DISCRETE AND GEOMETRIC APPROACHES TO LIFETIME MAXIMIZATION IN WIRELESS SENSOR NETWORKS

By

BEHNAM BEHDANI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2012

To my dearest parents, Abolghasem and Fatemeh

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. J. Cole Smith, for his guidance, patience, and constant support throughout my years at the University of Florida. He has been a great mentor for me and I learnt so much from him, not only about math but also life lessons. This accomplishment would not have been possible without his help and support.

I would also like to thank Dr. Joseph Geunes, Dr. Yongpei Guan, and Dr. Ye Xia for serving on my Ph.D. committee. Their valuable suggestions and comments certainly helped to improve the quality of this dissertation. I must particularly thank Dr. Xia for being a great coauthor and instructor for me.

During my time at the University of Florida, I had the luxury of meeting many fine individuals who became great friends of mine. I must specially thank my great friends Ehsan Salari and Soheil Hemmati for all the help and support that they provided me with during my years in Gainesville. I would also like to thank my dear friends Vera Tomaino, Somar Koria, Donatella Granata, Gudbjort Gylfadottir, Arni Olafur Jonsson, Saed Alizamir, Masuad Zarepisheh, Sibel Bilge Sonuc, Petros Xanthopoulos, Dmytro Korenkevych, Clay Koschnick, Mike Prince, Saeed Ghadimi, Reza Skandari, Ashwin Arulselvan, Oleg Shylo, Altannar Chinchuluun, Siqian Shen, Mehmet Onal, and Alexey Sorokin for all the wonderful moments and memories that I will always remember them by.

And last but certainly not the least, my special thanks goes to my dearest parents, Abolghasem and Fatemeh, and my lovely brothers and sisters, Behzad, Bahareh, Behrouz, and Bahereh, for their endless love and constant support. I owe them everything that I have, and am forever grateful to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

DISCRETE AND GEOMETRIC APPROACHES TO LIFETIME MAXIMIZATION IN WIRELESS SENSOR NETWORKS

By

Behnam Behdani

August 2012

Chair: Jonathan Cole Smith
Major: Industrial and Systems Engineering

We address several problems related to the efficient use of energy resources in wireless sensor networks (WSNs). We first consider the problem of maximizing the lifetime of a WSN with energy-constrained sensors and a mobile sink. In this setting, the sink travels among discrete locations to gather information from all sensors. Data can be relayed among sensors and then to the sink, as long as the sensors and the sink are within a certain threshold distance of each other. However, sending information along a data link consumes energy at both the sender and the receiver nodes. We describe linear programming and column generation approaches for the problem of maximizing the lifetime of a WSN. We also propose decomposition and distributed algorithms for the case in which the underlying application can tolerate some degree of delay in delivering data to the sink. Our distributed algorithm can be embedded into a network protocol so that the senor nodes and the sink can run it directly as part of the network operation.

We also address a variant of the traveling salesman problem known as the close-enough traveling salesman problem that is useful in devising energy-aware trajectories for the sink in a WSN. We formulate a mixed-integer programming model based on a discretization scheme for the problem. Both lower and upper bounds on the optimal tour length can be derived from the solution of this model, and the quality of the bounds obtained depends on the fidelity of the discretization scheme. Finally, we propose an extension to the lifetime maximization problem by examining the case

in which the sink periodically travels at finite speed among a subset of sink locations to collect data from sensor nodes. In addition to the decisions regarding data routing and the sink's dwelling times, the problem finds a tour for the sink among a subset of sink locations. We provide mixed-integer programming formulations to model this problem, along with cutting planes, preprocessing techniques, and a Benders decomposition algorithm to improve its solvability. Computational results demonstrate the efficiency of the proposed algorithms.

CHAPTER 1
INTRODUCTION

Advances in low-power wireless communications and microprocessor technologies during the past decade have triggered the rapid development and commercialization of low-cost wireless sensors that can constantly collect information from their surroundings. A wireless sensor network (WSN) consists of a large number of wireless sensors that are densely deployed in an area for monitoring purposes. Sensors usually measure a physical attribute of their surrounding area (such as the temperature) or sense the movement of objects in their vicinity. Each sensor contains a microprocessor with limited processing abilities, a limited amount of memory, a limited power source, and one or more sensing devices that continuously collect information from the sensor's surroundings. The collected information needs to be transmitted for further processing to a data collecting node called the *sink*. The sensors are usually deployed randomly, but in high quantities to prevent coverage breach. This dense deployment of the sensor nodes allows multihop delivery of the collected information: Each sensor can communicate with a few neighbors and use them to relay its collected information to the sink. The need for energy conservation and frequency reuse makes multihop routing an essential component of WSNs [1, 2]. Despite their simple structure, WSNs provide a useful monitoring tool in many real-life settings such as environmental monitoring, home automation and health-care operations.

It is well known that the energy required for processing a unit of data is much less than the energy required for its transmission [3]. Each sensor node constantly consumes energy to send its own data as well as to relay (possibly some of) its adjacent nodes' data to the sink or other intermediate nodes. However, sensor batteries are often irreplaceable due to the large number of sensors in the network and the fact that sensors may be deployed in hard-to-reach or hostile environments. Therefore, devising energy-aware routing schemes for sensor nodes is of crucial importance in the design

and operation of WSNs [4]. Network lifetime is usually defined as the time until the first sensor node completely exhausts its energy [5, 6]. Energy-aware operation of WSNs has been the focus of many researchers lately [7–10]. In this dissertation, we focus on mathematical-programming-based approaches to energy-aware operation of WSNs.

In Chapter 2, we consider the problem of maximizing the lifetime of a WSN with a mobile sink that moves over a set of pre-specified sink locations to collect data from sensor nodes. The decisions to be made in this model include the sink's dwelling time at each sink location, as well as data transmission patterns from sensor nodes to the sink at different sink locations. We develop a column generation algorithm for the lifetime maximization problem in this setting, and also for the case in which the underlying application can tolerate some amount of delay in data transmission. Exploiting the underlying network-flows structure allows us to further simplify the subproblem into separable shortest-path problems. Column generation algorithms are important not only because of their theoretical and computational contribution, but also because they provide a foundation for the development of semi-distributed algorithms for the lifetime maximization problem. Distributed algorithms are generally more widely used and valued compared to their centralized counterparts because they can be incorporated into the network protocol and be used as network control mechanisms. In Chapter 3, we provide a fully-distributed algorithm for the lifetime maximization problem in a delay-tolerant WSN with a mobile sink. Our algorithm is proven to converge to an optimal lifetime in the long-run.

We also address the problem of finding an energy-efficient trajectory for a mobile sink in WSNs where the sink needs to travel to different areas of the sensor field to collect information from sensor nodes. It has been shown by Ciullo et al. [11] that an energy-efficient trajectory for the sink gets closer to sensor nodes with higher data generation rates. We use this fact to model this problem as an instance of the close-enough traveling salesman problem (CETSP) [12, 13]. The CETSP is a geometric

generalization of the well known traveling salesman problem (TSP) [14]. While in the TSP the salesman has to visit all customer locations, in the CETSP, each customer has a compact neighborhood set around its location within which the customer is willing to travel to meet with the salesman. Therefore, each customer is considered to be visited if the salesman's tour visits at least one point inside its neighborhood set. We propose an integer-programming-based approach that can solve the CETSP with any level of accuracy in Chapter 4.

Finally, we present an important extension to the lifetime maximization problem in WSNs with a mobile sink by lifting the commonly-used assumption that the sink's travel times are negligible. In many environmental monitoring applications of WSNs, assuming zero travel times for the sink is not realistic due to the large area spanned by the network. We formulate this problem as a mixed-integer program and provide cutting-plane and decomposition algorithms to solve it in Chapter 5.

CHAPTER 2
DECOMPOSITION ALGORITHMS FOR MAXIMIZING THE LIFETIME OF WIRELESS SENSOR NETWORKS WITH MOBILE SINKS

2.1 Literature Survey

A wireless sensor network (WSN) typically consists of a large number of sensor nodes that gather information from their neighborhoods. The sensors can communicate with each other via wireless links if they are within their mutual communication range. The collected data will be delivered to the *sink* for further processing. The delivery is often through multi-hop communication with the sensors relaying the data toward the sink. The sensor nodes are usually expected to operate with batteries and may be deployed in not-easily-accessible or hostile environments in large quantities. Therefore, it may be difficult or impossible to replace or recharge the batteries. On the other hand, the sink is typically rich in energy. Because the sensor energy is a scarce resource in a WSN, efficient energy utilization to prolong the WSN lifetime has been an important research subject. Recently, how to improve WSN lifetime by taking advantage of mobility has attracted significant attention from researchers [7–10, 15–19]. In this chapter, we will focus on sink mobility.

Following the convention, we define the lifetime of a WSN as the time until the first sensor node exhausts its energy [5, 6]. If the sink is immobile, the energy consumption rate can be severely imbalanced with the nodes near the sink exhausting their energy much sooner [20–22]. This is known as the “energy hole problem”. The reason is that the collected data is directed toward the sink and the nodes near the sink can be burdened with relaying a large amount of traffic from other nodes [23]. By moving the sink in the sensor field, one can mitigate the energy hole problem and expect an increased network lifetime.

Various lifetime-maximization problems are routinely formulated as linear or integer optimization problems [5–10, 24–26]. This chapter focuses on two relatively recent linear programming formulations that involve a mobile sink. They will be called the

Mobile Sink Model (MSM) and the *Delay-Tolerant Mobile Sink Model* (DT-MSM). The difference between the two is that DT-MSM is relevant to applications that tolerate delayed information delivery. Our main contributions are column generation algorithms for these problems. These algorithms can help to overcome two challenges. First, solving these problems can be very time-consuming using (centralized) standard linear programming (LP) solvers. We will demonstrate that our algorithm for the MSM is often significantly faster than highly-tuned linear programming solvers even when it runs in a centralized fashion. Although our algorithm for the DT-MSM is often slower than those LP solvers, it enjoys other potential advantages associated with being a decomposition algorithm. [1]

Second, there are few known decomposition algorithms for solving these two problems that are also fast. Decomposition algorithms are often more useful and more sought-after than monolithic ones in the networking area since portions of the algorithms can be built into network protocols and become network control algorithms. Our algorithm for the MSM involves solving *separate* shortest-path subproblems. This structural property can be a key to adapting the algorithm into distributed network control algorithms, similar to how Dijkstra's algorithm is implemented distributedly [27]. Moreover, if the goal is algorithm speed instead of network-wide distributed implementation, these subproblems can be solved in parallel on multiple CPUs to further speed up the process of finding an optimal solution. The algorithm for the DT-MSM, after decomposition, involves solving a single shortest-path subproblem on a special network with the same set of nodes as the original network and a moderately larger arc

[1] Finding an optimal solution with a centralized solver can be useful. In some applications, it is possible to manage the network with a central controller, which can control the network elements based on the knowledge of the optimal solution. In addition, even when decentralized algorithms must be applied in practice, an optimal centralized solution provides a performance benchmark and shows how much lifetime improvement opportunity may exist.

set. Hence, distributed implementation of the shortest-path algorithm appears possible in the DT-MSM case as well.

We next briefly review related work. Gatzianas and Georgiadis [9] formulate a similar lifetime maximization problem to our first problem, the MSM, and provide a distributed subgradient algorithm to solve it. However, they do not exploit the uncapacitated nature of the flow between the sensor nodes and the sink. As a result, they solve minimum cost flow subproblems instead of shortest path subproblems. Another drawback of their algorithm is that it is very sensitive to the choice of starting dual multipliers, while the column generation algorithms are known to be relatively robust to the choice of the starting point. Sankar and Liu [25] formulate a static-sink lifetime-maximization problem and give a distributed, local algorithm based on a general network-flow algorithm. Unlike the subgradient algorithm, it relaxes the flow-conservation constraints instead of the energy constraint.

Yun and Xia [26] propose a framework of extending the network lifetime by taking advantage of both sink mobility and application delay tolerance. For situations where the underlying applications tolerate delayed information delivery (see [16]), each sensor node can store the data temporarily for up to a prescribed maximum delay, and transmit it when the mobile sink is at the location most favorable for achieving the longest network lifetime. The proposed model in [26] is exactly the DT-MSM in this chapter. In this chapter we give a column generation algorithm for the DT-MSM, whereas finding efficient algorithms is not a focus of [26].

Wang et al. [8] formulate a linear programming problem (LP) for determining how to move the mobile sink and how long the sink should stay at each stop along the movement path in order to maximize the network lifetime. Unlike the models we consider in this chapter, they do not incorporate the routing decisions into their optimization problem. Instead, they use shortest path algorithms to determine how to route the data from each sensor node to the sink. Although shortest path routing protocols are

common choices in practice (see for example [28]), in this case, it causes an imbalance in energy consumption which results in shorter lifetime [29].

The problems of selecting an optimal set of sink stops and/or finding an optimal tour of the locations are very hard in general. Shi and Hou [10] investigate how to find the optimal sink stops and compute the length of stay at each of the stops. When the candidate locations where the sink may stop are not specified, the problem is NP-hard, and the authors propose an approximation algorithm for the problem. In contrast, we restrict the set of sink stops to be from a set of given candidate locations in this chapter.

The rest of the chapter is organized as follows. Section 2.2 describes the network and energy consumption concepts employed throughout the chapter. Section 2.3 presents a column generation algorithm for the MSM, while Section 2.4 describes a column generation algorithm for the DT-MSM. We evaluate the computation time of our algorithms in Section 2.5.

2.2 Network and Energy Consumption Model

We begin by briefly discussing some key concepts and notation relevant to the models presented in this chapter. Let N be the set of sensor nodes. Each node i starts with an initial energy value, denoted by E_i, and, during its lifespan, generates data at a constant rate of d_i.

Let L be the set of possible locations where the sink can stop (also known as *sink stops*). The sink can move within the sensor field, stop at one of these locations to gather data from the sensor nodes for a period of time, and then move on to another location. The sink does not necessarily stop (i.e., stay for a positive duration) at all locations in L in the interest of maximizing the network lifetime [8, 10].

Throughout this chapter, we assume that the sink's traveling time between locations is negligible. This is appropriate when the sink's traveling time is much smaller than the time spent by the sink to collect data in each location. With this assumption, the order of visiting the sink stops does not affect the network lifetime.

Similar to [30, 31], the energy required per unit of time to transmit data at the rate of x_{ij} on the wireless link (i,j) can be modeled as[2]

$$E_{ij}^t = C_{ij}^t x_{ij}, \tag{2–1}$$

where C_{ij}^t is the required energy for transmitting one unit of data on the link (i,j). This value is given by

$$C_{ij}^t = \alpha + \beta \tau_{ij}^e, \tag{2–2}$$

where τ_{ij} is the Euclidean distance between nodes i and j, α and β are non-negative constants, and e is the path loss exponent. Typically, e is in the range of 2 to 6, depending on the environment. Note that C_{ij}^t does not depend on the link rate, and, in the low rate regime, this is a valid assumption. Hence, we need to assume that the traffic rate x_{ij} is sufficiently small compared to the capacity of the wireless link.

The energy spent at node i per unit of time for receiving data from node k is given by

$$E_{ki}^r = \gamma x_{ki}, \tag{2–3}$$

where γ is a given constant.

This chapter does not consider MAC-layer contention. It is assumed that contention is resolved by some MAC-layer protocol. The operation of the MAC-layer protocol determines the link rates, which are assumed to be large enough so that they do not impose a constraint on the data rates.

[2] Note that when we refer to a wireless link (i,j), i is a sensor node, and j may be a sensor node or the sink in its current location. This will be made more precise when needed.

