time.

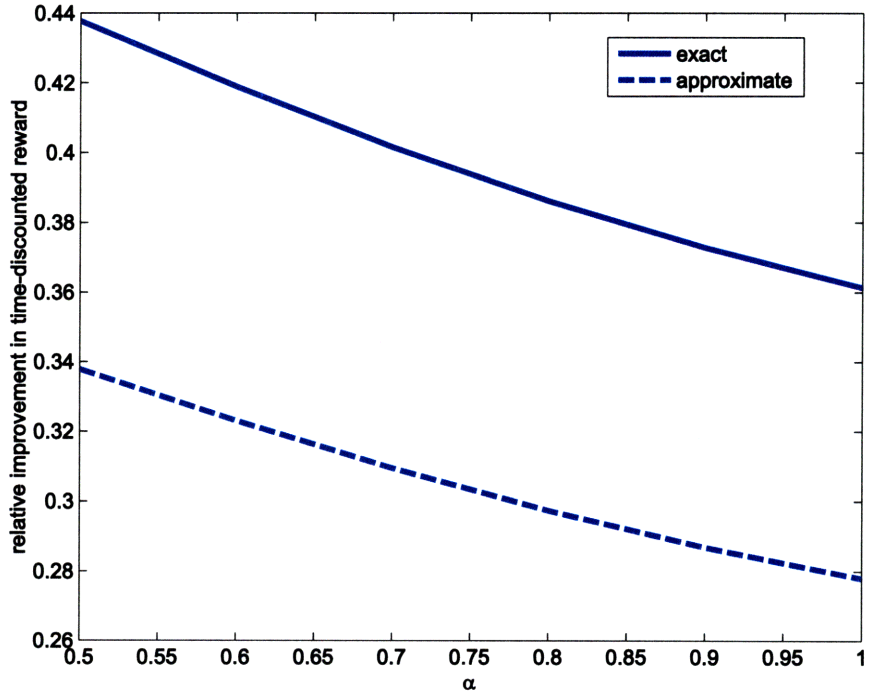



Figure 4-12: Average improvement in time-discounted reward of the exact and approximate versions of Algorithm A, relative to the time-discounted reward of Algorithm B.

### 4.5.2 Theoretical Performance Analysis

As described previously, there are two complicating aspects of the exploration problem under communication constraints. One is the issue of motion planning, which is a difficult problem even when communication constraints are neglected. The other is the impact of communication constraints, as considered in this chapter. To isolate the effect of communication constraints on the efficiency of exploration, we assume for purposes of analysis that the regular nodes are unrestricted in their movement, i.e., a regular node can reach any location from any other location in a single time step. In this case, a trivial upper bound on the time required to visit all locations is $T \leq \lceil \frac{L}{K} \rceil$, but a tighter upper bound can be found.

First, note that the total number of locations visited is a submodular function of the set of configurations of regular nodes and mobile backbone nodes that have been realized, where a configuration of regular nodes and mobile backbone nodes includes both their locations and the assignment of regular nodes to mobile backbone nodes. This is easy to see: if a measurement is taken from configuration $j$, this measurement cannot increase the total number of locations visited by a greater quantity if measurements have already been taken from configurations $S \cup \{i\}$ than if measurements have been taken from configurations $S$, since the measurement taken from configuration $j$ may involve locations that are also measured in configuration $i$.

Using this insight, one can derive a performance bound on the time required to explore all locations using an exact 1-step lookahead approach such as Algorithm A. Let $T^*$ denote the time required to visit all locations using an exact algorithm. Because of the submodularity property, at time $t = T^*$, a greedy algorithm will have visited at least $\lceil (1 - (1 - \frac{1}{T^*})^{T^*})L \rceil \leq \lceil (1 - \frac{1}{e})L \rceil$ locations. Furthermore, at each time $t > T^*$, the greedy algorithm can visit at least $K$ new locations (assuming that $K$ locations remain to be visited). So, the time required to visit the remaining locations is at most

$$\begin{aligned} \left\lceil \frac{L - \lceil (1 - \frac{1}{e})L \rceil}{K} \right\rceil &= \left\lceil \frac{L + \lfloor (\frac{1}{e} - 1)L \rfloor}{K} \right\rceil \\ &= \left\lceil \frac{\lfloor \frac{L}{e} \rfloor}{K} \right\rceil . \end{aligned}$$

This yields an overall bound for the time $T$ required to visit all locations of

$$T \leq \min\left\{\left\lceil\frac{L}{K}\right\rceil, T^* + \left\lceil\frac{\left\lfloor\frac{L}{e}\right\rfloor}{K}\right\rceil\right\},$$

which means that although even an exact algorithm may take up to $\lceil\frac{L}{K}\rceil$ time steps to completely explore all locations, a greedy algorithm is guaranteed to take no longer than $\left\lceil\frac{\lfloor\frac{L}{e}\rfloor}{K}\right\rceil$ more time steps than an exact algorithm, up to a maximum of $\lceil\frac{L}{K}\rceil$ total time steps.

This bound could be used, for instance, when an approximate solution to the problem has been obtained and it is desirable to determine the amount by which this solution might be improved by expending additional computational effort to increase the horizon of the limited lookahead algorithm.

## 4.6 Integration into an Exploration Algorithm

Section 4.5 developed algorithms for visiting pre-defined sensing locations. This section demonstrates how the algorithms developed in Section 4.5 can be combined with existing algorithms to create an overall architecture for exploring a given area using a given sensor footprint in a communication-sensitive manner.

### 4.6.1 Measurement Location Selection

The primary goal of the algorithms described in this section is to cover an area of interest as efficiently as possible. A related problem is that of minimizing the number of (stationary) sensors placed such that an entire area is covered. The difference between these two problems is that in the context of exploration (dynamic coverage), measurements are taken sequentially rather than simultaneously. However, algorithms designed to minimize the number of stationary sensors deployed also serve the purpose of minimizing the number of measurements that need to be taken by mobile sensors.

In addition to coverage of the area of interest, accessibility of the sensing locations by mobility-constrained regular nodes is also a concern in the exploration problem. In the framework described in this chapter, regular nodes can move from one location to another in

a single time step if these locations are within the radius of movement $r$ of the regular nodes. Thus, each location must be within a distance of $r$ of at least one other location in order to be accessible. Fortunately, algorithms that guarantee connected coverage of stationary sensor networks (in which communication connectivity rather than physical separation is the primary concern [24]) can easily accommodate this constraint.

Although a number of algorithms for sensor placement exist, this section utilizes a coverage algorithm proposed by Kar and Banerjee [24]. This algorithm was selected because it is designed to create a connected network, and because it is a computationally efficient algorithm that carries a theoretical performance guarantee. This performance guarantee bounds the ratio of the density of sensors placed by this algorithm to the density of sensors in an optimal solution. In particular, this ratio is at most $2.693(1 + \frac{2.243Lr}{A})$, where $L$ is the diameter of the region to be covered, $r$ is the sensing and movement radii of the regular nodes (for simplicity, these radii are assumed to be equal, although this is not necessary), and $A$ is the area of the region to be covered, where $A \geq \pi r^2$ [24].

The algorithm described by Kar and Banerjee works as follows: for an infinite plane, a near-optimal configuration of nodes is derived. The density of nodes in the approximate solution is within 3% of the optimal density. This pattern consists simply of strips of sensor nodes, tiled such that they form a dense packing, with one additional strip placed to guarantee connectivity. In the case in which the area to be covered is finite (rather than an infinite plane), the nodes that intersect this finite area are selected, and final strip is simply placed such that it intersects all other strips used. Figure 4.6.1 shows examples of these tilings.

The algorithm described by Kar and Banerjee assumes a convex sensing region. Since the regions of interest in practice can be nonconvex, it is necessary to first decompose the region into a set of convex subregions. This problem is referred to as *convex decomposition.*

### 4.6.2 Convex Decomposition

Given a nonconvex polygon, a convex decomposition algorithm divides the polygon into convex subregions. Many such algorithms exist; for example Keil [25] uses dynamic pro-

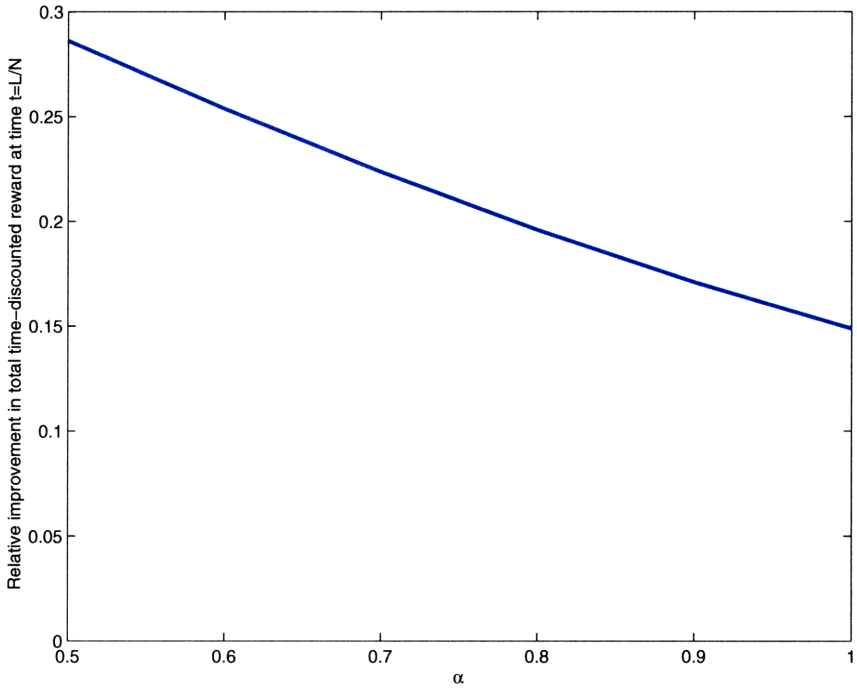



Figure 4-13: Relative improvement in total time-discounted reward when values of locations are taken into account.

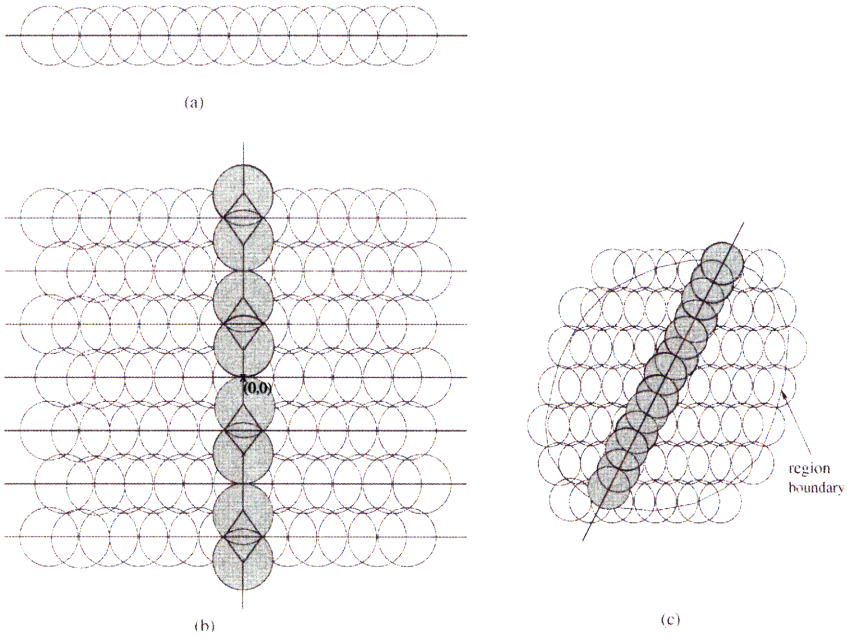



Figure 4-14: Sensor tilings for connected coverage. (a) A single strip of sensors. (b) A tiling of sensors for an infinite plane. (c) A tiling of sensors for a finite region. Figure is taken from Ref. [24].

gramming techniques, modified to improve their efficiency, to decompose polygons into a minimal number of convex subregions.

In general, the region of interest will not be polygonal but can instead contain obstacles or regions that do not need to be sensed. However, heuristic techniques exist to divide arbitrary regions into simple polygons.

### 4.6.3 Example

To demonstrate the use of these algorithms in the context of cooperative exploration, a small example is provided. The area shown in Figure 4-15 is given, with obstacles shown as red rectangles. This area is then subdivided into convex polygons using a heuristic technique applicable to rectangular regions containing rectangular obstacles: vertical boundaries extend from each obstacle's vertices, thus dividing the area into finitely many rectangular regions. Next, each of these regions is covered by sensors using the algorithm of Kar and Banerjee, as shown in Figure 4-16. Figure 4-16 shows these sensors with their sensing radii; for clarity, the sensing locations are shown without radii in Figure 4-17. Finally, Figure 4-18 shows the connectivity of these locations with regard to regular node movement: a line connects two locations if a regular node can reach one location from another in a single time step.

Once a problem has been preprocessed in this manner, it can be used as input to the algorithm described in Section 4.5.

## 4.7 Summary

This chapter presented a generalization of existing mobile backbone network problems that models the motion of both controlled regular nodes and mobile backbone nodes. A MILP-based exact solution to this problem was shown to perform well in practice for problems of moderate size, and a polynomial-time approximation algorithm was given for larger problems. The performance of a 1-step lookahead algorithm based these techniques was given for a cooperative exploration problem in which sensing locations are given *a priori*, and a performance bound was established for a special case of this exploration problem. Finally,

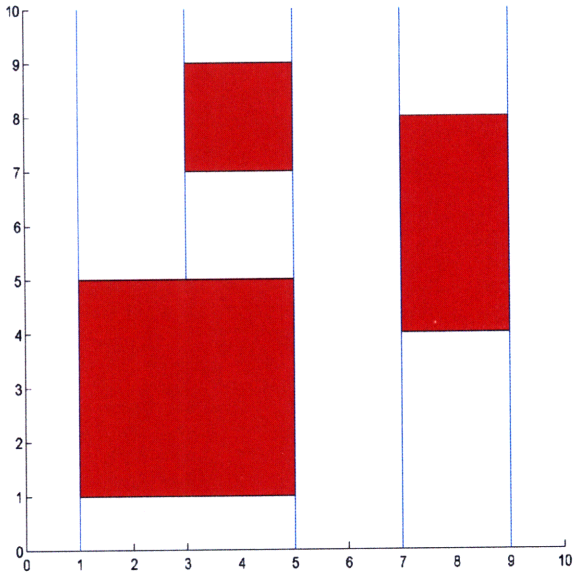



Figure 4-15: An area to be explored. Obstacles are indicated by red rectangles. The area to be explored is divided into convex polygonal subregions.

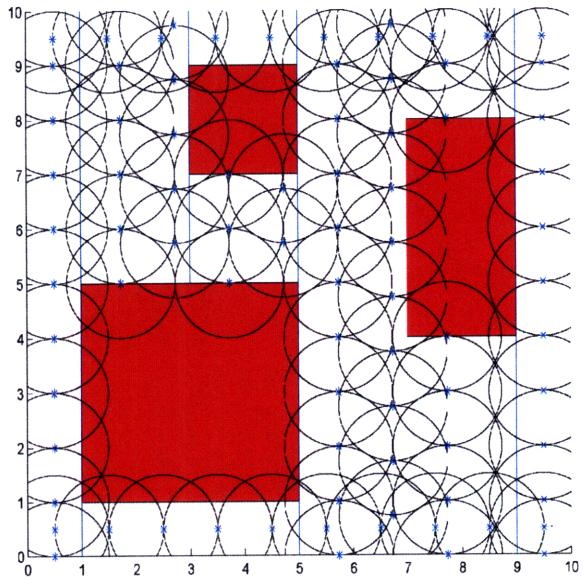



Figure 4-16: Each subregion is covered by sensing locations, and sensing locations are placed to connect the subregions. Sensing radii are indicated by circles.

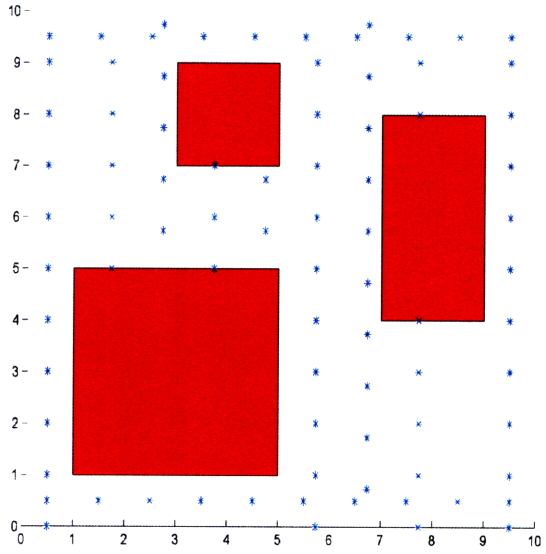



Figure 4-17: Sensing radii have been removed for clarity.