2.3 Mobile Sink Model without Delay Tolerance

In this section, we formulate the lifetime-maximization problem in a WSN with a mobile sink but without delay tolerance and provide a column generation algorithm for its solution.

The *sink sojourn time* at a location $l \in L$, represented by decision variables t_l, is the time that the sink spends at l to collect data from the sensor nodes. The overall network lifetime is $Z = \sum_{l \in L} t_l$. Note that if the sink does not stop at location $l \in L$, then $t_l = 0$.

When the sink is at stop l, the set of neighbors of node i is defined as $S_i^l = \{j \in N \cup \{l\} | \tau_{ij} \leq \bar{\tau}\}$, where $\bar{\tau}$ is the maximum transmission range of any node. Note that depending on the distance to l, S_i^l can include the sink itself. To find the optimal network lifetime, we need to consider the routing of the data (including rate) as well as the sink's sojourn time at each stop (also see [7–10]).

Let x_{ij}^l be the flow on link (i, j) while the sink is at stop l. The lifetime maximization problem can be formulated as follows [7].

$$\max \sum_{l \in L} t_l \tag{2–4}$$

$$\text{s.t.} \sum_{l \in L} t_l \left(\sum_{j \in S_i^l} C_{ij}^t x_{ij}^l + \sum_{j : i \in S_j^l} \gamma x_{ji}^l \right) \leq E_i \qquad \forall i \in N \tag{2–5}$$

$$\sum_{j : i \in S_j^l} x_{ji}^l + d_i = \sum_{j \in S_i^l} x_{ij}^l \qquad \forall i \in N, l \in L \tag{2–6}$$

$$t_l \geq 0 \qquad \forall l \in L \tag{2–7}$$

$$x_{ij}^l \geq 0 \qquad \forall i \in N, l \in L, j \in S_i^l. \tag{2–8}$$

Constraints (2–5) state that the total energy consumed at any node i cannot exceed the initial energy E_i. Constraints (2–6) ensure flow conservation for all nodes when the sink is at l.

To convert the above formulation into an LP, one can introduce variables y_{ij}^l representing the volume of data transmitted from node i to the node j through the

link (i, j), while the sink is at location l. That is, y_{ij}^l is equivalent to $x_{ij}^l t_l$.

$$\max \sum_{l \in L} t_l \tag{2–9}$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} C_{ij}^t y_{ij}^l + \sum_{j : i \in S_j^l} \gamma y_{ji}^l \right) \leq E_i \qquad \forall i \in N \tag{2–10}$$

$$\sum_{j : i \in S_j^l} y_{ji}^l + d_i t_l = \sum_{j \in S_i^l} y_{ij}^l \qquad \forall i \in N,\ l \in L \tag{2–11}$$

$$t_l \geq 0 \qquad \forall l \in L \tag{2–12}$$

$$y_{ij}^l \geq 0 \qquad \forall i \in N,\ j \in S_i^l,\ l \in L. \tag{2–13}$$

Note that constraints (2–11) can be obtained by multiplying constraints (2–6) by t_l and substituting by y_{ij}^l.

We now describe a standard column generation algorithm for the linear program (2–9)–(2–13), and then present modifications that allow us to exploit the underlying network structure. Consider the set of feasible solutions to (2–11)–(2–13). The well known representation theorem (see, e.g., [32]) states that one can represent a polyhedral set as a convex combination of its extreme points plus a non-negative combination of its extreme rays. Because (2–11)–(2–13) is a conic set, it has a single extreme point that lies at the origin, and so all of its points can be represented as a non-negative combination of its extreme rays. Now, let K be the set of all extreme rays of (2–11)–(2–13), indexed by $k \in K$. Denote by $\overline{y}_{ij}^{lk}$ and $\overline{t}_l^k$ the components of vector $k \in K$. Then finding a feasible solution to (2–9)–(2–13) is equivalent to finding

non-negative scalars $\mu_k,\ k \in K$, such that:

$$\begin{aligned}
& y_{ij}^l = \sum_{k \in K} \overline{y}_{ij}^{lk} \mu_k && \forall i \in N,\ j \in S_i^l,\ l \in L \\
& t_l = \sum_{k \in K} \overline{t}_l^k \mu_k && \forall l \in L \\
& \sum_{l \in L} \left(\sum_{j \in S_i^l} C_{ij}^t y_{ij}^l + \sum_{j : i \in S_j^l} \gamma y_{ji}^l \right) \leq E_i && \forall i \in N \\
& \mu_k \geq 0 && \forall k \in K.
\end{aligned} \tag{2–14}$$

Hence, we can reformulate our problem as follows.

$$\max \sum_{l \in L} \sum_{k \in K} \overline{t}_l^k \mu_k \tag{2–15}$$

$$\text{s.t.} \sum_{l \in L} \sum_{k \in K} \left(\sum_{j \in S_i^l} C_{ij}^t \overline{y}_{ij}^{lk} + \sum_{j : i \in S_j^l} \gamma \overline{y}_{ji}^{lk} \right) \mu_k \leq E_i \qquad \forall i \in N \tag{2–16}$$

$$\mu_k \geq 0 \qquad \forall k \in K. \tag{2–17}$$

This problem is normally referred to as the *master problem* and each of the variables μ_k is called a *column*. As an alternative to solving this master problem, we can iteratively solve a *restricted* master problem with a subset M of all columns K and add columns to M as necessary. In each iteration, a restricted master problem is solved and a new set of dual variables $\delta_i, \forall i \in N$, associated with constraints (2–16) is obtained. Using these dual variables, we can calculate the minimum reduced cost over all columns in K. If the minimum reduced cost is zero, the current restricted master problem solution is optimal to the original LP. Otherwise, there exists a column having a negative reduced cost, so we can potentially improve our solution by adding that column to M and re-solving the reduced master problem. To find a column having the minimum reduced cost, constrained by (2–11)–(2–13), we can solve the following *subproblem*.

$$\min \sum_{l \in L} t_l - \sum_{l \in L} \sum_{i \in N} \left(\sum_{j \in S_i^l} \delta_i C_{ij}^t y_{ij}^l + \sum_{j : i \in S_j^l} \delta_i \gamma y_{ji}^l \right) \tag{2–18}$$

$$\text{s.t.} \sum_{j:i\in S_j^l} y_{ji}^l + d_i t_l - \sum_{j\in S_i^l} y_{ij}^l = 0 \qquad \forall i \in N,\ l \in L \qquad (2\text{–}19)$$

$$t_l \geq 0 \qquad \forall l \in L \qquad (2\text{–}20)$$

$$y_{ij}^l \geq 0 \qquad \forall i \in N,\ j \in S_i^l,\ l \in L. \qquad (2\text{–}21)$$

If the optimal objective function value to this subproblem is zero, we can stop and conclude that there is no improving column. Otherwise, the subproblem must be unbounded, noting that the right-hand-sides of all constraints are zero, and that the feasible region is conic. In the latter case, we will add the corresponding direction of unboundedness as a new column and re-solve the master problem.

The column generation algorithm described above can be improved in several ways. Here, we describe a modified version of the algorithm, which leads to a subproblem that is much easier to solve. First, notice that the subproblem (2–18)–(2–21) is separable into $|L|$ linear programs, because the objective function is linear and the constraints governing flows for each sink location are disjoint. Therefore, it is more efficient to generate one column (in the lower dimension) from each of these conic sets in each iteration.

Note that an optimal solution exists to (2–9)–(2–13) in which there are no cyclic flows for any sink location l. Hence, in our column generation algorithm, we will only consider solutions with acyclic flow patterns. For convenience, we will refer to the set of feasible solutions to (2–11)–(2–13) corresponding to sink location l as the lth conic set. Let $\mathcal{C}_l = \{(\mathbf{y}^l, t_l) : (\mathbf{y}^l, t_l)$ is feasible to the lth conic set, $\mathbf{y}^l$ is acyclic$\}$. This is the set from which we will extract the necessary columns for the master problem. Once again, all feasible solutions to $\mathcal{C}_l$ can be written as non-negative linear combinations of its extreme rays. By setting $t_l = 1$ in (2–11)–(2–13), we get the following normalized set for each

$l \in L$:

$$\begin{aligned} \sum_{j:i\in S_j^l} y_{ji}^l + d_i = \sum_{j\in S_i^l} y_{ij}^l \qquad & \forall i \in N \\ y_{ij}^l \geq 0 \qquad & \forall i \in N,\ j \in S_i^l. \end{aligned} \tag{2–22}$$

Now, let $\mathcal{D}_l = \{\mathbf{y}^l : \mathbf{y}^l \text{ is feasible to (2–22)}, \mathbf{y}^l \text{ is acyclic}\}$. Then, the following lemma holds.

Lemma 2.1. *The set of all extreme rays of $\mathcal{C}_l$ is in one-to-one correspondence with the set of all extreme points of $\mathcal{D}_l$.*

Proof. First notice that each feasible vector $\mathbf{y}^l \in \mathcal{D}_l$ corresponds to a ray $(\mathbf{y}^l, 1)$ of $\mathcal{C}_l$. That is, this ray has the same y_{ij}^l components as $\mathbf{y}^l$, and $t_l = 1$. Moreover, if $\mathbf{y}^l$ is an extreme point of $\mathcal{D}_l$, then the vector $(\mathbf{y}^l, 1)$ is an extreme ray of the corresponding conic set $\mathcal{C}_l$. To see this, consider two distinct rays $(\mathbf{y}^{l\prime}, t_l')$ and $(\mathbf{y}^{l\prime\prime}, t_l'')$ in $\mathcal{C}_l$ such that $(\mathbf{y}^l, 1) = \lambda_1(\mathbf{y}^{l\prime}, t_l') + \lambda_2(\mathbf{y}^{l\prime\prime}, t_l'')$, where $\lambda_1, \lambda_2 > 0$ and $\lambda_1 + \lambda_2 = 1$. Note that since $\mathbf{y}^{l\prime}$ and $\mathbf{y}^{l\prime\prime}$ are assumed to be acyclic flows, t_l' and t_l'' are non-zero scalars. Therefore, $(\mathbf{y}^l, 1) = (\lambda_1 t_l')((1/t_l')\mathbf{y}^{l\prime}, 1) + (\lambda_2 t_l'')((1/t_l'')\mathbf{y}^{l\prime\prime}, 1)$ where $\lambda_1 t_l' + \lambda_2 t_l'' = 1$. This means that $(\mathbf{y}^l, 1)$ can be represented as a convex combination of two normalized rays $((1/t_l')\mathbf{y}^{l\prime}, 1)$ and $((1/t_l'')\mathbf{y}^{l\prime\prime}, 1)$, and so $\mathbf{y}^l$ itself is a convex combination of $(1/t_l')\mathbf{y}^{l\prime}$ and $(1/t_l'')\mathbf{y}^{l\prime\prime}$ with the same coefficients. The fact that $(1/t_l')\mathbf{y}^{l\prime}$ and $(1/t_l'')\mathbf{y}^{l\prime\prime}$ belong to $\mathcal{D}_l$ implies that $(1/t_l')\mathbf{y}^{l\prime} = (1/t_l'')\mathbf{y}^{l\prime\prime}$, which in turn gives us that $(\mathbf{y}^l, 1)$ is an extreme ray of $\mathcal{C}_l$. To prove the reverse, suppose that the normalized direction $(\mathbf{y}^l, 1)$ is an extreme ray of $\mathcal{C}_l$. We claim that $\mathbf{y}^l$ is an extreme point of $\mathcal{D}_l$. If not, then we can find $\mathbf{y}_1^l$ and $\mathbf{y}_2^l$ feasible in $\mathcal{D}_l$ such that $\mathbf{y}^l = \lambda\mathbf{y}_1^l + (1-\lambda)\mathbf{y}_2^l$ where $0 < \lambda < 1$. Therefore, $(\mathbf{y}^l, 1) = \lambda(\mathbf{y}_1^l, 1) + (1-\lambda)(\mathbf{y}_2^l, 1)$. This is a contradiction because $(\mathbf{y}_1^l, 1)$ and $(\mathbf{y}_2^l, 1)$ are rays of $\mathcal{C}_l$, which would then indicate that $(\mathbf{y}^l, 1)$ is not an extreme ray of $\mathcal{C}_l$. □

Since all the extreme rays to $\mathcal{C}_l$ have a one-to-one correspondence to the extreme points of $\mathcal{D}_l$, we can generate columns from $\mathcal{D}_l$ instead of $\mathcal{C}_l$. We will show that this allows the subproblem to be solvable by Dijkstra's algorithm.

Now, suppose that K_l is the set of all columns generated from $\mathcal{D}_l$. We denote the components of such a column by $\overline{y}_{ij}^{lk}$ where $k \in K_l$. Then, a feasible solution to $\mathcal{C}_l$ can be written in the form $y_{ij}^l = \sum_{k \in K_l} \overline{y}_{ij}^{lk} t_{lk}$ and $t_l = \sum_{k \in K_l} t_{lk}$, where each $t_{lk} \geq 0$. Plugging this solution into the energy constraint (2–10), we get the following master problem.

$$\max \sum_{l \in L} \sum_{k \in K_l} t_{lk} \tag{2–23}$$

$$\text{s.t.} \sum_{l \in L} \sum_{k \in K_l} \left(\sum_{j \in S_i^l} C_{ij}^t \overline{y}_{ij}^{lk} + \sum_{j : i \in S_j^l} \gamma \overline{y}_{ji}^{lk} \right) t_{lk} \leq E_i \qquad \forall i \in N \tag{2–24}$$

$$t_{lk} \geq 0 \qquad \forall l \in L,\ k \in K_l. \tag{2–25}$$

Observe that for each sink location $l \in L$, there is a set of K_l different shortest path trees, each with a weight of $t_{lk} \geq 0$ in the master problem. We can interpret t_{lk} as the length of time that the sensors communicate with the sink (while at location $l \in L$) using the shortest path tree $k \in K_l$.

Let $\delta_i \geq 0$ be the dual variable associated with the ith constraint in (2–24). Then, the reduced cost for a new column corresponding to variable t_{lk} and having components $\widetilde{y}_{ij}^l$ is:

$$1 - \sum_{i \in N} \left(\sum_{j \in S_i^l} \delta_i C_{ij}^t \widetilde{y}_{ij}^l + \sum_{j : i \in S_j^l} \delta_i \gamma \widetilde{y}_{ji}^l \right). \tag{2–26}$$

The lth subproblem is written as follows.

$$\min \sum_{i \in N} \left(\sum_{j \in S_i^l} \delta_i C_{ij}^t \widetilde{y}_{ij}^l + \sum_{j : i \in S_j^l} \delta_i \gamma \widetilde{y}_{ji}^l \right) \tag{2–27}$$

$$\text{s.t.} \sum_{j : i \in S_j^l} \widetilde{y}_{ji}^l + d_i = \sum_{j \in S_i^l} \widetilde{y}_{ij}^l \qquad \forall i \in N \tag{2–28}$$

$$\widetilde{y}_{ij}^{l} \geq 0 \qquad \forall i \in N,\ j \in S_i^l. \tag{2–29}$$

The underlying network for subproblem $l \in L$ is the same as the communication graph between the sensors and the sink while at location l, with an excess of d_i at each sensor node i and a deficit of $\sum_{i \in N} d_i$ at the sink. The cost of the arc between i and j is $\delta_i C_{ij}^t + \delta_j \gamma$. Each subproblem has the form of a minimum cost flow problem, but with infinite capacity on the arcs of the underlying network. Thus, each sensor node sends its excess flow along a shortest path to the sink. These shortest paths can be obtained by solving one shortest path problem that computes a shortest path tree from all sensor nodes to the sink. Note that because each $\delta_i \geq 0$, the costs on the arcs of the new network are all non-negative and we can use Dijkstra's algorithm to find the shortest paths. The optimal flow can be obtained by aggregating the flows on all $|N|$ paths in the original network. Also notice that because we solve this minimum cost flow problem by aggregating flows along the shortest paths from sensor nodes to the sink, it will not generate cyclic flows.