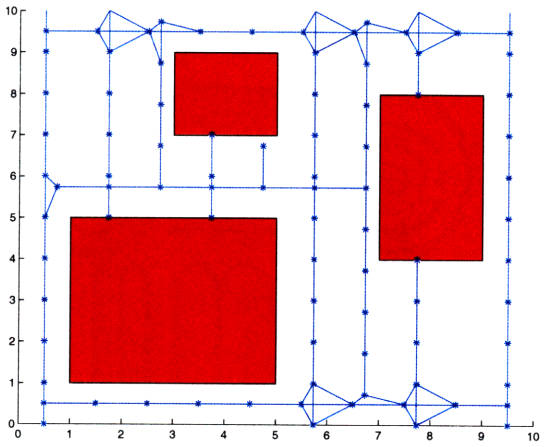



Figure 4-18: Lines connect locations that are accessible by regular nodes in a single time step.

the algorithms developed in this chapter were integrated into an exploration technique that takes a given area to be explored, divides it into convex subregions, and places sensing locations within these subregions such that the entire area is covered, and all sensing locations are reachable by regular nodes.

# Chapter 5

# Transmission Subproblem

Chapters 3 and 4 of this thesis focused on network optimization while taking an abstract view of the actual information being transmitted. In contrast, this chapter examines the problem of optimally *utilizing* available communication capacity. In the context of cooperative estimation, this chapter develops algorithms for selecting pieces of information for transmission among agents.

## 5.1 Model

Exploration is, in essence, an information-gathering process. If a probabilistic representation of the map estimate is assumed [56], the quality of a map estimate can be expressed as its Shannon entropy, or the degree to which the probability distribution of feature location estimates is compact or spread out. In the case of a Gaussian distribution, the entropy of the distribution is also related to the mean squared error of the estimate, a quantity of importance in control applications. Therefore, entropy is the quality metric used for this work.

The centralized version of the mapping problem can be expressed as a minimization of

entropy, subject to an energy constraint:

$$\min_{m_{it}, s_{it}, c_{it}} \quad U_T(m_{it}, s_{it}, c_{it}) \tag{5.1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \sum_{t=0}^{T} E(m_{it}, s_{it}, c_{it}) \leq E_{max} \tag{5.2}$$

where $m_{it}$ denotes agent $i$'s motion at time $t$, $s_{it}$ denotes agent $i$'s sensor measurements at time $t$, $c_{it}$ denotes agent $i$'s communication to other agents or to a central repository at time $t$, $U_T$ denotes entropy at a central repository at the final time $T$, and $E$ denotes energy. Note that although total energy is minimized in this formulation, other formulations are also of interest, including minimization of the maximum energy expenditure by any agent. An alternative problem formulation involves minimizing energy expenditure subject to an entropy constraint:

$$\min_{m_{it}, s_{it}, c_{it}} \quad \sum_{i=1}^{n} \sum_{t=0}^{T} E(m_{it}, s_{it}, c_{it}) \tag{5.3}$$

$$\text{subject to} \quad U_T(m_{it}, s_{it}, c_{it}) \leq U_{max} \tag{5.4}$$

It is desirable that a solution to this problem scale well with the number of map features, the spatial dimension of the map, and the number of agents performing the exploration. However, it rapidly becomes intractable to solve this problem to optimality for large numbers of agents, and communication limitations make it impractical for a central decision-maker to maintain adequate communication with all agents across large distances. In order to ensure scalability it is therefore appropriate to consider heuristic or approximation algorithms for decision-making.

## 5.2 The Static Problem

A first step toward developing a sequential decision-making algorithm is understanding the implications of a single decision. In the case of deciding the amount of energy to expend making communication transmissions in the exploration problem, one must know (or have an estimate of) the entropy reduction that can be realized for a given amount of information

transmitted before deciding whether it is worthwhile to transmit this amount of data. This prompts the question: what is the maximum amount of information that can be gained by transmitting a quantity of data using throughput not more than $B$?

To answer this question, this chapter examines the inter-agent communication problem in a static sense. It is a challenging problem to decide, for a single pair of agents in a single time step, what information should be communicated between them in order to minimize the entropy of the agents' probability distributions after incorporating the communicated information.

In the problem under consideration in this chapter, two agents jointly estimate the location of a number of stationary features. Without loss of generality, one agent is designated as the sender and the other the receiver. Both agents represent their feature estimates as Gaussian distributions, in the information (inverse covariance) form. It is assumed that the sender has a perfect estimate of the receiver's information matrix, and the goal is simply to choose the most beneficial elements of the receiver's own information matrix to send to the receiver. The utility of information is evaluated in terms of the reduction in entropy of the estimate, which in the Gaussian case is represented by

$$
\begin{aligned}
U &= \frac{1}{2}\log[(2\pi e)^n|\det(P)|] = \frac{1}{2}[\log((2\pi e)^n) + \log(|\det(P)|)] && (5.5)\\
&= \frac{1}{2}[\log((2\pi e)^n) + \log(|\det(Y^{-1})|)] = \frac{1}{2}[\log((2\pi e)^n) + \log(\frac{1}{|\det(Y)|})] && (5.6)\\
&= \frac{1}{2}[\log((2\pi e)^n) - \log(|\det(Y)|)] && (5.7)
\end{aligned}
$$

where $P$ is the receiver's covariance matrix, and $Y$ is the information matrix, after the receiver has incorporated the information transmitted by the sender.

It is assumed that it is not possible for the agents to exchange an arbitrarily large amount of information. In particular, the sender has a limit on the amount of information that can be sent, denoted by $B$. For simplicity of modeling, assume that each element of the information matrix requires a single unit of throughput to send. There is also a certain amount of communication overhead required to provide the receiver with labels describing which diagonal elements are being sent, and this cost is modeled as one unit of throughput per

diagonal element. (It is assumed and both agents are estimating the location of a common set of features, and that they share a system for labeling them.) Note that for a given set of diagonal elements to be communicated, the sender must choose whether to send information about off-diagonals, compounding the difficulty of the problem. Given the entropy metric and the throughput cost model, if the covariance intersection algorithm is used to combine the receiver's information matrix with the transmitted submatrices, the sender's problem is

$$
\begin{aligned}
\text{minimize} \quad & c^T x + \log(\det(Y(x)^{-1})) \\
\text{subject to} \quad & Y(x) > 0 \\
& Y(x) = \sum_{i=0}^{M} x_i y_i Y_i \\
& \sum_{i=1}^{M} y_i B_i \leq B \\
& y_i \in \{0,1\} \quad i = 1,...,M \\
& y_0 = 1 \\
& 0 \leq x_i \leq 1 \quad i = 0,...,M \\
& \sum_{i=0}^{M} x_i = 1
\end{aligned}
\tag{5.8}
$$

where $Y_0$ is the receiver's initial information matrix; $Y_i$ is the $i^{th}$ candidate submatrix, extracted from the original information matrix as described in [38] and in Chapter 2; $y_i$ is a binary decision variable indicating that the $i^{th}$ submatrix is selected for transmission; $x_i$ is the weighting coefficient used in the covariance intersection algorithm; $B_i$ is the throughput requirement of transmitting submatrix $Y_i$; $N$ is the number of map features; and $M$ is the number of submatrices of the information matrix.

This problem rapidly becomes difficult as the number of features to be mapped increases. For an $N \times N$ information matrix, the number of possible combinations of diagonal elements that could be sent is $2^N$. Given a selection of diagonal elements, there is also a decision to be made regarding which cross-correlation terms should be sent. Thus, the optimal set of data to transmit cannot be found through brute force search. A more intelligent

strategy is needed, and we turn our attention to this topic.

### 5.2.1 Block-Diagonal Case

We first consider the case of a natural measurement scenario that results in a structured information matrix that is not fully populated. Assume that a mobile robot, the location of which is adequately known (through GPS or other means), is taking measurements of feature locations using a noisy sensor. In this example, bearing-only measurements of feature locations are taken by means of a camera whose exact angle relative to the robot is not known, but is only estimated.

If the locations of the features are being estimated in a two-dimensional plane, the $x$ and $y$ locations of each feature are correlated through the uncertainty in the camera angle when measurements of the feature are taken. Because the robot's location is known, however, there is no correlation from feature to feature. Thus, the information matrix is block-diagonal with $2 \times 2$ blocks:

$$Y = \begin{bmatrix} a_1 & c_1 & 0 & 0 & \\ c_1 & b_1 & 0 & 0 & \\ 0 & 0 & a_2 & c_2 & \cdots \\ 0 & 0 & c_2 & b_2 & \\ & & \vdots & & \ddots \end{bmatrix} \tag{5.9}$$

Recall that the term in the entropy equation that can be altered is $\log(\det(P))$, which is to be minimized, or equivalently $\log(\det(Y))$, which is to be maximized. Because there are exponentially many possible values for $Y$, it is typically not feasible to solve this problem optimally, nor is there an obvious approximation algorithm for this problem. For the case of a block diagonal matrix, however, the objective decomposes into a sum of terms:

$$\log(\det(Y)) = \log\left(\begin{vmatrix} a_1 & c_1 \\ c_1 & b_1 \end{vmatrix}\right) + \log\left(\begin{vmatrix} a_2 & c_2 \\ c_2 & b_2 \end{vmatrix}\right) + \ldots + \log\left(\begin{vmatrix} a_m & c_m \\ c_m & b_m \end{vmatrix}\right) \tag{5.10}$$

For each block in this matrix, there are five possible decisions that can be made by the

sender. The sender can transmit:

1. Information about $a$ only,
2. Information about $b$ only,
3. Information about $a$ and $b$ with no off-diagonal information,
4. Information about $a$ and $b$ coordinates with cross correlation information, or
5. Nothing at all.