Let the optimal objective function value to the lth subproblem be z_l^*. Note that $z_l^* \leq 1$ since any column corresponding to a basic variable in the master problem has zero reduced cost, and the components of such a column form a feasible solution to (2–27)–(2–29) with an objective function value of 1. If $z_l^* < 1$ for some $l \in L$, then the reduced cost of the generated column is positive, and we add the corresponding column to K_l and re-solve the master problem with updated sets $K_l,\ l \in L$. If $z_l^* = 1, \forall l \in L$, then the optimal solution to the master problem is also optimal to the original problem (2–9)–(2–13).

2.4 Maximizing Lifetime in Delay-Tolerant Applications

This section describes the DT-MSM given in [26] and how column generation can be used to solve it. Recall that in this case, each node can defer the transmission of data until the sink is at the location most favorable for extending the network lifetime. When

a node is not transmitting the generated data to the sink, it stores the data for future transmission. Therefore, the net transmission rate at node i during the collection time could possibly differ from the constant data generation rate d_i.

Let D be the maximum tolerable delay, or the delay tolerance level. We assume that the sink completes one round of visits to all stops in D time units, and then repeats with another round. Therefore, two consecutive visits to the same stop takes D time units. This assumption is not restrictive when the sink's traveling times are negligible comparing to the lifetime of the network. Moreover, the order of visit to the locations is not relevant to the lifetime of the network; hence, we arbitrarily assume that the sink visits the locations in the order $1, 2, \ldots, |L|, 1, \ldots$. Note that the total stored data at each node i within a cycle of D time units is at most Dd_i.

For this problem, we define the network lifetime T to be the number of visiting cycles that the sink makes until the first node exhausts its energy. The actual lifetime is TD.

Two variations of the DT-MSM are proposed in [26] which correspond to two traffic buffering schemes. In the *sub-flow-based model*, the sensor nodes cannot store the incoming traffic from other nodes and must relay other nodes' traffic immediately. Therefore, each node can store only the self-generated traffic. On the other hand, in the *queue-based model*, sensor nodes can store the incoming traffic, as well as their own traffic, potentially building up a queue of mixed data at each node. In this chapter, we focus on the queue-based model and provide a column generation algorithm for it. We have also developed a column generation algorithm for the sub-flow-based model, but for brevity will omit its description.

The queue-based problem can be formulated as the following nonlinear program.

$$\max \; T \tag{2–30}$$

$$\text{s.t.} \quad \left[\sum_{l \in L} t_l \left(\sum_{j \in S_i^l} C_{ij}^t x_{ij}^l + \sum_{k : i \in S_k^l} \gamma x_{ki}^l \right) \right] T \leq E_i \qquad \forall i \in N \tag{2–31}$$

$$t_l \left(\sum_{k:i \in S_k^l} x_{ki}^l \right) + q_i^{l-1} - t_l \left(\sum_{j \in S_i^l} x_{ij}^l \right) = q_i^l \qquad \forall i \in N,\ l \in L \qquad (2\text{–}32)$$

$$q_i^0 = Dd_i \qquad \forall i \in N \qquad (2\text{–}33)$$

$$q_i^{|L|} = 0 \qquad \forall i \in N \qquad (2\text{–}34)$$

$$x_{ij}^l \geq 0 \qquad \forall i \in N,\ j \in S_i^l,\ l \in L \qquad (2\text{–}35)$$

$$q_i^l \geq 0 \qquad \forall i \in N,\ l \in L \qquad (2\text{–}36)$$

$$T \geq 0. \qquad (2\text{–}37)$$

The energy constraints (2–31) are similar to the mobile sink model, noting that the total time spent by the sink at location l is $t_l T$. They ensure that the energy consumed by each node cannot be more than its original energy endowment E_i. The flow conservation constraints are expressed through the queue length dynamics in (2–32). Here, variable q_i^l denotes the length of the queue at node $i \in N$ right after the sink departs from location l. Constraints (2–33) and (2–34) ensure that the data generated in the previous cycle will be transmitted to the sink during the current cycle.

Letting $u = 1/T$ and $y_{ij}^l = t_l x_{ij}^l$, the above problem can be written as the following linear programming problem:

$$\min\ u \qquad (2\text{–}38)$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} C_{ij}^t y_{ij}^l + \sum_{k:i \in S_k^l} \gamma y_{ki}^l \right) \leq E_i u \qquad \forall i \in N \qquad (2\text{–}39)$$

$$\sum_{k:i \in S_k^l} y_{ki}^l + q_i^{l-1} - \sum_{j \in S_i^l} y_{ij}^l = q_i^l \qquad \forall i \in N,\ l \in L \qquad (2\text{–}40)$$

$$q_i^0 = Dd_i \qquad \forall i \in N \qquad (2\text{–}41)$$

$$q_i^{|L|} = 0 \qquad \forall i \in N \qquad (2\text{–}42)$$

$$y_{ij}^l \geq 0 \qquad \forall i \in N,\ j \in S_i^l,\ l \in L \qquad (2\text{–}43)$$

$$q_i^l \geq 0 \qquad \forall i \in N,\ l \in L \qquad (2\text{–}44)$$

$$u \geq 0. \tag{2–45}$$

We again place the energy constraints (2–39) in the master problem and obtain columns from constraints (2–40)–(2–44). Similar to the previous model, problem (2–38)–(2–45) has an optimal solution in which the flow variables y_{ij}^l correspond to acyclic flow patterns. As a result, we will restrict ourselves to generate only such acyclic columns. Note also that all possible extreme rays of the polyhedral set (2–40)–(2–44) contain cyclic flows. Therefore, we can ignore the extreme rays of (2–40)–(2–44) in formulating the master problem. Denote by K the set of generated columns, where the components of the kth generated column are of the form $\overline{y}_{ij}^{lk}$ and $\overline{q}_i^{lk}$. We can write the following master problem:

$$\min\ u \tag{2–46}$$

$$\text{s.t.} \sum_{k\in K}\sum_{l\in L}\left(\sum_{j\in S_i^l} C_{ij}^t \overline{y}_{ij}^{lk} + \sum_{k:i\in S_k^l} \gamma \overline{y}_{ki}^{lk}\right) r_k \leq E_i u \qquad \forall i \in N \tag{2–47}$$

$$\sum_{k\in K} r_k = 1 \tag{2–48}$$

$$r_k \geq 0 \qquad \forall k \in K \tag{2–49}$$

$$u \geq 0. \tag{2–50}$$

Now, suppose that $-\delta_i$ and π are the dual variables associated with constraints (2–47) and (2–48), respectively. The reduced cost for a new column is:

$$\sum_{l\in L}\sum_{i\in N}\left(\sum_{j\in S_i^l} \delta_i C_{ij}^t y_{ij}^l + \sum_{k:i\in S_k^l} \delta_i \gamma y_{ki}^l\right) - \pi. \tag{2–51}$$

Therefore, the subproblem is written as follows.

$$\min \sum_{l\in L}\sum_{i\in N}\left(\sum_{j\in S_i^l} \delta_i C_{ij}^t y_{ij}^l + \sum_{k:i\in S_k^l} \delta_i \gamma y_{ki}^l\right) \tag{2–52}$$

$$\text{s.t.} \sum_{k:i\in S_k^l} y_{ki}^l + q_i^{l-1} - \sum_{j\in S_i^l} y_{ij}^l = q_i^l \qquad \forall i \in N,\ \forall l \in L \qquad (2\text{–}53)$$

$$q_i^0 = Dd_i \qquad \forall i \in N \qquad (2\text{–}54)$$

$$q_i^{|L|} = 0 \qquad \forall i \in N \qquad (2\text{–}55)$$

$$y_{ij}^l \geq 0 \qquad \forall i \in N,\ j \in S_i^l,\ l \in L \qquad (2\text{–}56)$$

$$q_i^l \geq 0 \qquad \forall i \in N,\ l \in L. \qquad (2\text{–}57)$$

Let z^* be the optimal objective function value of the above subproblem. If $z^* > \pi$, we add another column (extreme point) to set of columns K and re-solve the master problem. Otherwise, the solution to the master problem is optimal to the problem (2–38)–(2–45) because the reduced cost for all columns must be non-positive.

Subproblem (2–52)–(2–57) is a minimum cost flow problem on a special network. Figure 2-1 illustrates how we can build this new network from $|L|$ copies of the original network.

- There are $|N| \cdot |L| + 1$ nodes divided into $|L|$ layers, plus one node representing the sink.
- There is exactly one node corresponding to each sensor $i \in N$ in each of the $|L|$ layers.
- There is an arc from sensor node i to sensor node j in all $|L|$ layers of the network if and only if there is an arc from i to j in the original network.
- If there is an arc from a sensor i to the sink at location $l \in L$, then there is an arc from the corresponding node i in the lth layer to the sink node in the new network.
- For each sensor $i \in N$ and each layer $l \in L$, there is an arc from the node representing i in layer l to the corresponding node (representing i) in layer $l + 1$. We denote these as *queue arcs*.
- The cost of each queue arc is zero. The cost of all other arcs is of the form of $\delta_i C_{ij}^t + \delta_j \gamma$. Note that $\delta_i \geq 0, \forall i \in N$. Hence, each cost is non-negative.
- Node i in the first layer has an excess of Dd_i and the sink has a deficit of $D\sum_i d_i$. All other nodes have no flow excess or deficit.
- All arcs have unlimited capacity.

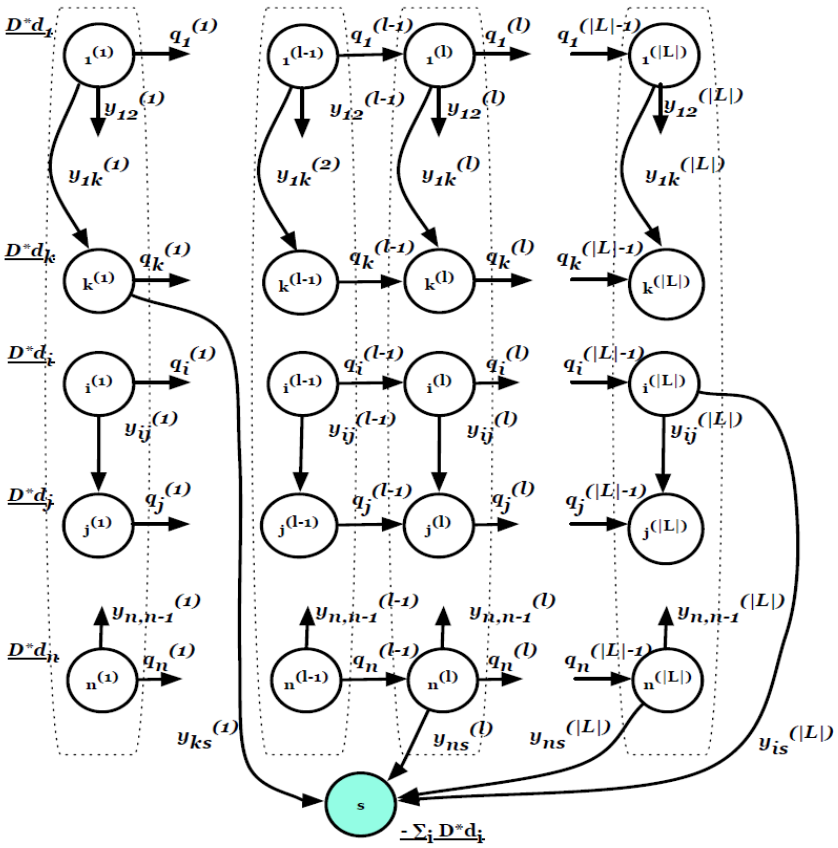


Figure 2-1. Illustration of queue-based delay-tolerant mobile sink model

Similar to the previous case, here we also have a minimum cost flow problem on an uncapacitated network and therefore, the subproblem reduces to finding a shortest path from each of the nodes in the first layer to the sink. This is because each node sends all its excess flow using the shortest path from that node to the sink. Therefore, the subproblem can be solved by one application of Dijkstra's algorithm.

When using the energy consumption model defined by (2–1)–(2–3), the above subproblem becomes even easier. Note that the cost of any arc between two sensor nodes is the same in all $|L|$ layers of the network since the sensor nodes do not move. The only difference between the layers is the arcs from the sensor nodes to the sink. The cost of such arcs can differ between the layers. Therefore, there is always an optimal solution to the above subproblem using only the arcs in the first layer, followed by a (possibly empty) succession of the queue arcs and ending with an arc to the sink

Table 2-1. Experimental parameters and their values

Number of sensor nodes	$\{60, 75, 80, 100, 120\}$
Number of possible sink locations	$\{20, 25, 30, 35, 40\}$
path loss exponent (e)	$\{4.0\}$
maximum transmission range	$\{6.5, 7.5, 8.0\}$
Data generation rate (d_i)	0.05 bps
$\alpha = \gamma$	10 pJ/bit/m^2
β	0.0013 pJ/bit/m^4
Initial Energy (E_i)	500 J

(possibly in another layer). Therefore, for each sensor node $i \in N$, we can add an arc (i, s) with the minimum cost over all layers (if any) to the first layer and solve the shortest path problem on this smaller network. The optimal solution to the expanded network can be easily retrieved by keeping track of the associated layer for each added arc.

2.5 Computational Results

We compare the computational efficiency of the algorithms that we proposed in this chapter against the direct solution of LP formulations. For each particular instance size, we randomly generate several networks and compare the results of running the two algorithms on these networks. All algorithms are implemented in C++ calling CPLEX version 11.0 via ILOG Concert Technology 2.5 to solve the linear programming problems, and compiled using GNU g++ version 4.1.2. All experiments are performed on a Dell Power Edge 2600 machine with two Pentium4 3.2Ghz/1M cache processors and 6 gigabytes of memory, running Redhat version 5.

In Table 2-1, we show the system parameters and their values used to generate the configuration for the experiments. The last three lines in the table are from [33]. The positions of the sensor nodes (N) and possible sink stops (L) are randomly generated according to a uniform distribution over a circular area with radius of 25. The maximum transmission range is carefully chosen so that resulting graph is strongly connected with high probability.

Table 2-2 shows the average execution times (in CPU seconds) for solving both the column generation algorithm and the original linear program (using CPLEX) for the

Table 2-2. Performance of the two algorithms for the mobile sink model

Instance Size	Column Generation		LP	Comparison
$(\|N\|, \|L\|)$	Time (s)	Number of Columns	Time (s)	C-G Wins
(60,20)	0.27	210	0.49	96%
(70,25)	0.54	317	1.33	100%
(80,30)	1.62	640	4.24	100%
(100,35)	2.92	737	7.59	100%
(120,40)	9.17	1456	26.66	100%

MSM. For each problem size, we generate 25 random network instances and solve them using both methods. The second column (Time (s)) reports the average time for the column generation algorithm based on the 25 instances, while the third column (Number of Columns) represents the average number of columns generated for these instances. The fourth column represents the average time of solving the 25 linear programs directly with the primal simplex solver of CPLEX. We also report in the fifth column (C-G Wins) the percentage of all the instances for which the column generation performs better than CPLEX in terms of CPU time. It is clear from these results that for the MSM, the column generation algorithm outperforms CPLEX in terms of average execution time, and also for the majority of the individual instances that are solved.