Under our communication model, the possible transmissions would require throughputs of 2, 2, 4, 5, and 0 units, respectively, and they would result in estimated reductions in entropy that are computable given an estimate of the receiver's information matrix. A brute force approach to solving this problem would still involve searching over $O(5^m)$ combinations of transmissions. However, because the objective function is now additive, the problem takes on the form of a multiple-choice knapsack problem.

The multiple-choice knapsack problem is a variation of the traditional knapsack problem in which each item belongs to one of several disjoint classes, and the goal is to choose exactly one item from each class such that the total profit of these items is maximized. This problem is *NP*-hard, but there exist fully polynomial-time approximation schemes (FPTASs) for solving it [26]. In the case of the map communication problem there is an item class for each feature, and the items correspond to specific decisions that may be made by the sender. The item weights are represented by the throughput required to send the information (including zero if no information is communicated about a given feature), and the values of the items are represented by the expected reduction in the receiver's entropy resulting from their communication.

Using a dynamic programming-based FPTAS for this problem based on that of Lawler [32], a performance guarantee of $(1-\varepsilon)$ can be achieved with a running time of $O(\frac{ml}{\varepsilon})$, where $m$ is the number of item classes and $l$ is the number of items per class. That is, if the value of the optimal solution to this problem is $E^*$ and the value of the solution generated by our algorithm is $\tilde{E}$, then it is guaranteed to be the case that

$$(1-\varepsilon)E^* \leq \tilde{E} \leq E^* \tag{5.11}$$

if the profits involved are integers, with a straightforward modification necessary in the case of non-integer profits.

In order to evaluate the performance of this algorithm, it is compared to a simple greedy strategy. In the greedy strategy, it is assumed that only complete feature submatrices will be transmitted, i.e. $2 \times 2$ blocks that lie along the diagonal of the sender's information matrix. In order to choose blocks to send, the difference in magnitude between the diagonal elements of the sender's information matrix and the receiver's information matrix is used as a ranking heuristic, and the block with the greatest total difference is ranked highest.

The performance of these two algorithms for a fixed communication throughput ($B = 25$) and various numbers of features $m$, averaged over 20 randomly generated simulation runs per data point, is shown in Figure 5-1. In these simulations, the locations of $m$ features are randomly generated in the plane, and two agents take a predetermined series of measurements of them. These measurements have a known uncertainty associated with them, which allows the computation of an information matrix for each agent.

The running time of the heuristic greedy algorithm is less than the approximation algorithm, and for small numbers of features, the greedy heuristic compares fairly well in terms of performance. (Note that for $B = 25$, all data may be sent for very small numbers of features presented here.) However, as the number of features increases and a trivial solution is no longer available, the approximation algorithm surpasses the greedy algorithm in terms of performance, while the difference in computation time between them remains small. Of particular note is the variability in solution quality from trial to trial; the small difference in solution quality for the approximation algorithm contrasts with the high level of variability in solution quality for the heuristic.

Figure 5-2 compares the performance characteristics of the two algorithms for a fixed number of features (20) and an increasing availability of communication throughput. In this case, the approximation algorithm makes better use of increasing availability of throughput, with little increase in computation time. Again, the approximation algorithm exhibits great consistency in solution quality due to its performance guarantee, while the quality of the heuristic solution is highly variable.

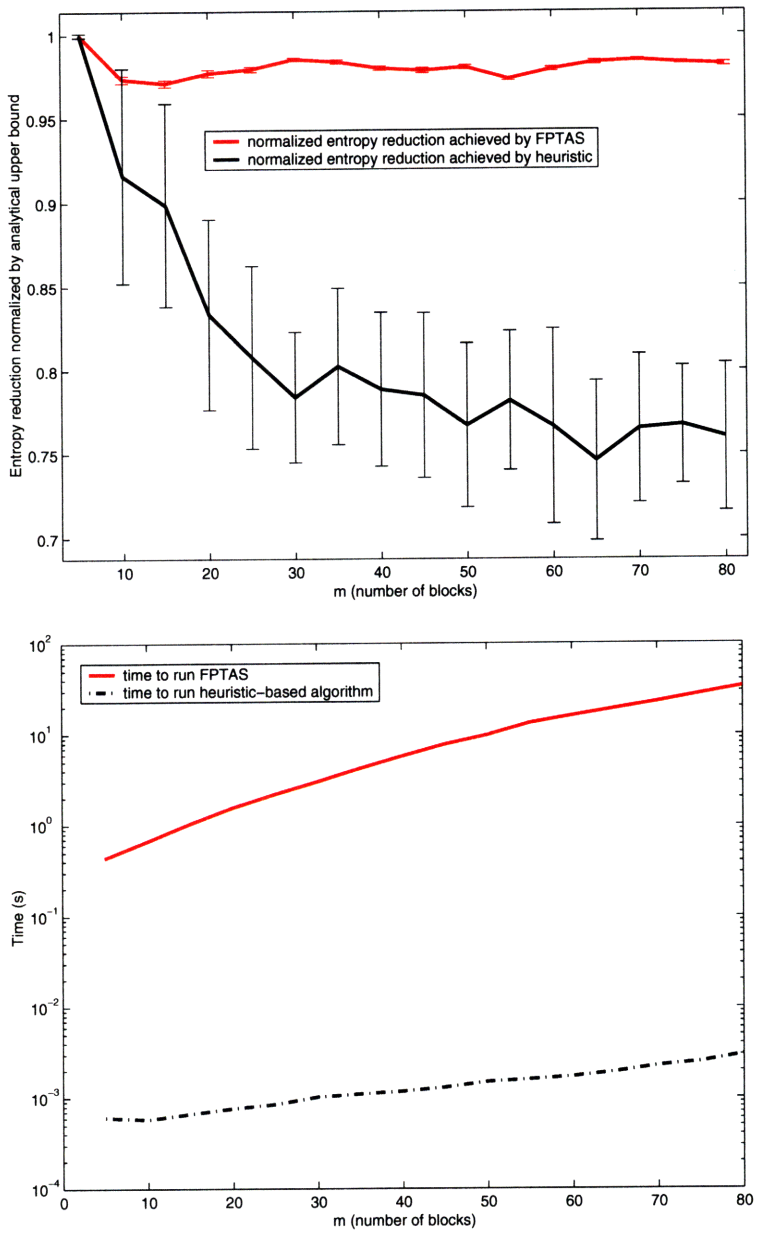



Figure 5-1: The performance of a simple greedy algorithm is compared to that of the fully polynomial time approximation algorithm developed in this section for various numbers of features and a fixed communication throughput of 25 units. The top figure depicts the reduction in the entropy of the receiver's estimate achieved by each algorithm, normalized by the maximum possible reduction in entropy as given by the performance bound of the approximation algorithm. Each data point represents an average of 20 randomly-generated scenarios, and error bars depict the standard deviation of performance over these trials. The bottom figure depicts the average time required to run both algorithms.