Table 2-3 contains a comparison between the performance of the column generation algorithm for the MSM and four other LP solvers of CPLEX. These results are for the same instances based on which Table 2-2 is built. The second column provides the average execution time of the column generation algorithm. For each LP solver, we report the average solution time of the 25 instances as well as the fraction of all the instances where the column generation algorithm solves the problem faster than the corresponding CPLEX solver. We can conclude from these results that the column generation algorithms outperforms all the implemented LP algorithms in CPLEX for smaller instances. As the instance size grows, the performance gap between the barrier optimizer of CPLEX and the column generation algorithm becomes smaller.

Table 2-3. Comparing the results of the column generation algorithm for the mobile sink model with different linear programming solvers of CPLEX

	C-G	Dual		Barrier		Sifting		Concurrent	
$(\|N\|, \|L\|)$	Time	Time	C-G Wins	Time	C-G Wins	Time	C-G Wins	Time	C-G Wins
(60,20)	0.27	1.56	100%	0.74	96%	5.17	100%	1.57	100%
(70,25)	0.54	5.37	100%	1.27	96%	10.05	100%	5.40	100%
(80,30)	1.62	19.43	100%	2.46	88%	19.42	100%	19.37	100%
(100,35)	2.92	34.59	100%	4.09	80%	32.33	100%	33.92	100%
(120,40)	9.17	140.96	100%	8.57	52%	56.86	100%	142.08	100%

Table 2-4. Comparing the column generation and the subgradient algorithms for the mobile sink model

	Column Generation		Subgradient	
Problem	Time (s)	Number of Columns	Time (s)	Number of Iterations
TP1	0.01	54	3.43	47689
TP2	0.01	67	> 1000	> 13450599
TP3	0^+	36	74.88	1026989
TP4	0.01	53	388.96	5094048
TP5	0.03	75	683.63	8937513
TP6	0.01	5	0.21	3254
TP7	0.01	26	> 1000	> 13633391
TP8	0.01	62	485.13	6419664
TP9	0.02	65	> 1000	> 13454004
TP10	0.02	79	726.8	9918074

Table 2-4 provides a comparison between the performance of our column generation algorithm for the MSM and a variation of the subgradient algorithm presented in [9]. The reader is referred to [9] for details on the distributed implementation of the corresponding subgradient algorithm. Our implementation is the same as that of [9] with some minor modifications. More specifically, we solve the minimum cost flow subproblems using Dijkstra's algorithm instead of the scaled ϵ-relaxation algorithm that the authors use in [9]. Also, we use random starting values for the dual variables since the choice of the starting point had little impact on the convergence of the algorithm in our experiments. For the step length we use $s^k = \frac{1}{k}$, as suggested by the authors. Ten instances of size $|N| = 12$ and $|L| = 3$ have been generated and solved using both algorithms. In generating these sparse instances, we set the maximum transmission range to 35 to ensure connectivity. For the column generation algorithm, we report the running time and the number of columns generated. For the subgradient algorithm, we report the time until the first solution with less than 5% optimality violation is generated as well as the number of iterations required until such a solution is generated. A 1000-second time limit is imposed for each run. It is clear from these results that the column generation algorithm is much faster than our implementation of the subgradient algorithm.

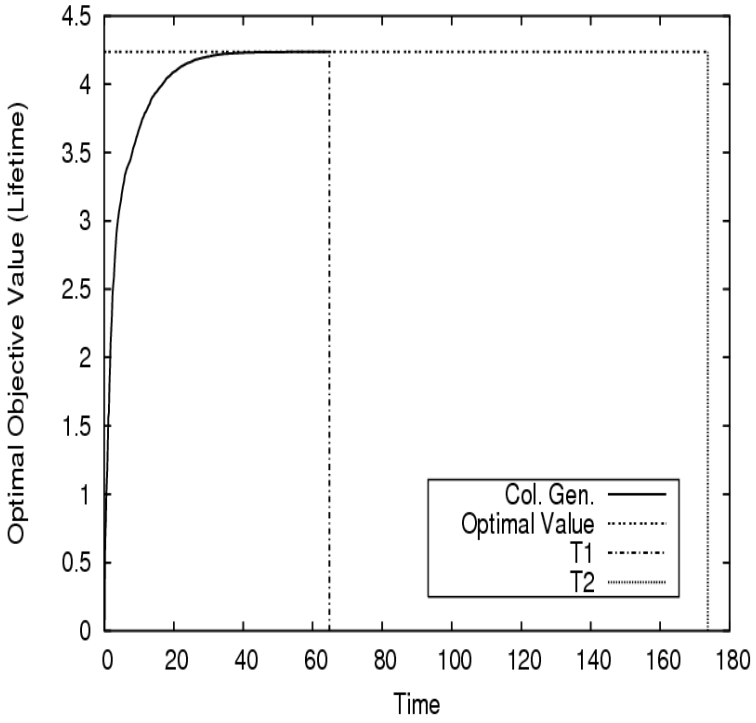


Figure 2-2. Convergence of column generation algorithm to the optimal solution with $|N| = 120$ and $|L| = 40$

In Figure 2-2, we illustrate the convergence of the column generation method to the optimal objective value on a random MSM instance with $|N| = 120$ and $|L| = 40$. The vertical line T1 represents the time when our column generation algorithm terminates, and the vertical line T2 represents the time at which the LP terminates. The graph shows that the intermediate solutions of the master problem in the column generation algorithm rapidly approach the optimal value. Therefore, in the case that a near optimal solution is sufficient for the application, the column generation algorithm may find one such solution quickly.

Table 2-5 contains the results of running the column generation algorithm for the DT-MSM versus solving the original linear program with the primal simplex solver of CPLEX. These results are for the same instances as those in the MSM case, with the maximum tolerable delay being $D = 5000$ for all networks. The results suggest that

Table 2-5. Performance of the two algorithms for the delay-tolerant mobile sink model

Instance Size	Column Generation		LP	Comparison
$(\|N\|, \|L\|)$	Time (s)	Number of Columns	Time (s)	C-G Wins
(60,20)	0.23	35	0.30	76%
(70,25)	0.56	37	0.67	60%
(80,30)	1.58	56	1.11	60%
(100,35)	4.67	63	2.42	40%
(120,40)	19.10	108	6.45	28%

the column generation algorithm works better on the smaller DT-MSM instances, while solving the original LP is generally faster for instances of larger size. However, as we mentioned before, one might still prefer the column generation algorithm since the subproblem can be built into network protocols and be solved in a decentralized manner. Table 2-6 compares the performance of the column generation algorithm with four other LP algorithms implemented in CPLEX, and verifies that while column generation is the most effective algorithm on smaller instances, the CPLEX primal simplex solver is the most efficient algorithm on larger instances.

Table 2-6. Comparing the results of the column generation algorithm for the delay-tolerant mobile sink model with different linear programming solvers of CPLEX

	C-G	Dual		Barrier		Sifting		Concurrent	
$(\|N\|, \|L\|)$	Time	Time	C-G Wins	Time	C-G Wins	Time	C-G Wins	Time	C-G Wins
(60,20)	0.23	0.68	84%	0.61	96%	2.19	100%	0.69	88%
(70,25)	0.56	2.25	96%	1.09	96%	4.75	100%	2.23	96%
(80,30)	1.58	1.78	60%	1.92	72%	8.49	100%	1.80	60%
(100,35)	4.67	5.99	72%	3.35	56%	15.70	96%	6.03	72%
(120,40)	19.10	29.79	72%	7.34	32%	41.36	88%	30.43	72%

CHAPTER 3
A DISTRIBUTED ALGORITHM FOR THE LIFETIME MAXIMIZATION PROBLEM IN DELAY-TOLERANT APPLICATIONS

3.1 Motivation and Literature Survey

The lifetime of a wireless sensor network (WSN) is often defined as the time until the first sensor node fails because of energy depletion [5, 6]. Due to the multi-hop data routing from sensor nodes to the sink, the sensor nodes that are closer to the sink are usually the most burdened with relaying data from distant nodes. This traffic imbalance can cause early energy depletion for nodes near the sink, which creates an "energy hole" around the sink. The result may be an early disconnection of the sink from the sensor nodes that may still have plenty of energy [20, 21, 34].

Recently, exploiting mobility (particularly the sink's mobility) to improve network lifetime has attracted the interest of researchers [7–10, 15–19, 26]. As explained in Chapter 2, a mobile sink can better balance the traffic load across the sensor field by making stops at different locations in the network to receive data from sensor nodes. Therefore, sink mobility can mitigate the energy-hole problem and increase the network lifetime.

In Chapter 2, we presented a column generation algorithm for the Delay Tolerant Mobile Sink Model (DT-MSM). DT-MSM is suitable for applications where some amount of delay in data delivery to the sink is permitted [16]. In this setting, sensor nodes may delay the transmission of the collected data for a while and wait until the sink arrives at a location that is most favorable for improving the network lifetime.

The goal of this chapter is to find a distributed algorithm that solves the lifetime maximization problem associated with DT-MSM. The decisions to be made include how long the sink should stay at each potential stop, and how to route the data to the sink when it stops. We present two main contributions in this chapter. First, our algorithm is both distributed and mostly local. The overall solution is decomposed into smaller decision problems and each decision can be done locally in a sensor node. For the

most part, only local information at a node itself and at its neighbors is needed. In general, distributed algorithms are more useful for networking problems because they can be readily built into network protocols and become network control algorithms. Local algorithms have the additional benefit of restricting the control traffic to be among locally interacting nodes. Second, we analyze our algorithm and show that in the long-run, it converges to the optimal objective function value of the lifetime maximization problem, and that the virtual queue sizes are bounded in the long-run average sense.

We next briefly survey related works. Gatzianas and Georgiadis [9] study the lifetime maximization problem in a WSN with a mobile sink and propose a distributed subgradient algorithm for solving it. Unlike their algorithm, standard convergence results for subgradient algorithms do not apply to our algorithm and the analysis about convergence and algorithm performance in our case relies on a different framework. In addition, their approach relaxes the energy constraints, whereas we relax the flow conservation constraints.

Our convergence and stability analysis is based on analyzing a Lyapunov drift. The Lyapunov drift technique is widely used for studying the stability of control and optimization algorithms for networks of queues. A representative work is [35] where the authors apply this technique to the study of a link scheduling algorithm in multi-hop packet radio networks. Our method is closely related to that of [36], which also analyzes a dynamic control algorithm in wireless networks.

Similar to our assumptions in Chapter 2, we assume that the sink's travel times between different stops are negligible in this chapter. This assumption is widely used in the literature of similar mobile sink problems [7, 8, 10, 17–19, 26]. Moreover, we restrict the set of possible sink locations (where the sink can stop at) to a given finite set.

The rest of this chapter is organized as follows. Section 3.2 describes the underlying system model and the problem formulation. The distributed algorithm for

the problem is discussed in Section 3.3. In Section 3.4, we analyze the optimality and convergence of our algorithm and Section 3.5 contains the experimental results.

3.2 Problem Formulation

We model a wireless sensor network as a directed graph $G^0 = (\mathcal{N} \cup \mathcal{L}, \mathcal{A})$, where $\mathcal{N} = \{1, \dots, N\}$ is the set of sensor nodes, $\mathcal{L} = \{1, \dots, L\}$ is the set of the sink locations, and $\mathcal{A}$ is the set of wireless links interconnecting the sensor nodes and the sink locations. Let τ_{ij} be the Euclidean distance between nodes i and j. We define the set of downstream neighbors of i as $N(i) = \{j \in \mathcal{N} \cup \mathcal{L} | (i,j) \in \mathcal{A}\} = \{j \in \mathcal{N} \cup \mathcal{L} |\ \tau_{ij} \leq \bar{\tau}\}$, where $\bar{\tau}$ is the maximum transmission range of the nodes in the network. Each sensor node i generates data at a constant rate d_i and has an initial energy endowment E_i. Let $c : \mathcal{A} \rightarrow R^+$ be a cost function on the arc set such that $c(i,j)$ defines the required energy for the transmission of one unit of data along arc $(i,j) \in \mathcal{A}$.

An important parameter in DT-MSM is the maximum delay D that the underlying application can tolerate. That is, the sensors can locally store and delay the transmission of their collected data for up to D time units. We also assume that the sink must complete one of its tours within D time units (see [26]). Therefore, maximizing network lifetime is equivalent to maximizing the total number of sink tours, denoted by T. Throughout this chapter, we will assume that the order in which the sink visits the sink locations is given as $1 \rightarrow 2 \rightarrow \cdots \rightarrow L$.

Decision variables are the amount of time that the sink stays at each sink location $l \in \mathcal{L}$ within every tour (denoted by t_l) and the data transmission rate from i to j when the sink is at l (denoted by $a_{ij}^{(l)}$). We define $x_{ij}^{(l)} = t_l a_{ij}^{(l)}$ as the traffic volume on link (i,j) when the sink is at l. Let $G^l = (\mathcal{N} \cup \{l\}, \mathcal{A}^l)$ be the underlying graph when the sink is at l, where $\mathcal{A}^l = \{(i,j) \in \mathcal{A} | i \in \mathcal{N}, j \in \mathcal{N} \cup \{l\}\}$. When the sink is at location l, the neighborhood set of i is denoted by $N_l(i) = \{j | (i,j) \in \mathcal{A}^l\}$. We assume that for each sensor node $i \in \mathcal{N}$ and every sink location $l \in \mathcal{L}$, there exists at least one path from i to l

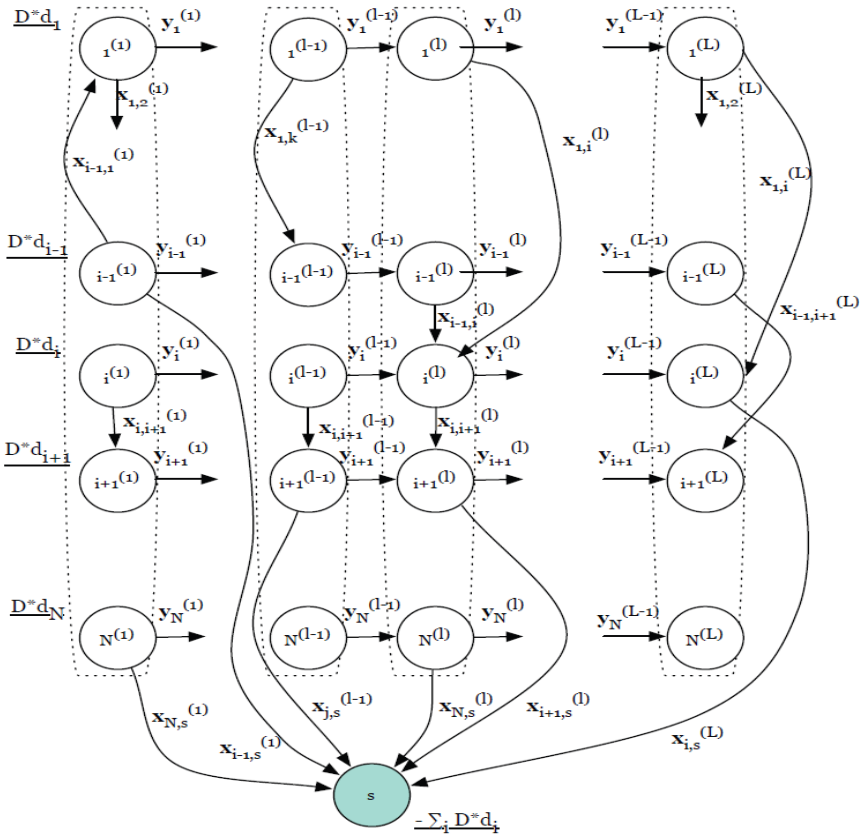


Figure 3-1. Expanded graph of delay-tolerant mobile sink model

in G^l. This assumption is required to ensure the possibility of delivering data from every sensor node to the sink at any point in time.