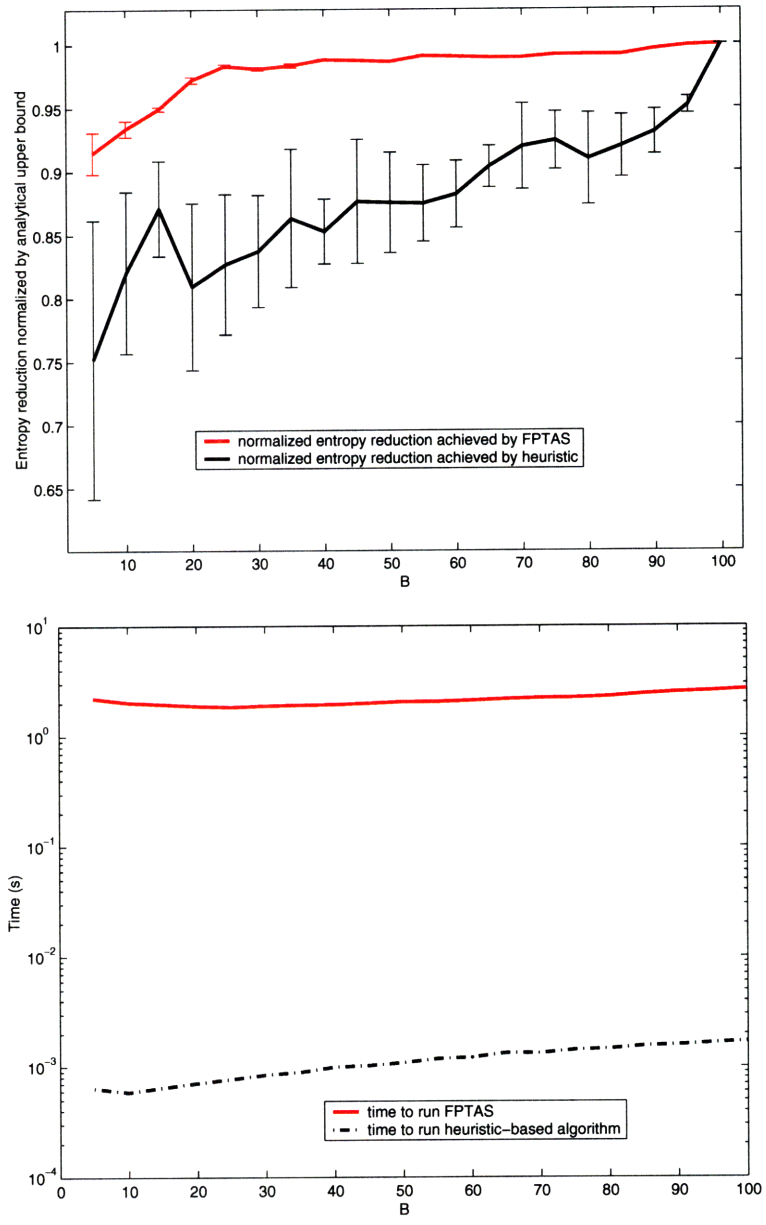



Figure 5-2: The performance of the benchmark greedy algorithm is again compared to that of the fully polynomial time approximation algorithm, this time for varying $B$ and and a fixed number of features, $m = 20$. The top figure shows the reduction in the entropy of the receiver's estimate achieved by each algorithm, normalized by the maximum possible reduction in entropy as given by the performance bound of the approximation algorithm. Each data point represents an average of 20 randomly-generated scenarios, and error bars depict the standard deviation of performance over these trials. The bottom figure depicts the average time required to run both algorithms.

### 5.2.2 Fully-Populated Case

The measurement scheme described in the previous section assumed perfect knowledge of vehicle location in order to arrive at a block diagonal form for the information matrix. In a more general scenario, the vehicle location is unknown and must be estimated along with feature locations using the same noisy sensor measurements. This simultaneous localization and mapping (SLAM) problem naturally leads to a fully-populated information matrix [61], although many terms are often small in the normalized information matrix. There exist sparsification techniques that seek to take advantage of the small magnitude of these values in the normalized information matrix to reach an approximate matrix which is sparse. However, some of these sparsification techniques produce solutions that overestimate confidence in feature locations [61]. Other sparsification techniques work by managing the number of terms of cross correlation and do not explicitly manage the size of blocks in the information matrix. Note that in the previous section, the total number of items in the multidimensional knapsack problem was $5m$ because for each 2x2 block, only five possible choices could be made. If the block size were to increase, however, the number of choices that could be made would grow exponentially with the dimension of the block. Thus, the solution technique applied in the previous section does not scale well as the size of the blocks in the information matrix grows.

The case of a fully-populated information matrix was examined in [38], in which a single large block from the information matrix containing all diagonal and off-diagonal information is transmitted. The features selected for transmission are those about which the sender has learned the most since the previous transmission. However, because of the constraint that only a single large submatrix is sent, a great deal of throughput is used to transmit relatively unimportant cross correlation terms in many cases [61].

Two improvements to the approach given in [38] are proposed in this work. First, rather than evaluate the benefit of sending information by what the sender has learned the most about since the previous transmission, this work utilizes an estimate of the receiver's information matrix to evaluate the value of information. It should be noted that such an estimate is not always available, but when available it should be used to the maximum

possible benefit. Second, rather than selecting a single large submatrix for transmission, one could instead select multiple smaller submatrices. Allowing this option causes the time complexity of the problem to increase greatly, but it also allows a potentially more intelligent use of scarce communication throughput. Thus, even an approximate solution to this more difficult problem may lead to improved performance over the existing simple technique.

Because of the difficulty of solving the more flexible version of the problem, both heuristics and a relaxation approach are utilized in the submatrix-selection algorithm presented in this chapter. In order to solve a relaxed version of the problem, work done initially by Vandenberghe et al. [60] is leveraged. If the sender's problem as stated in Section 5.2 is relaxed so that the throughput constraint and binary decision variables are eliminated, the formulation becomes

$$\begin{aligned} \text{minimize} \quad & c^T x + \log(\det(Y(x)^{-1})) \\ \text{subject to} \quad & Y(x) > 0 \\ & Y(x) = \sum_{i=0}^{M} x_i Y_i \\ & 0 \leq x_i \leq 1 \quad i = 0, \ldots, M \\ & \sum_{i=0}^{M} x_i = 1 \end{aligned} \tag{5.12}$$

This formulation, an extension of the semidefinite programming problem, admits an efficient solution to the relaxed version of the problem under consideration [23]. Note, however, that this formulation includes a decision variable for every possible submatrix of the information matrix. This is an intractably large number for even moderate numbers of map features. Thus, some technique must be found for identifying promising candidate submatrices for consideration in the relaxed algorithm.

Recall that the algorithm in [38] selects a single large submatrix for transmission. We also allow large submatrices to make up some of the transmission candidates. To allow flexibility in the size of candidate submatrices, however, candidates of other sizes are also selected. In this work, large submatrix pairs are also considered in which one submatrix

takes up approximately two thirds of the available throughput and the other takes up one third, as well as submatrix pairs in which each submatrix takes up approximately half of the available throughput.

Selection of these candidate submatrices is accomplished using a stochastic selection heuristic. Candidate submatrices are generated feature by feature in a probabilistic fashion. The first feature is selected according to a probability distribution determined by the magnitudes of the differences between the diagonal elements of the sender's and receiver's information matrices. The next feature is selected according to a distribution based on the magnitude of the cross correlation between the unselected diagonal elements and the previously selected element. Subsequent features are selected in the same fashion, according to a distribution based on the sum of the off-diagonal terms between selected and unselected terms. This process is repeated until the desired number of features for the candidate have been selected. The rationale behind this selection heuristic is that it favors the selection of submatrices containing highly correlated features, which increases the informativeness of the transmitted submatrix.

Transmitting any of these large submatrices (or submatrix pairs) utilizes most of the available throughput for cases of interest. However, when significant throughput is left over, we use the same stochastic selection heuristic to select multiple smaller candidate submatrices of appropriate size.

Following selection of candidate submatrices, the relaxed problem described above is solved to decide among these candidates. For each set of large submatrices, the relaxed problem is formulated with multiple small submatrices. The small submatrices with the highest weightings in the solution to the relaxed problem are selected for inclusion in a second instance of the relaxed problem, this time subject to the throughput constraint. In this second problem, the optimal weightings for the covariance intersection algorithm are found, and a post-transmission entropy is calculated for this set of submatrices. This process is repeated for all large submatrices selected by the stochastic selection heuristic, and the set of large and small submatrices with the lowest post-transmission entropy is selected for transmission.

To summarize, the algorithm proposed for the case of a fully-populated information

matrix consists of the following steps:

1. A set of large candidate submatrices is selected using a heuristic method; in this chapter, a stochastic metric is considered that is based on the magnitude of differences in the sender's and receiver's information matrices.
2. For each large candidate submatrix (or pair of submatrices), multiple small candidates are generated according to the same heuristic.
3. For each large candidate submatrix (or pair of submatrices) and its associated small submatrices, a relaxed version of the transmission problem is formulated, and this problem is solved with no throughput constraint.
4. The small submatrices that received the highest weighting in the relaxed problem are selected to accompany the large submatrix, and the problem is re-solved to find the optimal weights for covariance intersection. Entropy for this set of submatrices is calculated. The submatrices with the lowest post-transmission entropy are ultimately selected for transmission.

Figures 5-3 and 5-4 depict the performance of this algorithm. As in Section 5.2.1, the results of a number of randomly-generated scenarios are shown. In each scenario, sequences of measurements of $N$ features are taken by the sender and the receiver. The sender then uses the algorithm described above to select information for transmission to the receiver. Additionally, a slightly modified version of the algorithm developed in [38] is used as a benchmark. As described above, the features selected in [38] are those about which the sender has accumulated the most information since the previous transmission. Because our formulation considers the results of a single transmission, features in the modified benchmark algorithm are selected based on the magnitude of the difference in information between the sender's and the receiver's information matrices, rather than between the sender's current information matrix and the sender's previous information matrix at the time of the last transmission. This selection process is similar to that of [38] in that a single large submatrix is selected.

The metric by which the algorithms are compared is the reduction in the entropy of the receiver's estimate after incorporating the transmitted submatrices, normalized by the en-

tropy reduction achieved by transmitting submatrices selected through an exhaustive search over all large submatrices. That is, a value of 0.9 in these figures indicates that the algorithm in question resulted in a reduction in the receiver's entropy that was 90% as large as that achieved through exhaustive search; this result would be superior to a value of 0.8, for example.

In Figure 5-3, the amount of throughput available is held constant at $B = 15$ units while the number of map features varies. Note that the algorithm described in this chapter achieves reductions in entropy that are very close to those achieved through an exhaustive search, even when no small submatrices are considered. When small submatrices are considered, the algorithm outperforms the exhaustive search over large submatrices by a significant margin. (An exhaustive search including small submatrices was not performed due to the excessive time that would be required.) Although the performance of this algorithm is comparable to that of an exhaustive search, its computation time is dramatically reduced and scales well with problem size.

Figure 5-4 shows the algorithm's performance for varying availability of communication throughput and a fixed number of features ($N = 10$). Again, performance is good, and computation time increases gracefully with problem size.

Figure 5-5 compares the performance of the algorithm presented in this chapter to the performance of the benchmark algorithm based on that in [38] for maps of realistic dimension ($N$ ranging from 10 to 100). These trials are too large to reasonably include an exhaustive search, but the relative performance of the two heuristic algorithms is similar to their performance in the smaller cases, indicating that our algorithm also performs well for large problems. As the figure indicates, the algorithm developed in this chapter consistently achieves up to a 35% greater reduction in entropy than benchmark algorithm.

## 5.3 Summary

This chapter has provided insight into the way to best utilize limited communication capabilities in information-sensitive tasks, as well as the performance one can hope to achieve with a limited amount of communication capability. While some work has been done in this

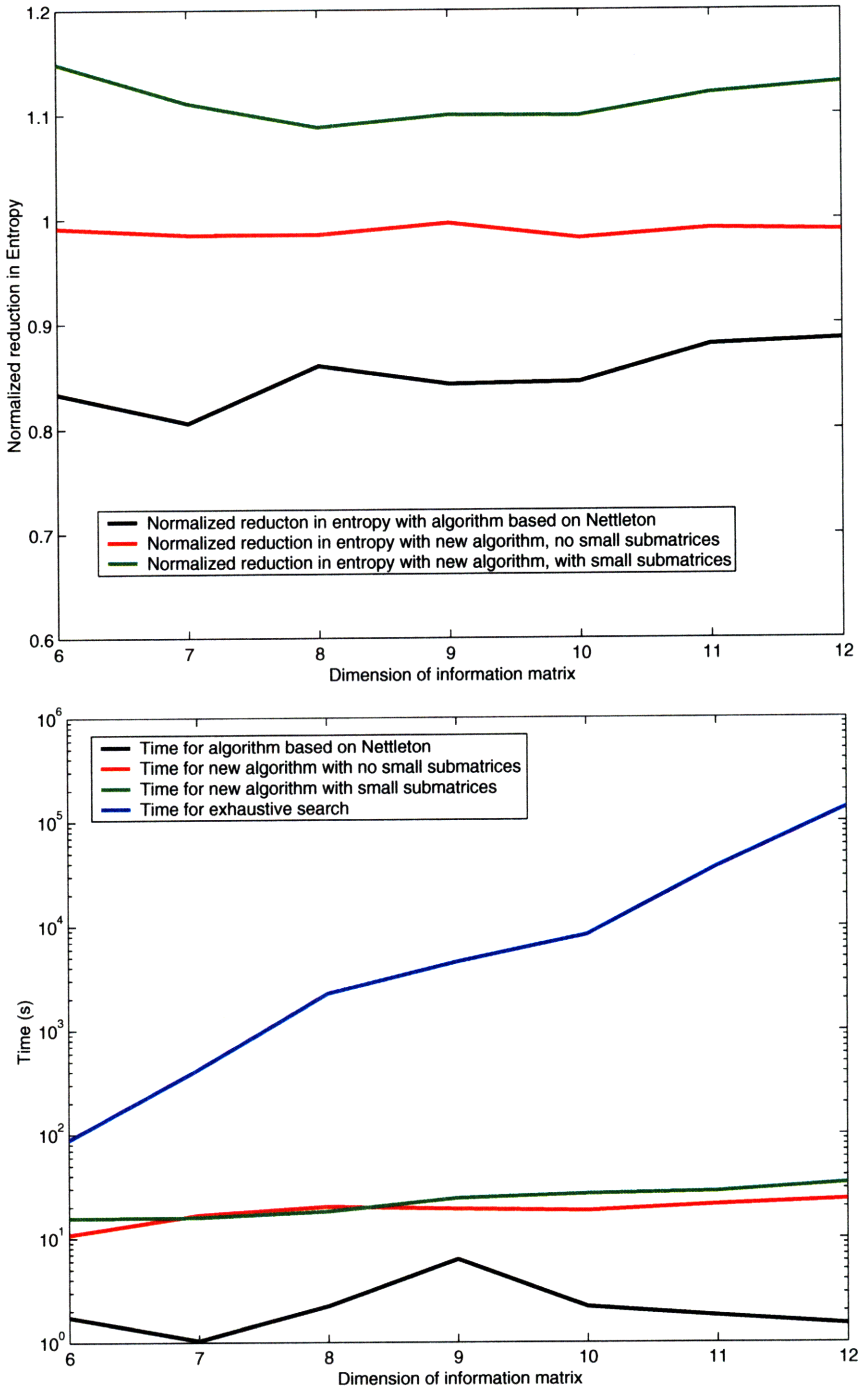



Figure 5-3: As these simulation results indicate, the algorithm developed in this chapter performs very well, reaching entropies very close to those achieved through exhaustive search even when no small submatrices are considered, with a great reduction in computation time. When small submatrices are considered, the algorithm presented in this chapter outperforms an exhaustive search over large submatrices with almost no increase in computation time.

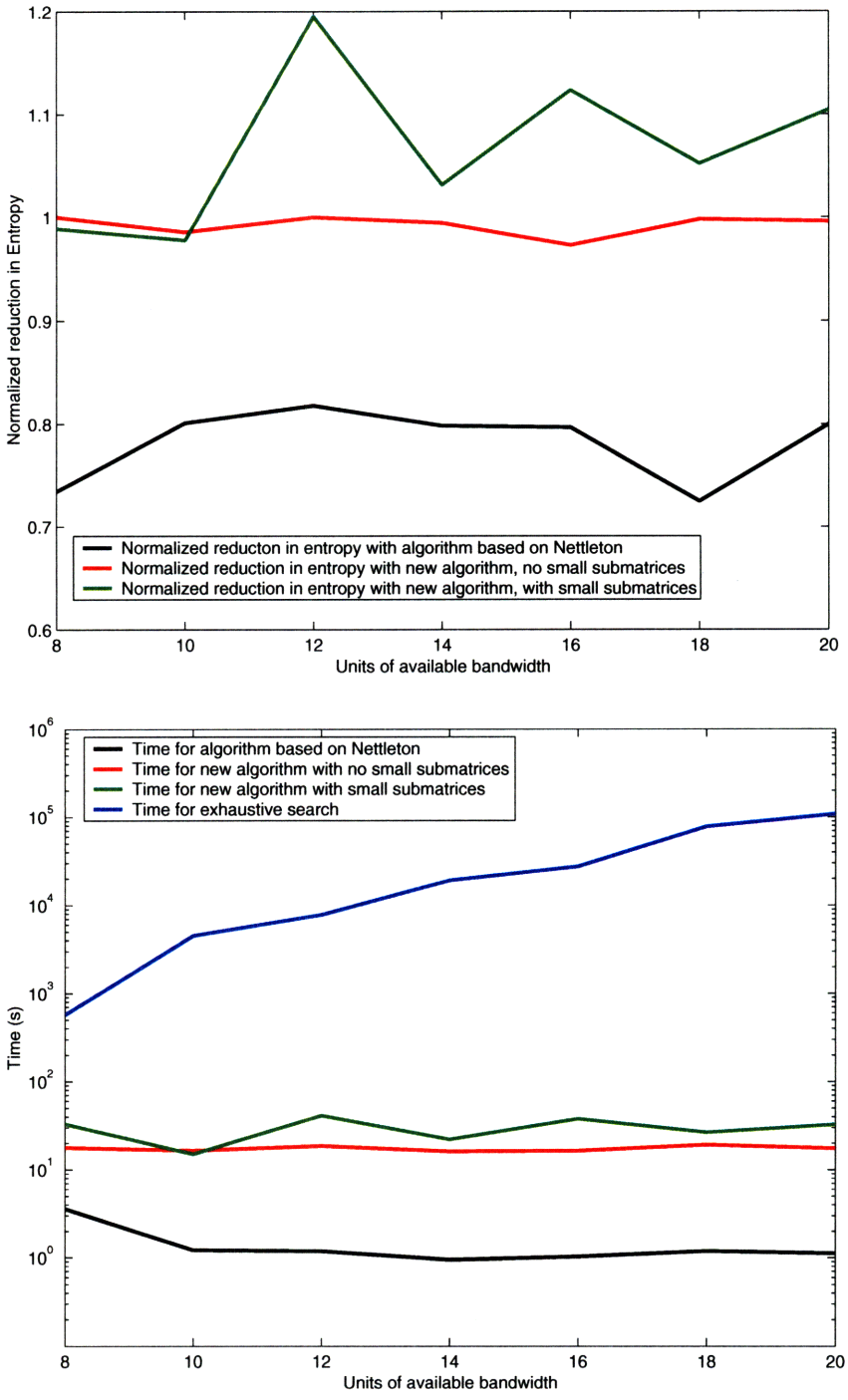



Figure 5-4: The algorithm developed in Section 5.2.2 also performs well for varying availability of communication throughput, and again computation time remains low.