Similar to the construction in Section 2.4, we create an expanded graph from the graphs G^l, $l \in \mathcal{L}$, such that the maximum network lifetime can be obtained by solving a special network-flows problem on this expanded graph. Figure 3-1 illustrates this expanded graph. The cost of each vertical arc $(i^{(l)}, j^{(l)})$ is assigned as follows:

$$e_{ij}^{(l)} = \begin{cases} c(i,j), & \text{if} \quad i,j \in \mathcal{N} \\ c(i,l), & \text{if} \quad i \in \mathcal{N}, j = l \\ \infty, & \text{otherwise.} \end{cases} \tag{3–1}$$

The cost of each horizontal arc $(i^{(l)}, i^{(l+1)})$ is 0. The flow conservation constraints for the sensor nodes can be stated as follows.

$$\begin{cases} \sum_{j:i\in N_1(j)} x_{ji}^{(1)} - \sum_{j\in N_1(i)} x_{ij}^{(1)} - y_i^{(1)} = -D\cdot d_i, \\ \qquad\qquad \text{for } l=1, i\in\mathcal{N} \\ \sum_{j:i\in N_l(j)} x_{ji}^{(l)} - \sum_{j\in N_l(i)} x_{ij}^{(l)} + y_i^{(l-1)} - y_i^{(l)} = 0, \\ \qquad\qquad \text{for } l\in\{2,\dots,L-1\}, i\in\mathcal{N} \\ \sum_{j:i\in N_L(j)} x_{ji}^{(L)} - \sum_{j\in N_L(i)} x_{ij}^{(L)} + y_i^{(L-1)} = 0, \\ \qquad\qquad \text{for } l=L, i\in\mathcal{N} \end{cases} \tag{3–2}$$

At the beginning of each cycle of length D time units, node i has accumulated Dd_i units of data from the previous cycle, which must be delivered to the sink during the current cycle. Variable $x_{ij}^{(l)}$ represents the amount of data sent on arc (i,j) when the sink is at location l, while $y_i^{(l)}$ denotes the amount of buffered data (queue size) at node i right after the sink leaves location l. Thus, $y_i^{(l-1)} - y_i^{(l)}$ is the change in the buffered data at node i while the sink is at location l. In addition, at the sink (node s), all arrival traffic must be drained. Thus, we must have

$$\sum_{l=1}^{L}\sum_{j:s\in N_l(j)} x_{js}^{(l)} - \sum_{i=1}^{N} Dd_i = 0. \tag{3–3}$$

The problem that we address in this chapter is to maximize the number of sink tours (or cycles), T, while maintaining the flow conservation (3–2) and (3–3), subject to the energy constraints at the sensor nodes. More precisely, the problem can be written as

$$\max\ T \tag{3–4}$$

$$\text{s.t. (3–2) and (3–3)} \tag{3–5}$$

$$\left(\sum_{l=1}^{L}\sum_{j\in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)}\right) T \le E_i, \quad \forall i \in \mathcal{N} \tag{3–6}$$

$$x_{ij}^{(l)} \ge 0, \quad \forall l \in \mathcal{L},\ i \in \mathcal{N},\ j \in N_l(i) \tag{3–7}$$

$$y_i^{(l)} \ge 0, \quad \forall l \in \mathcal{L},\ i \in \mathcal{N} \tag{3–8}$$

$$T \ge 0. \tag{3–9}$$

Constraints (3–6) ensure that the total energy expenditure at every sensor node during T cycles does not exceed the node's initial energy endowment.

3.3 A Lagrangian Relaxation Approach

In this section, we propose a distributed algorithm to solve the lifetime maximization problem described in Section 3.2. We start by presenting an equivalent linear program that is obtained from problem (3–4)–(3–9) by replacing $1/T$ with z. For convenience, we define $M = \sum_{i=1}^{N} Dd_i$.

$$\min z \tag{3–10}$$

$$\text{s.t.} \sum_{l=1}^{L}\sum_{j\in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)} \le z E_i, \quad \forall i \in \mathcal{N} \tag{3–11}$$

$$\begin{cases} \displaystyle\sum_{j:i\in N_l(j)} x_{ji}^{(l)} - \sum_{j\in N_l(i)} x_{ij}^{(l)} + y_i^{(l-1)} - y_i^{(l)} = 0, \\ \qquad\qquad \forall l \in \mathcal{L},\ i \in \mathcal{N} \\ \displaystyle\sum_{l=1}^{L}\sum_{j:s\in N_l(j)} x_{js}^{(l)} - M = 0 \end{cases} \tag{3–12}$$

$$y_i^{(0)} = Dd_i, \qquad y_i^{(L)} = 0, \quad \forall i \in \mathcal{N} \tag{3–13}$$

$$x_{ij}^{(l)} \ge 0, \quad \forall l \in \mathcal{L},\ i \in i \in \mathcal{N},\ j \in N_l(i) \tag{3–14}$$

$$y_i^{(l)} \ge 0, \quad \forall l \in \mathcal{L},\ i \in \mathcal{N} \tag{3–15}$$

$$z \ge 0. \tag{3–16}$$

Note that M is an upper bound on the traffic volume of any arc in the network (including buffered data). In the remainder of this chapter, we will use the terms *flow* and *volume* interchangeably. Hence, the new formulation minimizes the maximum energy consumption (normalized with respect to E_i) among all nodes in one cycle, while satisfying the flow conservation constraints.

Let $\pi_i^{(l)}$ and π_s be the dual variables associated with the constraints in (3–12). The Lagrangian function of (3–10) is

$$L(z,x,y,\pi) = z + \pi_s(\sum_{l=1}^{L}\sum_{j:s\in N_l(j)} x_{js}^{(l)} - M) + \sum_{l=1}^{L}\sum_{i=1}^{N}\pi_i^{(l)}(\sum_{j:i\in N_l(j)} x_{ji}^{(l)} - \sum_{j\in N_l(i)} x_{ij}^{(l)} + y_i^{(l-1)} - y_i^{(l)}),$$

where $\pi = (\pi_i^{(l)}, \pi_s)$. Rearranging the above equality, we obtain

$$\begin{aligned} L(z,x,y,\pi) = z + &\sum_{l=1}^{L}\sum_{(i,j)\in\mathcal{A}^l}(\pi_j^{(l)} - \pi_i^{(l)})x_{ij}^{(l)} + \\ &\sum_{i=1}^{N}\sum_{l=1}^{L-1}(\pi_i^{(l+1)} - \pi_i^{(l)})y_i^{(l)} - D\sum_{i=1}^{N}(\pi_s - \pi_i^{(1)})d_i. \end{aligned} \tag{3–17}$$

For convenience, we define $\pi_s^{(1)} = \pi_s^{(2)} = \ldots = \pi_s^{(L)} = \pi_s$ in the second term of equation (3–17). Note that the last term of (3–17) is a constant for a given π. Therefore, the Lagrangian dual function $\theta(\pi)$ is given by

$$\theta(\pi) = \min L(z,x,y,\pi) \tag{3–18}$$

$$\text{s.t.} \sum_{l=1}^{L}\sum_{j\in N_l(i)} x_{ij}^{(l)} e_{ij}^{(l)} - zE_i \le 0,\ \forall i \in \mathcal{N} \tag{3–19}$$

$$x_{ij}^{(l)} \ge 0,\ \forall l \in \mathcal{L},\ i \in \mathcal{N},\ j \in N_l(i) \tag{3–20}$$

$$0 \le y_i^{(l)} \le M,\ \forall l \in \{1,\ldots,L-1\},\ i \in \mathcal{N} \tag{3–21}$$

$$z \ge 0. \tag{3–22}$$

We can decompose the problem (3–18)–(3–22) into the following two subproblems.

$$S_1 : \min \sum_{i=1}^{N} \sum_{l=1}^{L-1} (\pi_i^{(l+1)} - \pi_i^{(l)}) y_i^{(l)}$$
$$\text{s.t.} \quad 0 \le y_i^{(l)} \le M, \quad \forall l \in \{1, \dots, L-1\},\ i \in \mathcal{N}. \tag{3–23}$$

$$S_2 : \min \left\{ z + \sum_{l=1}^{L} \sum_{(i,j)\in\mathcal{A}^l} (\pi_j^{(l)} - \pi_i^{(l)}) x_{ij}^{(l)} \right\}$$
$$\text{s.t.} \quad 0 \le x_{ij}^{(l)} \le M, \quad \forall l \in \mathcal{L},\ i \in \mathcal{N},\ j \in N_l(i)$$
$$\sum_{l=1}^{L} \sum_{j\in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)} - z E_i \le 0, \quad \forall i \in \mathcal{N}$$
$$z \ge 0. \tag{3–24}$$

3.3.1 Distributed Algorithms for Subproblems

The solution method for S_1 is straightforward: If $(\pi_i^{(l+1)} - \pi_i^{(l)})$ is negative, then we set the corresponding variable $y_i^{(l)}$ to its upper bound M. Otherwise, $y_i^{(l)}$ equals zero. Clearly, this algorithm can be implemented in a distributed and local manner.

We now turn our attention to subproblem S_2. Consider first the case in which z is fixed at some nonnegative value $\bar{z}$ and define $f(\bar{z})$ as follows.

$$f(\bar{z}) = \max \left\{ -\bar{z} + \sum_{i=1}^{N} \sum_{l=1}^{L} \sum_{j:j\in N_l(i)} (\pi_i^{(l)} - \pi_j^{(l)}) x_{ij}^{(l)} \right\}$$
$$\text{s.t.} \quad 0 \le x_{ij}^{(l)} \le M, \quad \forall l \in \mathcal{L},\ i \in \mathcal{N},\ j \in N_l(i)$$
$$\sum_{l=1}^{L} \sum_{j\in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)} \le \bar{z} E_i, \quad \forall i \in \mathcal{N}.$$

For $i \in \mathcal{N}$, let

$$f_i(\bar{z}) = \max \sum_{l=1}^{L} \sum_{j:j\in N_l(i)} (\pi_i^{(l)} - \pi_j^{(l)}) x_{ij}^{(l)}$$
$$\text{s.t.} \quad 0 \le x_{ij}^{(l)} \le M, \quad \forall l \in \mathcal{L},\ j \in N_l(i)$$

$$\sum_{l=1}^{L} \sum_{j \in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)} \leq \bar{z} E_i.$$

Then, $f(\bar{z}) = -\bar{z} + \sum_{i=1}^{N} f_i(\bar{z})$. Therefore, maximizing $f(\bar{z})$ can be decomposed into smaller problems in which each node i will find $f_i(\bar{z})$. Finding the value of $f_i(\bar{z})$ at each node i is equivalent to solving an instance of the *fractional knapsack problem*, which is polynomially solvable [37]. Details are listed in Algorithm 1. Again, this algorithm can also be implemented in a distributed and local manner.

Algorithm 1 Fractional Knapsack ($\bar{z}$) for Finding $f_i(\bar{z})$

```
sort (i, l, j) in the nonincreasing order of (π_i^(l) − π_j^(l))/e_ij^(l)
U ⇐ z̄E_i
for each of (i, l, j) in the sorted list do
  if (π_i^(l) − π_j^(l) ≤ 0) then
    break
  else if (U − Me_ij^(l)) < 0 then
    x_ij^(l) ⇐ U/e_ij^(l)
    break
  else
    x_ij^(l) ⇐ M
    U ⇐ U − Me_ij^(l)
  end if
end for
```

Algorithm 1 solves subproblem S_2 for a given value of z. To complete our argument, we need to suggest a way of finding an optimal value for z, so that the overall objective function $f(z)$ is maximized. Note that each $f_i(z)$ is a nondecreasing, concave, and piecewise-linear function of z. Hence, $f(z)$ is also concave and piecewise-linear. Suppose we want to search for an optimal solution by starting from 0 and incrementally increasing z until we find an optimal value. Note that we only need to consider z-values that mark the beginning or end of a linear segment of $f(z)$. Let z^* be the first optimal solution encountered in the search. Then, $f(z)$ must be increasing for $z \leq z^*$ (except for the trivial case where $f(z)$ equals zero at optimality). For $z \geq z^*$, $f(z)$ must be nonincreasing. Hence, we are looking for a value $z^* \geq 0$ such that $f'(z) > 0$ for $z < z^*$,

and $f'(z) \leq 0$ for $z > z^*$. Alternatively,

$$\begin{cases} \sum_{i \in \mathcal{N}} f_i'(z) > 1, & z < z^* \\ \sum_{i \in \mathcal{N}} f_i'(z) \leq 1, & z > z^*. \end{cases} \tag{3–25}$$

Also note that $f'(z)$ changes only when one of the $f_i'(z)$ changes. From Algorithm 1, we observe that for each i, $f_i'(z)$ changes only when we select the most profitable item in the list of the remaining items. Suppose item (i, j, l) is selected. The new $f_i'(z)$ is given by $f_i'(z) = (\pi_i^{(l)} - \pi_j^{(l)})/e_{ij}^{(l)}$. Furthermore, the next time when $f_i'(z)$ changes again is when z is incremented by $Me_{ij}^{(l)}/E_i$.

Algorithm 2 Solution for S_2

```
for each i ∈ N do
  sort (i, l, j) in nonincreasing order of (π_i^(l) − π_j^(l))/e_ij^(l)
  discard any item (i, l, j) if (π_i^(l) − π_j^(l)) ≤ 0
  k ⇐ 0; z_k ⇐ 0
  for each of (i, l, j) in the sorted list do
    z_k ⇐ z_k + Me_ij^(l)/E_i
    P_i[k] ⇐ (z_k, (π_i^(l) − π_j^(l))/e_ij^(l))
    k ⇐ k + 1
  end for
end for
find z* which satisfies (3–25) by searching (P_i)_{i∈N}
each node i applies Algorithm 1 with z*
```

To summarize, the procedure for searching z^* is to keep track of the sequence of points where $f'(z)$ changes (or the sequence of points where $f_i'(z)$ changes, for each i). For a fixed i, suppose that $(\pi_i^{(l)} - \pi_j^{(l)})/e_{ij}^{(l)}$ is sorted in nonincreasing order where any item (i, l, j) for which $(\pi_i^{(l)} - \pi_j^{(l)}) \leq 0$ is discarded. Starting with $z_0 = 0$, we can generate a sequence $z_k = z_{k-1} + Me_{ij}^{(l)}/E_i$ iteratively, where (i, j, l) used in the update to get z_k is the k^{th} item in the list. Then, $f_i'(z)$ can change only at each of the points z_k. Algorithm 2 formalizes this idea. The array $P_i[\,]$ records the sequence of z_k and $f_i'(z_k)$.

3.3.2 The Main Algorithm

We now assume that the network operates in a time-slotted fashion. The Lagrangian dual problem of (3–10)–(3–16) is

$$\text{Dual:} \quad \max_{\pi} \ \theta(\pi) \tag{3–26}$$

Hence, one can propose a subgradient projection method for (3–26) where updated π-values at each iteration can be obtained using the following equations.

$$\pi_i^{(l)}(k+1) = [\pi_i^{(l)}(k) - \delta(\sum_{j \in N_l(i)} x_{ij}^{(l)}(k) + y_i^{(l)}(k) - \sum_{j:i \in N_l(j)} x_{ji}^{(l)}(k) - y_i^{(l-1)}(k))]^+, \quad \forall i \in \mathcal{N}, l \in \mathcal{L} \tag{3–27}$$

$$\pi_s(k+1) = [\pi_s(k) - \delta(M - \sum_{l=1}^{L} \sum_{j:(j,s) \in \mathcal{A}^l} x_{js}^{(l)}(k))]^+. \tag{3–28}$$

Here, $[b]^+ = \max\{0, b\}$ and $\delta(> 0)$ is a sufficiently small step size. Letting $\delta q_i^{(l)}(k) = \pi_i^{(l)}(k)$, we propose the following modification of the above algorithm for the lifetime maximization problem given by (3–10)–(3–16). For technical reasons, here we have changed the upper bound of the flow variables from M to $M(\epsilon) \triangleq M + NL\epsilon$, where ϵ is a small positive value.

Main Algorithm

$$y(k) = \underset{\substack{y_i^{(l)} \in [0, M(\epsilon)], \\ l \in \{1,\ldots,L-1\}, \ i \in \mathcal{N}}}{\arg\min} \{\sum_{i=1}^{N} \sum_{l=1}^{L-1} (q_i^{(l+1)}(k) - q_i^{(l)}(k)) y_i^{(l)}\} \tag{3–29}$$

$$(z(k), x(k)) = \underset{\substack{x_{ij}^{(l)} \in [0, M(\epsilon)], \ l \in \mathcal{L}, \ i \in \mathcal{N}, \ j \in N_l(i) \\ \sum_{l=1}^{L} \sum_{j \in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)} \le z E_i, \ i \in \mathcal{N} \\ z \ge 0}}{\arg\min} \{\frac{z}{\delta} + \sum_{l=1}^{L} \sum_{i \in \mathcal{N}} \sum_{j \in N_l(i)} (q_j^{(l)}(k) - q_i^{(l)}(k)) x_{ij}^{(l)}\} \tag{3–30}$$

$$q_i^{(l)}(k+1) = [q_i^{(l)}(k) - (\sum_{j \in N_l(i)} x_{ij}^{(l)}(k) + y_i^{(l)}(k) -$$

$$\sum_{j:i \in N_l(j)} x_{ji}^{(l)}(k) - y_i^{(l-1)}(k))]^+, \ \forall l \in \mathcal{L}, \ i \in \mathcal{N} \tag{3–31}$$

$$q_s(k+1) = 0. \tag{3–32}$$

Problems (3–29) and (3–30) are solved using the algorithms presented in Section 3.3.1 with a suitable modification in the notation. One can think of $q_i^{(l)}$ as the virtual queue at node $i^{(l)}$ in Figure 3-1, and (3–31) can be understood as the queue dynamics in the two consecutive slots. Since $q_s^{(k+1)} = 0$, all flow reaching the sink must be drained out. Note that the above algorithm is not exactly a subgradient optimization one because of the projection that we use in updating q-values and also because $q_s(k+1) = 0$. Therefore, standard subgradient optimization results cannot be applied to prove the correctness of our algorithm. We will use a Lyapunov drift analysis method in the next section to show convergence and stability properties of this algorithm.

3.4 Convergence Results

In this section, we show that our algorithm converges to an optimal solution and the virtual queues are bounded (both in the long-run average sense). The analytical technique that we use is similar to that of [36].

We first define an ϵ-perturbed problem, which will be used later. Here, ϵ is a small positive value.

$$\min z \tag{3–33}$$

$$\text{s.t.} \sum_{l=1}^{L} \sum_{j \in N_l(i)} e_{ij}^{(l)} x_{ij}^{(l)} \leq z E_i, \quad \forall i \in \mathcal{N} \tag{3–34}$$

$$\begin{cases} \sum_{j:i\in N_l(j)} x_{ji}^{(l)} - \sum_{j\in N_l(i)} x_{ij}^{(l)} + y_i^{(l-1)} - y_i^{(l)} = -\epsilon, \\ \qquad\qquad \forall l \in \mathcal{L}; i \in \mathcal{N} \\ \sum_{l=1}^{L} \sum_{j:s\in N_l(j)} x_{js}^{(l)} = M + NL\epsilon \end{cases} \tag{3–35}$$

$$y_i^{(0)} = Dd_i, \qquad y_i^{(L)} = 0, \quad \forall i \in \mathcal{N}. \tag{3–36}$$

The usual nonnegativity constraints are still required. In the above problem, we inject an extra supply of magnitude ϵ at each node $i^{(l)}, i \in \mathcal{N},\ l \in \mathcal{L}$. The demand at the destination node s is also modified to $M + NL\epsilon$ to ensure the existence of a feasible flow.

We first discuss the properties of the optimal objective value function of the above ϵ-perturbed problem. For brevity, we replace our problem with a general linear programming problem as follows.

$$\begin{aligned} (P)\ \ \min\ & c^T x \\ \text{s.t. } & Ax = d \\ & x \geq 0. \end{aligned} \tag{3–37}$$

In problem $(P(\epsilon))$ below, the right hand side of (P) is perturbed by ϵ along the direction Δd.

$$\begin{aligned} (P(\epsilon))\ \ \min\ & c^T x \\ \text{s.t. } & Ax = d + \epsilon\Delta d \\ & x \geq 0. \end{aligned} \tag{3–38}$$

Let f^* and $f^*(\epsilon)$ be the optimal objective values for problem (P) and $(P(\epsilon))$, respectively.

Lemma 3.1. *$f^*(\epsilon)$ is continuous, convex, and piecewise-linear in ϵ.*

Proof. The dual of $(P(\epsilon))$ is defined as

$$(D(\epsilon)) \quad \max\ (d+\epsilon\Delta d)^T\pi$$
$$\text{s.t. } A^T\pi \le c$$

Let $\mathcal{S}$ denote the feasible region of $(D(\epsilon))$. By the strong duality theorem in linear programming [32], we know that $f^*(\epsilon) = \max\{(d+\epsilon\Delta d)^T\pi | \pi \in \mathcal{S}\}$. Also, we know that for every ϵ, the optimal objective function value of $(D(\epsilon))$ can be obtained by examining all extreme points of $\mathcal{S}$. Let $\hat{\mathcal{S}}$ be the set of extreme points of $\mathcal{S}$. Hence, $f^*(\epsilon) = \max\{d^T\pi + \epsilon\Delta d^T\pi | \pi \in \hat{\mathcal{S}}\}$. This means that $f^*(\epsilon)$ is obtained by taking the maximum of a finite number of linear functions of ϵ. Therefore $f^*(\epsilon)$ is a continuous, convex, and piecewise-linear function of ϵ [38]. □

Theorem 3.1. *Let $z^*(\epsilon)$ be the optimal objective function value of problem (3–33)–(3–36) and z^* be the optimal objective function value of the unperturbed problem. Then, $z^*(\epsilon) \to z^*$ as $\epsilon \to 0$.*

Proof. If $\epsilon = 0$, then the ϵ-perturbed problem is identical to the unperturbed problem defined as in (3–10)–(3–16). Therefore, the result holds using Lemma 3.1 and the fact that $z^*(\epsilon)$ is a continuous function of ϵ. □

Next, we want to prove our algorithm converges to the optimal objective function value in the time average sense. Let us define a Lyapunov function of the queues by $V(q) = \sum_{i\in\mathcal{N}}\sum_{l\in\mathcal{L}}(q_i^{(l)})^2$. Let $\Delta(k) \triangleq V(q(k+1)) - V(q(k))$.

Lemma 3.2. *There exist positive B, ϵ, and δ such that for all time slot k and for all $q(k)$ the following condition holds.*

$$\Delta(k) + \frac{2}{\delta}z(k) \le B + \frac{2}{\delta}\hat{z}(\epsilon) - 2\epsilon\sum_{l\in\mathcal{L}}\sum_{i\in\mathcal{N}} q_i^{(l)}, \tag{3–39}$$

where $\hat{z}(\epsilon)$ is the optimal solution of the ϵ-perturbed problem.

Proof. By squaring (3–31) and arranging it, we get

$$(q_i^{(l)}(k+1))^2 - (q_i^{(l)}(k))^2 \leq g^2(i,l,k) - 2q_i^{(l)}(k)g(i,l,k),$$

where $g(i,l,k) = \sum_{j\in N_l(i)} x_{ij}^{(l)}(k) - \sum_{j:i\in N_l(j)} x_{ji}^{(l)}(k) + y_i^{(l)}(k) - y_i^{(l-1)}(k)$. Note that $g(i,l,k) \leq NM$ because $\sum_{j\in N_l(i)} x_{ij}^{(l)}(k) \leq (N-1)M$ and $y_i^{(l)}(k) \leq M$ for all k. Aggregating the above inequality over all $i \in \mathcal{N}$ and $l \in \mathcal{L}$, we have

$$\begin{aligned}
&\sum_{l=1}^{L}\sum_{i=1}^{N}\left[\left(q_i^{(l)}(k+1)\right)^2 - \left(q_i^{(l)}(k)\right)^2\right] \\
\leq&\sum_{l=1}^{L}\sum_{i=1}^{N} g^2(i,l,k) - 2\sum_{l=1}^{L}\sum_{i=1}^{N} q_i^{(l)}(k)g(i,l,k) \\
\leq&LN^3M^2 + 2\sum_{l=1}^{L}\sum_{i=1}^{N} q_i^{(l)}(k)\left(-\sum_{j\in N_l(i)} x_{ij}^{(l)}(k) + \right. \\
&\left.\sum_{j:i\in N_l(j)} x_{ji}^{(l)}(k) - y_i^{(l)}(k) + y_i^{(l-1)}(k)\right) \\
=&B - 2\sum_{l=1}^{L}\sum_{(j,s)\in\mathcal{A}^l} q_j^{(l)}(k)x_{js}^{(l)}(k) \\
&+ 2\sum_{l=1}^{L}\sum_{(i,j)\in\mathcal{A}^l; j\neq s}\left(q_j^{(l)}(k) - q_i^{(l)}(k)\right)x_{ij}^{(l)}(k) \\
&+ 2\sum_{i=1}^{N}\sum_{l=1}^{L-1}\left(q_i^{(l+1)}(k) - q_i^{(l)}(k)\right)y_i^{(l)}(k) \\
&+ 2\sum_{i=1}^{N} q_i^{(1)}(k)y_i^{(0)}(k),
\end{aligned} \tag{3–40}$$

where $B \triangleq LN^3M^2$. Equation (3–40) can be obtained by regrouping the terms based on variables x and y. Note that the forth term in the last equality excludes links to the sink.

Adding $2q_s(k)(\sum_l \sum_{(j,s)\in\mathcal{A}^l} x_{js}^{(l)}(k)) = 0$ to (3–40), we have

$$\begin{aligned}\Delta(k) \leq & B + 2\sum_{i=1}^{N} q_i^{(1)}(k) D d_i \\ & + 2\sum_{l} \sum_{(i,j)\in\mathcal{A}^l} \left(q_j^{(l)}(k) - q_i^{(l)}(k)\right) x_{ij}^{(l)}(k) \\ & + 2\sum_{i=1}^{N}\sum_{l=1}^{L-1} \left(q_i^{(l+1)}(k) - q_i^{(l)}(k)\right) y_i^{(l)}(k).\end{aligned} \tag{3–41}$$

Note that the third term now includes the links to the sink. We also used the fact $y_i^{(0)}(k) = D d_i$. Adding $(2/\delta)z(k)$ to both sides of inequality (3–41), we get

$$\begin{aligned}\Delta(k) + \frac{2}{\delta} z(k) \leq & B + 2\sum_{i=1}^{N} q_i^{(1)}(k) D d_i \\ & + 2\left\{ \frac{z(k)}{\delta} + \sum_{l} \sum_{(i,j)\in\mathcal{A}^l} (q_j^{(l)}(k) - q_i^{(l)}(k)) x_{ij}^{(l)}(k) \right\} \\ & + 2\sum_{i=1}^{N}\sum_{l=1}^{L-1} \left(q_i^{(l+1)}(k) - q_i^{(l)}(k)\right) y_i^{(l)}(k) \\ & \leq B + 2\sum_{i=1}^{N} q_i^{(1)}(k) D d_i \\ & + 2\left\{ \frac{\hat{z}(\epsilon)}{\delta} + \sum_{l} \sum_{(i,j)\in\mathcal{A}^l} (q_j^{(l)}(k) - q_i^{(l)}(k)) \hat{x}_{ij}^{(l)}(\epsilon) \right\} \\ & + 2\sum_{i=1}^{N}\sum_{l=1}^{L-1} \left(q_i^{(l+1)}(k) - q_i^{(l)}(k)\right) \hat{y}_i^{(l)}(\epsilon),\end{aligned} \tag{3–42}$$

where $(\hat{z}(\epsilon), \hat{x}(\epsilon), \hat{y}(\epsilon))$ is an optimal solution of ϵ-perturbed problem defined as (3–33) - (3–36). Note that the last inequality holds because $(z(k), x(k), y(k))$ is a minimizer of the objective function in subproblems S_1 and S_2. After regrouping inequality (3–42) according to the dual multipliers, we have

$$\begin{aligned}\Delta(k) + \frac{2}{\delta} z(k) \leq & B + \frac{2}{\delta}\hat{z}(\epsilon) + \\ & 2\sum_{l=1}^{L}\sum_{i=1}^{N} q_i^{(l)}(k) \Big(\sum_{j:i\in N_l(j)} \hat{x}_{ji}^{(l)}(\epsilon) - \sum_{j\in N_l(i)} \hat{x}_{ij}^{(l)}(\epsilon) - \hat{y}_i^{(l)}(\epsilon) + \hat{y}_i^{(l-1)}(\epsilon) \Big)\end{aligned}$$

$$= B + \frac{2}{\delta}\hat{z}(\epsilon) - 2\epsilon \sum_{l=1}^{L}\sum_{i=1}^{N} q_i^{(l)}(k) \tag{3–43}$$

In the first inequality, $2\sum_{i=1}^{N} q_i^{(1)}(k)Dd_i = 2\sum_{i=1}^{N} q_i^{(1)}(k)\hat{y}_i^{(0)}(\epsilon)$, which is the second term in (3–42) and is now combined with the third term. Also, recall that $(\hat{z}(\epsilon), \hat{x}(\epsilon), \hat{y}(\epsilon))$ satisfies the flow conservation constraint (3–35). Therefore, $\sum_j \hat{x}_{ji}^{(l)}(\epsilon) - \sum_j \hat{x}_{ij}^{(l)}(\epsilon) - \hat{y}_i^{(l)}(\epsilon) + \hat{y}_i^{(l-1)}(\epsilon) = -\epsilon$ and the first equality is satisfied. This concludes the proof. □

We show in the theorem 3.2 that the time averaged solution of $(z(k), x(k), y(k))$ converges to the optimal solution of the primal problem. Define $Q(k) = \sum_{l=1}^{L}\sum_{i=1}^{N} q_i^{(l)}(k)$. Note that $Q(k)$ can be interpreted as the sum of all virtual queue sizes in the system at the time slot k.