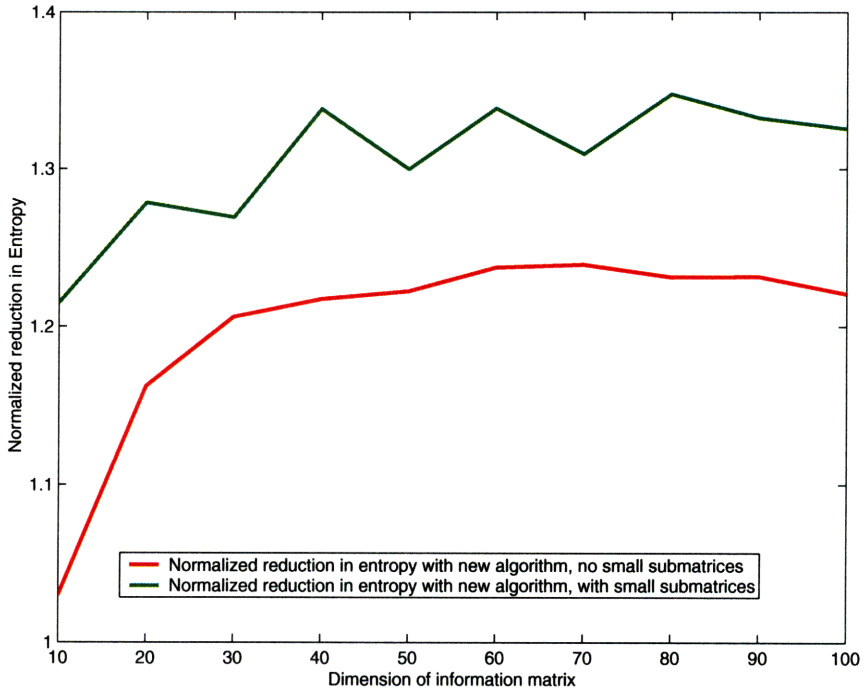



Figure 5-5: The algorithm presented in this chapter continues to outperform the benchmark algorithm as the number of map features and the availability of communication throughput increase to realistic magnitudes. Shown is the reduction in entropy obtained by this algorithm, normalized by the entropy reduction obtained by the benchmark algorithm.

area previously, the algorithms presented in this chapter provide significant improvements in performance relative to existing techniques, with relatively small increases in computational complexity.

For the case of a block-diagonal information matrix, a polynomial-time approximation algorithm that carries a theoretical performance guarantee was developed. This algorithm, which is based on reduction to a multiple-choice knapsack problem, provides provably good performance in computationally tractable time. Since block-diagonal information matrices arise quite naturally in certain sensing models, this algorithm has important practical applicability.

For the case of a fully-populated information matrix, this chapter has developed a heuristic algorithm that demonstrates good performance in simulation for problems of realistic complexity. Although it carries no performance guarantee, this algorithm can be applied to problems in which correlations exist between all pairs of feature locations being

estimated; thus, this algorithm has very general applicability.

# Chapter 6

# Conclusions and Future Work

The ability to explore and map an unknown environment is seen as a capability of central importance in autonomous robotics, and it is widely acknowledged that in many instances, this task will be done cooperatively and in a communication-limited environment. Yet, the impact of limited communication on the fulfillment of this task has not been adequately studied, nor have algorithms been developed to fully take advantage of the communication throughput available.

This thesis has made significant contributions to this area, both in terms of network optimization as the exploration task is being carried out, and in terms of efficient use of available communication throughput.

## 6.1 Contributions

The specific contributions of this thesis are as follows. In the area of network optimization, this thesis has:

- Developed an improved exact algorithm for maximizing the number of regular nodes that achieve a given minimum throughput level.
- Developed a polynomial-time approximation algorithm for maximizing the number of regular nodes that achieve a given minimum throughput level.

- Developed an improved exact algorithm for the MFPA problem - the new algorithm's computation time is 2-3 orders of magnitude less than existing techniques' for problems of practical scale.

- Developed an exact algorithm with computation time polynomial in the number of regular nodes for maximizing the total throughput achieved by all regular nodes, for the case in which regular nodes adjust their transmission power such that the mobile backbone nodes receive transmissions at the same power level for all regular nodes.

- Reduced the scheduling problem for a time-division multiple access scenario to an open shop scheduling problem and developed an LP-based algorithm for finding optimal time allocations to maximize total throughput.

- Developed a mobile backbone network optimization technique that can accommodate mobile, controlled regular nodes.

- Developed a greedy algorithm for cooperative exploration in mobile backbone networks with a performance guarantee (in some cases) and good empirical performance (in the general case).

In the area of transmission selection, this thesis has

- Formulated the entropy minimization problem in terms of the information filter and posed optimal data fusion as a semidefinite programming problem.

- Developed a polynomial-time approximation algorithm for minimizing post-transmission entropy in the case of block-diagonal information matrices.

- Developed a stochastic heuristic for arbitrary matrices that performs well in practice.

## 6.2 Future Work

The network optimization algorithms described in Chapter 3, while already a significant improvement over existing techniques, can be made more realistic and more general. For

example, modifications should be made to ensure that the algorithms perform robustly in the face of uncertain regular node location information. Additionally, the presence of obstacles in the environment impacts both the movement capabilities of the nodes, and the communication throughput achieved. While the impact of obstacles on regular node mobility is easy to model, determining the impact of obstacles on feasible mobile backbone node placements and the quality of the resulting solution requires additional work. Additionally, the throughput functions considered in this work only consider the distance between regular nodes and mobile backbone nodes and the number of regular nodes assigned, not the presence of obstacles between regular nodes and mobile backbone nodes.

The exploration algorithms described Chapter 4 are computationally tractable and carry a performance guarantee for some classes of problems. Future work will extend the analysis of these algorithms to more general classes of problems and will extend the algorithms themselves to more realistic communication models. However, the models presented in Chapters 3 and 4 already represent a significant improvement over the more simplistic, disc-based models present in the literature.

The transmission selection algorithms described in Chapter 5 provide a good solution to the static problem described in Section 5.2, but they do not address the dynamic decision problem set forth in Section 5.1. Future research will utilize the work described in this thesis in the solution of this dynamic decision problem.

Throughout Chapter 5 it was assumed that the sender possesses an accurate estimate of the receiver's information matrix, and this has played a key role in the development of the algorithms presented in this chapter. In practice, the quality of this estimate is unknown. Future work will examine the degradation of the solution quality with declining accuracy in the sender's estimate.

# Bibliography


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

[2] R. Ahuja, T. Magnanti and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[3] R. Ahuja, J. Orlin, C. Stein and R. Tarjan, "Improved Algorithms for Bipartite Network Flows," *SIAM Journal of Computing*, Vol. 23 , Issue 5, pp. 906-933, October 1994.

[4] M. Alighanbari and J. How, " Robust Decentralized Task Assignment for Cooperative UAVs." *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.

[5] A. S. Asratian, T. M. J. Denley and R. Häggkvist, *Bipartite Graphs and their Applications*. Cambridge University Press, 1998.

[6] R. Balasubramanian, S. Ramasubramanian and A. Efrat, "Coverage Time Characteristics in Sensor Networks," in *2006 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2006)*, October 2006.

[7] M. A. Batalin and G. S. Sukhatme, " The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* pp. 3478-3485, 2005.

[8] D. Bertsimas and R. Weismantel, *Optimization over Integers*, Dynamic Ideas, 2005, p. 88.

[9] W. Burgardt, M. Moorstt, D. Fox, R. Simmons and S. Thrun, "Collaborative Multi-Robot Exploration," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.