Theorem 3.2. *The following inequalities hold:*

$$\limsup_{T\to\infty,\epsilon\to 0} \bar{z}(T) \le \frac{\delta B}{2} + z^*, \tag{3–44}$$

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{k=0}^{T-1} Q(k) \le \frac{B}{2\epsilon} + \frac{1}{\delta\epsilon}\hat{z}(\epsilon) \tag{3–45}$$

Proof. Aggregating inequalities (3–39) over $k = 0, 1, \dots, T-1$, we have

$$\begin{aligned} V(q(T)) - V(q(0)) + \frac{2}{\delta}\sum_{k=0}^{T-1} z(k) \le BT + \frac{2}{\delta}T\hat{z}(\epsilon) \\ -2\epsilon\sum_{k=0}^{T-1} Q(k) \end{aligned} \tag{3–46}$$

Multiplying $\delta/(2T)$ and arranging appropriately, we get

$$\begin{aligned} \bar{z}(T) \triangleq \frac{1}{T}\sum_{k=0}^{T} z(k) &\le \frac{\delta B}{2} + \hat{z}(\epsilon) \\ &- \frac{\delta\epsilon}{T}\sum_{k=0}^{T-1} Q(k) - \frac{\delta V(q(T))}{2T} + \frac{\delta V(q(0))}{2T} \\ &\le \frac{\delta B}{2} + \hat{z}(\epsilon) + \frac{\delta V(q(0))}{2T}. \end{aligned} \tag{3–47}$$

Hence

$$\limsup_{T\to\infty,\epsilon\to 0} \bar{z}(T) \leq \frac{\delta B}{2} + z^*, \tag{3–48}$$

where z^* is the optimal objective function value of (3–10)–(3–16). Note that $\hat{z}(\epsilon) \to z^*$ as $\epsilon \to 0$ by the Theorem 3.1. From (3–46), we have

$$\begin{aligned} & 2\epsilon \sum_{k=0}^{T-1} Q(k) \\ & \leq BT + \frac{2T}{\delta}\hat{z}(\epsilon) + V(q(0)) - V(q(T)) - \frac{2}{\delta}\sum_{k=0}^{T-1} z(k) \\ & \leq BT + \frac{2T}{\delta}\hat{z}(\epsilon) + V(q(0)) \end{aligned} \tag{3–49}$$

Multiplying the both side of the above inequality by $1/(2T\epsilon)$, we get

$$\frac{1}{T}\sum_{k=0}^{T-1} Q(k) \leq \frac{B}{2\epsilon} + \frac{1}{\delta\epsilon}\hat{z}(\epsilon) + \frac{V(q(0))}{2T\epsilon}. \tag{3–50}$$

Hence,

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{k=0}^{T-1} Q(k) \leq \frac{B}{2\epsilon} + \frac{1}{\delta\epsilon}\hat{z}(\epsilon), \tag{3–51}$$

which concludes the proof. □

3.5 Computational Experiments

In this section, we present two numerical experiments to prove the validity of our algorithm. More specifically, we show that our algorithm converges to the optimal objective function value of problem (3–10)–(3–16). We also show how the Lyapunov drift and the queue sizes evolve. For these experiments, we randomly place 50 sensors in a circular region with a radius 25m and select 6 sink locations in the same region for the sink to visit. The transmission cost between two nodes depends on the distance between them [30]. Data generation rate of node i is selected from $[0, 500]$ bps and each node has 500 J of initial energy. In the subgradient projection method, we use a constant step size of $\delta = 10^{-8}$. In all experiments, the perturbation parameter is $\epsilon = 10^{-8}$.

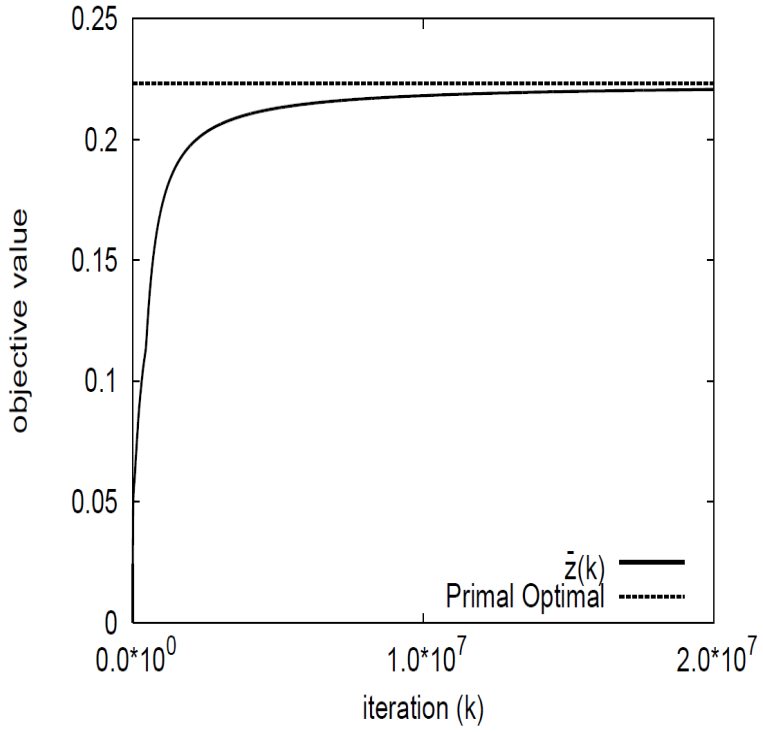


Figure 3-2. Convergence to the optimal value

Figure 3-2 shows the convergence of our algorithm to the optimal objective function value. As a reference, the optimal solution of the primal problem (3–10)–(3–16) is obtained using CPLEX. The curve labeled as $\bar{z}(k)$ is the average value of $z(k)$ at iteration k. Figure 3-2 verifies the validity of the first inequality in Theorem 3.2. Finally, Figure 3-3 shows the long-run average value of the total queue size, $\sum_l \sum_i q_i^{(l)}$. By the second part of Theorem 3.2, this value is bounded from above, which is verified here.

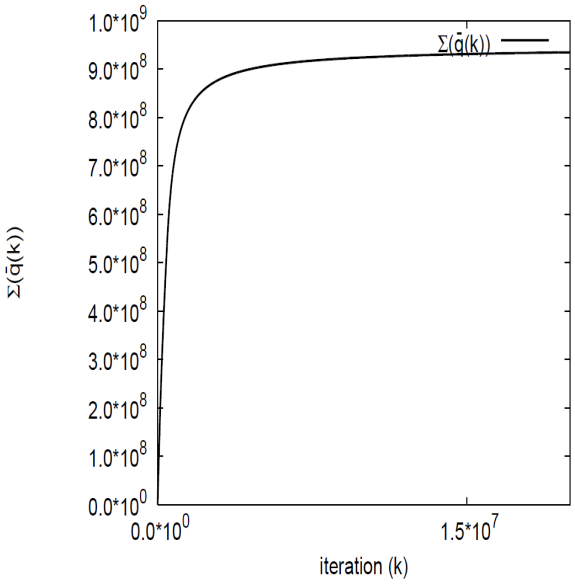


Figure 3-3. Time average of total virtual queue size over time

CHAPTER 4
AN INTEGER-PROGRAMMING-BASED APPROACH TO THE CLOSE-ENOUGH TRAVELING SALESMAN PROBLEM

4.1 Problem Definition

Given a collection of nodes and the set of distances between each pair of nodes, the traveling salesman problem (TSP) seeks to find a shortest tour that visits each node exactly once. The broad applicability of the TSP, or slight variations thereof, have resulted in an impressive slate of research on TSP-related problems. Several TSP extensions can be found in the literature, such as the covering salesman problem [39], prize collecting TSP [40, 41], and the covering tour problem (CTP) [42]. For a comprehensive survey on the history, algorithms, and applications of the TSP, the reader is referred to [14].

The close-enough traveling salesman problem (CETSP) is a generalization of the TSP in which the salesman does not need to visit the exact location of each customer. Instead, a compact region of the plane containing each node is specified as its *neighborhood set*, and the goal is to find a shortest tour that starts from a specified depot location and intersects all of these neighborhood sets. Intuitively speaking, in the CETSP, each of the salesman's clients is willing to travel to any point inside its particular neighborhood to meet with the salesman. A typical application of the CETSP arises when an airborne vehicle is trying to find a stationary target on a two-dimensional field by scanning a collection of candidate locations. Suppose that the detection occurs when the aircraft is no more than r units away from the target. Then the neighborhood set of a target is a disc of radius r at the target's location. The problem of finding the shortest trajectory for the aircraft is equivalent to finding a tour of minimum length that intersects each neighborhood set.

Another important application for the CETSP arises in wireless sensor network operations. As we discussed in Chapters 2 and 3, in these networks sensor nodes periodically relay data that they have accumulated to a mobile data-collecting device

called a *sink*. Energy minimization is an important component of wireless sensor networks. Ciullo et al. [11] show that by letting the sink come closer to sensors with higher data generation rates, one can significantly reduce the consumed energy for the transmission of data. Therefore, it is reasonable to assign a sensing radius r_i to each sensor node i based on its data generation rate, and require that the sink must visit a point within the r_i-neighborhood of node i to collect its data. Again, the problem of finding a shortest trajectory of the sink to retrieve data from all sensors is an instance of the CETSP (see also [43]).

Gulczynski et al. [44] propose several heuristics for a common special case of the CETSP where the neighborhood sets of all nodes are discs of the same radius. The problem arises in situations where radio frequency identification (RFID) tags are used that can remotely provide a mobile data collector with the required data. As an example, many utility companies are now using RFID-based automated meter readers that can read the usage of each customer remotely [12]. Mennell et al. [13] propose a heuristic algorithm based on *Steiner zones*, which are nonempty intersections of the neighborhood sets. Their approach consists of three phases: (1) identifying a collection of Steiner zones that cover every neighborhood set; (2) representing each Steiner zone with one of its points; and (3) finding a TSP tour over these representative points.

Arkin and Hassin [45] propose polynomial-time approximation algorithms for several special cases of the problem. In particular, they provide algorithms that yield error bounds for the CETSP in which the neighborhood sets either take the form of parallel unit segments, translates of polygonal regions, or discs. Other approximation algorithms include the work of Mata and Mitchell in [46] and Dumitrescu and Mitchell [47].

The heuristic algorithm of Mennell [13] and the polynomial-time approximation algorithms in [45–47] are able to efficiently find a good feasible CETSP tour. One difficulty in evaluating the quality of such feasible solutions is the lack of exact algorithms for the CETSP in the literature. Hence, developing tight lower bounds for the CETSP is

of crucial importance. Such lower bounds would enable one to (conservatively) evaluate the quality of a feasible tour obtained by a non-exact algorithm. While there exist several methods of efficiently obtaining such lower bounds for the TSP, developing lower bounds for the CETSP is considered to be a non-trivial task. Our contribution is targeted toward finding arbitrarily tight lower and upper bounds on the optimal CETSP tour length via mixed-integer programming models.

Some of the most successful exact algorithms for solving the symmetric TSP combine the use of cutting planes and efficient heuristics in a branch-and-cut scheme. Examples of such branch-and-cut algorithms are the work of Padberg and Rinaldi [48] and the well known Concorde TSP solver [49]. The approach that we propose in this chapter requires solving integer programming problems to obtain a series of upper and lower bounds that converge to an optimal CETSP tour. The discrete nature of this algorithm provides a framework for using cutting planes that have been proved to be most successful in solving large symmetric TSP instances.

The rest of this chapter is organized as follows. Section 4.2 establishes foundational concepts related to optimal CETSP tours and the discretization scheme used in this chapter. Section 4.3 provides a mathematical programming formulation that yields a lower bound on the optimal CETSP tour length. Based on the anticipated difficulties of solving this model, we propose an alternative model and associated two-stage optimization algorithm in Section 4.4. We present computational results that demonstrate the efficiency of our approach in Section 4.6.

4.2 Preliminaries

We begin by stating the formal definition of the CETSP and describing the notation used in this chapter in Section 4.2.1, and then present our core discretization scheme in Section 4.2.2.

4.2.1 Definitions and Notation

Let M be a set of points in a two-dimensional plane along with a depot point p_0. For each point $m \in M$, let S_m be a compact set that contains m. The CETSP seeks to find a shortest tour (with respect to Euclidean distance) that starts from p_0, intersects every set $S_1, \ldots, S_{|M|}$ (in any order), and terminates at p_0.

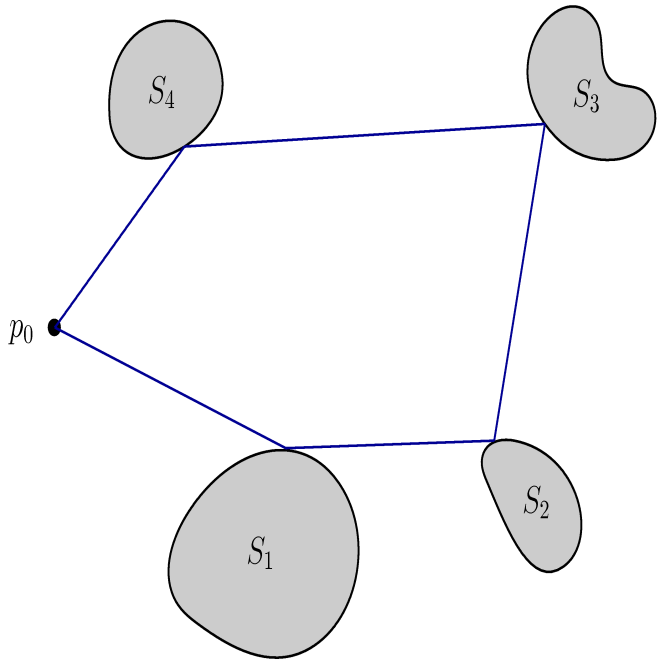


Figure 4-1. A feasible tour with $|M| = 4$

Figure 5-1A illustrates a CETSP tour that intersects four compact sets. Note that the CETSP is clearly NP-hard as it reduces to the TSP when each set S_m consists of a single point. Without loss of generality, we assume that $p_0 \notin S_m$, $\forall m \in M$, or else we can simply ignore all $m \in M$ for which $p_0 \in S_m$.

We will refer to the elements of M as the *target locations* and their associated compact sets as the *neighborhood sets*. These neighborhood sets are often discs, e.g., for the air monitoring and meter reading applications mentioned previously. However,

the approach that we present in this chapter for solving the CETSP in general does not depend on the shape of the neighborhood sets.

The following proposition is useful in the development of our algorithm.

Proposition 4.1. *All optimal solutions to the CETSP consist of a finite set of connected line segments* (p_0p_1), (p_1p_2), ..., $(p_{k-1}p_k)$, (p_kp_0), *where* $k \leq |M|$. *Moreover, for each point* p_i, $i = 1, \dots, k$, *there exists at least one* $m \in M$ *such that* p_i *is on the boundary of* S_m.

Proof. Let T be any feasible CETSP tour. We can select a finite number of points $p_1, \dots, p_k$ on T such that $k \leq |M|$ and $\left(\cup_{j=1}^{k} p_j\right) \cap S_m \neq \emptyset$, $\forall m \in M$. Note that the line segment between any pair of these points is the unique minimizer of distance between them. Therefore, the class of solutions composed of line segments connecting these points dominates all other classes of solutions, and so an optimal CETSP tour must consist of some $k \leq |M|$ line segments.

Hence, we now consider an optimal tour consisting of line segments (p_0p_1), (p_1p_2), ..., $(p_{k-1}p_k)$, (p_kp_0), where $1 \leq k \leq |M|$. For each $i = 1, \dots, k$, define V_i as the set of target neighborhoods visited by p_i, but not by $p_1, \dots, p_{i-1}$, i.e.,

$$V_i = \{m \in M : p_i \in S_m \text{ and } m \notin V_j \text{ for } j < i\}. \tag{4–1}$$

We assume that each V_i is nonempty; otherwise, if $V_i = \emptyset$ then we can skip the corresponding point p_i in the tour (moving from p_{i-1} to p_{i+1}, where $p_{k+1} = p_0$), which results in a tour whose length is no more than the original $(k+1)$-link tour. Let

$$R_i = \cap_{m \in V_i} S_m, \qquad \forall i = 1, \dots, k. \tag{4–2}$$

Each R_i is a closed set since all the sets S_m are so. We claim that there exists an optimal tour in which p_i is on the boundary of R_i. By contradiction, suppose that p_i belongs to the interior of R_i and the two line segments $(p_{i-1}p_i)$ and (p_ip_{i+1}) intersect the boundary of R_i at points p_i' and p_i'', respectively (Figure 4-2). An alternative tour that

traverses from p_i' directly to p_i'' and bypasses p_i is no longer than T, but still intersects R_i. Therefore, there must exist an optimal solution in which each point p_i, $i = 1, \ldots, k$, lies on the boundary of at least one set S_m for some $m \in M$. This completes the proof.