[10] Y. U. Cao, A. S. Fukunaga and A. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions," *Autonomous Robots*, Springer Netherlands, Vol 4, Num. 1, March, 1997.

[11] S. Chan, A. P. New and I. Rekleitis, "Distributed Coverage with Multi-Robot System," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL 2006.

[12] H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," *International Conference on Field and Service Robotics*, 1997.

[13] H. Choset, "Coverage for robotics – A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, 31: 113-126, Kluwer Academic Publishers, 2001.

[14] J. Cortes, S. Martinez, T. Karatas and F. Bullo, "Coverage control for mobile sensing networks," *IEEE International Conference on Robotics and Automation*, 2002.

[15] E. Craparo, J. How and E. Modiano, "Simultaneous Placement and Assignment for Exploration in Mobile Backbone Networks," submitted to the IEEE Conference on Decision and Control, 2008.

[16] E. Craparo, J. How and E. Modiano, "Optimization of Mobile Backbone Networks: Improved Algorithms and Approximation," *Proceedings of the American Control Conference*, June 2008.

[17] R. K. Dash, A. Rogers, N. R. Jennings, S. Reece and S. Roberts, "Constrained Bandwidth Allocation in Multi-Sensor Information Fusion: A Mechanism Design Approach", *Proceedings of The Eighth International Conference on Information Fusion*, Philadelphia, PA, 2005.

[18] R. Eustice, M. Walter and J. Leonard, "Sparse Extended Information Filters: Insights into Sparsification," In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, Canada, August 2005.

[19] S. Fujishige, *Submodular Functions and Optimization (Second Edition)*, Annals of Discrete Mathematics, Vol. 58, 2005.

[20] S. Ge and C-H. Fua, "Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, 2005.

[21] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *Journal of the ACM* Volume 23, Issue 4, October 1976, 665-679.

[22] B. Grocholsky, *Information-Theoretic Control of Multiple Sensor Platforms*. PhD thesis, The University of Sydney, 2002.

[23] S. J. Julier and J. K. Uhlmann, "A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations," *Proceedings of the American Control Conference*, 1997.

[24] K. Kar and S. Banerjee, "Node Placement for Connected Coverage in Sensor Networks," *Proceedings of WiOpt 2003*.

[25] J. M. Keil, "Decomposing a Polygon into Simpler Components," *SIAM Journal on Computing*, Volume 14, Issue 4, Pages 799-817, 1985.

[26] H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack Problems*. Springer-Verlag, 2004.

[27] E. L. Lawler and J. Labetoulle, "On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming," *Journal of the ACM*, Volume 25, Issue 4, October 1978, 612-619.

[28] K. Konolige, D. Fox, B. Limketkai, J. Ko and B. Stewart, "Map merging for distributed robot navigation," *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[29] A. Krause and C. Guestrin. "Near-optimal Nonmyopic Value of Information in Graphical Models." *Proc. of Uncertainty in Artificial Intelligence (UAI)*, 2005.

[30] A. Krause, C. Guestrin, A. Gupta, J. Kleinberg. "Near-optimal Sensor Placements: Maximizing Information while Minimizing Communication Cost." *Proc. of Information Processing in Sensor Networks (IPSN)*, 2006.

[31] K. S. Kwok, B. J. Driessen, C. A. Phillips and C. A. Tovey, "Analyzing the Multiple-target-multiple-agent Scenario Using Optimal Assignment Algorithms," *Journal of Intelligent and Robotic Systems* , Vol. 35, No. 1, September, 2002.

[32] E. Lawler, "Fast approximation algorithms for knapsack problems," *Mathematics of Operations Research*, 4:339-356, 1979.

[33] J. R. T. Lawton, R. W Beard and B. J. Young,. "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation*, Vol. 19, Issue: 6. Dec. 2003.

[34] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot." In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 1442-1447, Osaka, Japan, November 1991.

[35] J. McLurkin and J. Smith, "Distributed Algorithms for Dispersion in Indoor Environments using a Swarm of Autonomous Mobile Robots," Distributed Autonomous Robotic Systems Conference, June 23, 2004.

[36] A. G. O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. CRC Press, 1998.

[37] G. Nemhauser and L. Wolsey, "Maximizing submodular set functions: Formulations and Analysis of Algorithms," *Studies on Graphs and Discrete Programming*, P. Hansen, ed., 1981, pp. 279-301.

[38] E. Nettleton, S. Thrun, H. F. Durrant-Whyte and H. Sukkarieh, "Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles," *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003.

[39] F. Preparta and M. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[40] J. G. Proakis and M. Salehi, *Communication Systems Engineering*, 2nd Edition. Prentice Hall, 2002.

[41] M. Purchla, "Reduction of the near-far effect in mobile communication systems with Code-Division Multiple-Access," *Proceedings of SPIE*, Vol. 6159, 61594B, April 2006.

[42] S. Reece and S. Roberts, "Robust, Low-Bandwidth, Multi-Vehicle Mapping," *8th International Conference on Information Fusion*, 2005.

[43] I. Rekleitis, V. Lee-Shue, A. P. New and H. Choset, "Limited Communication, Multi-Robot Team Based Coverage," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA 2004. 71, 1998.

[44] I. Rubin, A. Behzadm R. Zhang, H. Luo, and E. Caballero, "TBONE: a Mobile-Backbone Protocol for Ad Hoc Wireless Networks," *Proc. IEEE Aerospace Conference*, 6, 2002.

[45] V. Sharma, M. Savchenko, E. Frazzoli and P. Voulgaris, "Transfer Time Complexity of Conflict-Free Vehicle Routing with No Communications," *International Journal of Robotics Research*, Vol. 26, No. 3, pp. 255-272. March 2007.

[46] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun and H. Younes, "Coordination for Multi-Robot Exploration and Mapping," *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.

[47] R. Smith, M. Self and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," *Autonomous Robot Vehicles*, p.167-193, 1990.

[48] J. Sun and E. Modiano, "Channel Allocation Using Pricing in Satellite Networks," 40th Annual Conference on Information Sciences and Systems, March 2006.

[49] A. Srinivas, *Mobile backbone architecture for wireless ad-hoc networks : algorithms and performance analysis*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[50] A. Srinivas and E. Modiano, "Joint node placement and assignment for throughput optimization in mobile backbone networks," To appear in *Proc. IEEE INFOCOM'08*, Apr. 2008.

[51] A. Srinivas, G. Zussmanm and E. Modiano, "Mobile Backbone Networks: Construction and Maintenance," *ACM MOBIHOC 2006*, May 2006.

[52] J. D. Sweeney, H. Li, R. A. Grupen and K. Ramamritham, "Scalability and Schedulability in Large, Coordinated, Distributed Robot Systems." *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, p. 4074- 4079, 2003.

[53] S. Thrun, "Exploration and Model Building in Mobile Robot Domains," in *Proceedings of the IEEE International Conference on Neural Networks*, 1993.

[54] S. Thrun. "Learning Metric-Topological Maps for Indoor Mobile Robot Navigation," *Artificial Intelligence*, 99:1.

[55] S. Thrun, "Robotic Mapping: A Survey." CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, 2002.

[56] S. Thrun, W. Burgard and D. Fox,*Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2005.

[57] S. Thrun, D. Fox and W. Burgard, "A probabilistic approach for concurrent map acquisition and localization for mobile robots." Technical Report CMU-CS-97-183, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1997.

[58] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani and H. Durrant-Whyte, H. "Simultaneous localization and mapping with sparse extended information filters." *International Journal of Robotics Research*, 23(7-8), 2004.

[59] J. K. Uhlmann, "General Data Fusion for Estimates with Unknown Cross Covariances," *Proceedings of SPIE, the International Society for Optical Engineering*, 1996.

[60] L. Vandenberghe, S. Boyd and S.-P. Wu, "Determinant maximization with linear matrix inequality constraints," *SIAM Journal on Matrix Analysis and Applications*, 19(2):499-533, 1998

[61] M. Walter, R. Eustice and J. Leonard, "A Provably Consistent Method for Imposing Sparsity in Feature-based SLAM Information Filters," *Proceedings of the 12th International Symposium of Robotics Research (ISRR)*, San Francisco, CA, October 2005.

[62] K. Xu, X. Hong, and M. Gerla, "Landmark Routing in Ad Hoc Networks with Mobile Backbones," *Journal of Parallel and Distributed Computing*, 63, 2, pp 110-122, 2003.

[63] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. of the Second International Conference on Autonomous Agents*, Minneapolis, MN, USA, 1998.

[64] R. M. Zlot, A. Stentz, M. B. Dias and S. Thayer, "Multi-Robot Exploration Controlled By A Market Economy," *IEEE International Conference on Robotics and Automation*, May, 2002.