□

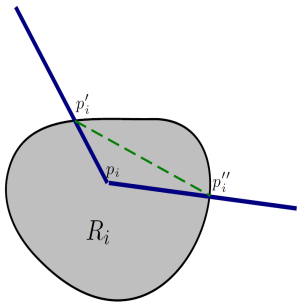


Figure 4-2. Illustration of boundary point optimality

Proposition 5.4 implies that any optimal solution to the CETSP can be characterized by a discrete set of points in the plane, where each point belongs to the boundary of at least one neighborhood set. In the rest of the chapter, we will call these the *turn points* of the corresponding tour. Proposition 4.2 further characterizes those sections of the boundary sets that may contain a turn point of an optimal tour.

Proposition 4.2. *Suppose that every neighborhood set S_m is a disc centered at $m \in M$. Let T be an optimal CETSP tour that is characterized by a set of turn points $\{p_0, \ldots, p_k\}$. Then, $p_i \in conv(M \cup \{p_0\})$, for $i = 1, \ldots, k$, where $conv(M \cup \{p_0\})$ denotes the convex hull of the corresponding points.*

Proof. First, define $D = conv(M \cup \{p_0\})$. The dimension of D must be at least one (or else $D = \{p_0\}$ and thus $|M| = 0$). If the dimension of D is one, the proposition follows by noting that an optimal tour traverses the length of D from p_0 toward each of its two endpoints, stopping once all neighborhood sets are visited near each end of D and then back to p_0. Now consider the case in which D is two-dimensional, and by contradiction, suppose that T is an optimal tour that contains points outside D. Let q' be a point on

T that intersects the boundary of D, such that the trajectory of T immediately leaves D (i.e., $\forall\varepsilon$ such that $0 < \varepsilon \leq \bar{\varepsilon}$, point $q' + \varepsilon d$ is on T but is not in D, for some trajectory vector d and a positive $\bar{\varepsilon}$). Because q' is on the boundary of D, it must be a convex combination of two points c_1 and c_2 in $M \cup \{p_0\}$. Consider the (infinite) line intersecting c_1 and c_2 and define H as the halfspace with respect to this line that contains D; also, define $\overline{H}$ as the halfspace consisting of this same line and points on the opposite side of D (so that $H \cap \overline{H}$ is just the line intersecting c_1 and c_2). Following the trajectory of T after it leaves D at q', suppose that T re-enters H at q''. This point must exist because the tour must return to $p_0 \in H$.

Now consider an alternative tour $\hat{T}$, which is the same as T except that the segment of T between q' and q'' (all of which lies outside of H) is replaced with a segment that lies completely on the line spanning c_1 and c_2. Let $\overline{M} = \{m \in M : T$ visits the neighborhood set of m on the segment between q' and q'' that lies in $\overline{H}\}$. If $\overline{M} = \emptyset$, then replacing the segment of T with the straight segment between q' and q'' retains the feasibility of the tour while reducing its length; hence, T could not be optimal (Figure 4-3). Otherwise, suppose that $\overline{M} \neq \emptyset$. Because the line segment intersecting c_1 and c_2 induces a facet of D, all targets in $\overline{M}$ can be visited by traversing this segment (or else, some $m \in \overline{M}$ would not belong to D). Suppose that $[\lambda_{\min}, \lambda_{\max}]$ is the smallest interval such that the set of all points $q' + \lambda(q'' - q')$ for $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ intersects neighborhood set for each $m \in \overline{M}$. Define $\overline{\lambda}_{\min} = \min\{0, \lambda_{\min}\}$ and $\overline{\lambda}_{\max} = \max\{1, \lambda_{\max}\}$. The unique shortest tour segment τ that is completely contained in $\overline{H}$, starts at q', intersects the neighborhood set of each $m \in \overline{M}$, and ends at q'' is described as follows. Segment τ starts at the point on the line segment where $\lambda = 0$, moves to the point where $\lambda = \overline{\lambda}_{\min}$, then to the point where $\lambda = \overline{\lambda}_{\max}$, and returns to the point where $\lambda = 1$ (Figure 4-4). Now, suppose that we create a new tour $\hat{T}$, which is the same as T except where τ replaces the segment from q' to q'' in T. Because the segment in T joining q' and q'' is contained in $\overline{H}$ and is not identical to τ, its distance is longer than that of τ. Thus, tour $\hat{T}$ is still

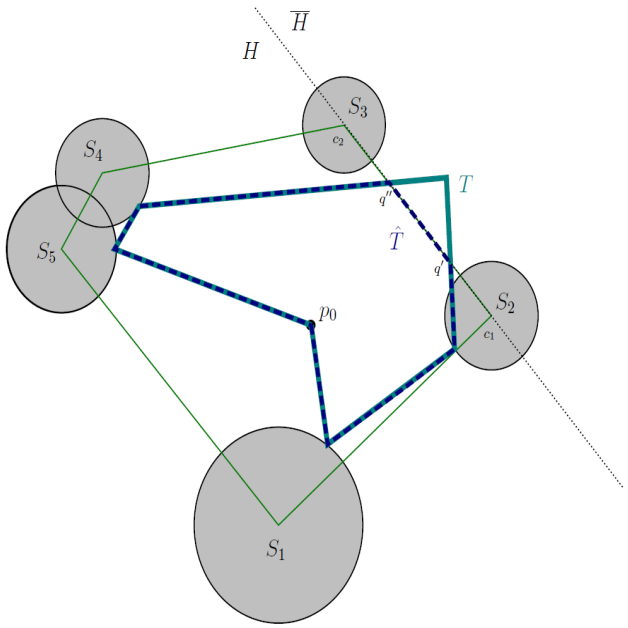


Figure 4-3. Illustration of the convex hull of target points

feasible but is shorter than T, which contradicts the optimality of T. This completes the proof. □

4.2.2 Partitioning Schemes

Since an optimal CETSP tour can be represented by a finite number of turn points, a natural way of obtaining a feasible tour is to approximate the solution space by a discrete set of points. However, such an approach results only in an upper bound on the optimal CETSP tour length. Our approach for obtaining lower bounds on the optimal CETSP tour length in this chapter is based on partitioning the continuous solution space into smaller sets and identifying those partitions that possibly contain a turn point of an optimal CETSP tour.

Definition 4.1. A set $C = \{C_0, \dots, C_n\}$ is called a CETSP-partitioning of the two dimensional plane if:

1. $C_0 = \{p_0\}$.
2. C_i is a nonempty compact set, for all $1 \leq i \leq n$.

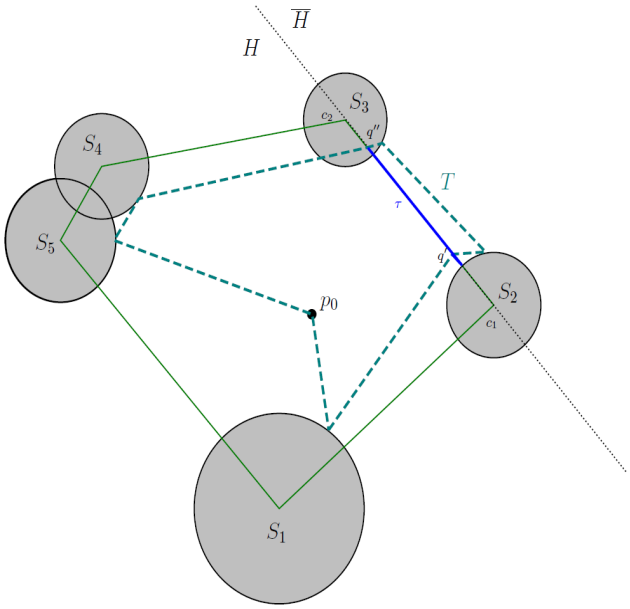


Figure 4-4. Illustration of the convex hull argument in proof of Proposition 4.2

3. $conv(C_i \cap C_j)$ has an empty interior, for all $1 \leq i, j \leq n$.

4. $(\cup_{i=1}^{n} C_i) \supseteq (\cup_{i \in M} B_m)$, where B_m is the boundary of S_m.

When every neighborhood set S_m is a disc that is centered at its corresponding target location, B_m in the above definition can be restricted to include those points on the boundary of S_m that lie inside $conv(M \cup \{p_0\})$. □

We refer to the elements of a CETSP-partitioning C as *cells*. For any two cells C_i and C_j, l_{ij} is defined as the shortest line segment length that connects the boundaries of C_i and C_j. See Section 4.5 for details on computing l_{ij}-values.

In this chapter, we consider two ways of partitioning the plane: grid-based and arc-based. In a grid-based partitioning, each grid cell C_i is a rectangle that intersects at least one neighborhood set (see Figure 4-5). The grid cells are chosen so that they collectively contain all the neighborhood sets and the depot. Moreover, we will restrict ourselves to grid-based partitionings in which there exists no neighborhood set that is entirely contained in one of the rectangles. We define a set P of *grid points* as the collection of p_0 and any other point that is a vertex of at least one grid cell. Throughout

the chapter, we will represent grid cell i by a tuple $(a_i, b_i, w_{1i}, w_{2i})$, where (a_i, b_i) denotes the lower-left vertex, and w_{1i} and w_{2i} are the two side lengths of the rectangle, so that $(a_i + w_{1i}, b_i + w_{2i})$ denotes the upper-right vertex.

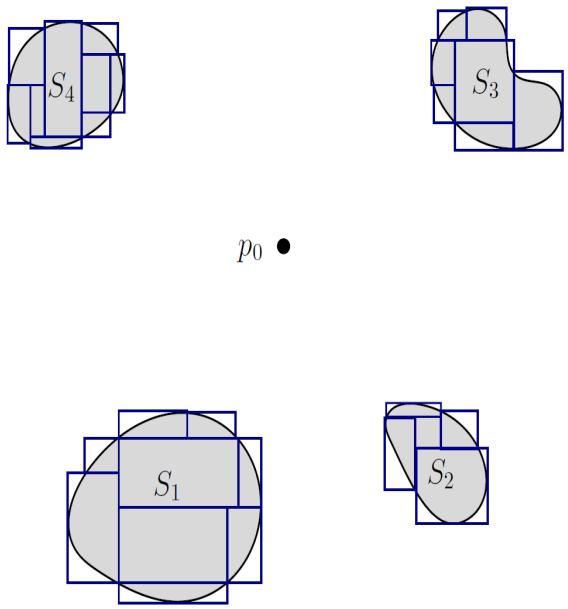


Figure 4-5. Illustration of grid-based partitioning

As opposed to the grid-based scheme, arc-based partitioning is primarily intended for circular neighborhood sets. Figure 4-6 illustrates an example of an arc-based partitioning for a CETSP instance. Here, a cell is defined as an arc (specified by its two endpoints) on the boundary of a set S_m that lies on or inside the convex hull of the target locations and the depot. Note that by Proposition 4.2, every turn point of an optimal CETSP tour belongs to at least one arc in this partitioning. We will specify an arc with $(a_i, b_i, r_i, \alpha_{1i}, \alpha_{2i})$ where (a_i, b_i) and r_i are respectively the center and radius of the corresponding disc, and α_{1i} and α_{2i} respectively denote the start and finish angles of the corresponding arc. The *central angle* of an arc is defined as $\alpha_i = \alpha_{2i} - \alpha_{1i}$. Throughout the chapter, we will assume that $\alpha_i < \pi$ for all CETSP-partitioning arcs. Similar to the grid-based case, set P is defined as the set of endpoints of all arcs unioned with p_0.

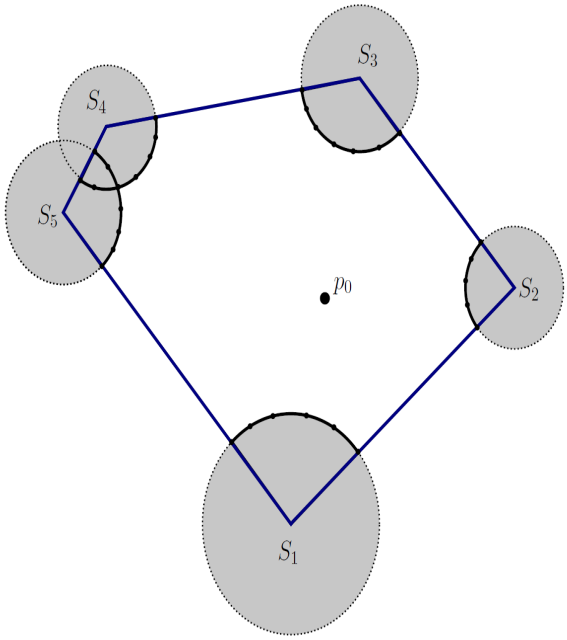


Figure 4-6. Illustration of arc-based partitioning

4.3 Lower Bounding Model

We first formulate a mixed-integer program (MIP) for obtaining lower bounds to the optimal CETSP tour length in Section 4.3.1, along with a closed-form expression for an accompanying upper bound. We then describe two cutting-plane strategies to aid in the solution of this model in Section 4.3.2, and comment on the complexity of their separation routines.

4.3.1 Formulation and Bounds

Let $C = \{C_0, \dots, C_n\}$ be a CETSP-partitioning of the plane with a pairwise distance matrix $L = \{l_{ij}\}$. For each $m \in M$ define the set of cells intersecting S_m as $N(m) = \{1 \leq i \leq n : C_i \cap S_m \neq \emptyset\}$. Note by Definition 4.1 we have $N(m) \neq \emptyset$ for all $m \in M$. Consider the following MIP.

$$\text{Min} \quad \sum_{i=0}^{n} \sum_{j=0}^{n} l_{ij} x_{ij} \tag{4–3a}$$

$$\text{s.t.} \quad \sum_{j=0}^{n} x_{ij} = \sum_{j=0}^{n} x_{ji}, \quad \forall i = 0, \dots, n \tag{4–3b}$$

$$y_i = \sum_{j=0}^{n} x_{ij}, \quad \forall i = 0, \dots, n \tag{4–3c}$$

$$\sum_{i \in N(m)} y_i \geq 1, \quad \forall m \in M \tag{4–3d}$$

$$\sum_{i \in \mathcal{S}} \sum_{j \notin \mathcal{S}} x_{ij} \geq y_v, \quad \forall \mathcal{S} \subset \{1, \dots, n\} : 2 \leq |\mathcal{S}| \leq |C| - 2 \text{ and } v \in \mathcal{S} \tag{4–3e}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i = 0, \dots, n, \ j = 0, \dots, n \tag{4–3f}$$

$$0 \leq y_i \leq 1, \ \forall i = 1, \dots, n; \quad y_0 = 1. \tag{4–3g}$$

An optimal solution to (5–21) is in fact a shortest TSP tour with respect to distance matrix L, over a subset of C that visits at least one element in $N(m)$ for each $m \in M$. Decision variable x_{ij} indicates whether the corresponding tour moves from C_i to C_j, while auxiliary binary variable y_i indicates whether C_i is visited on the tour. The objective function (4–3a) minimizes the total distance traveled in the tour. Constraints (4–3b) ensure that for each node i, the number of incoming tour arcs equals the number of outgoing tour arcs. Constraints (4–3c) define y-variables in terms of x-variables. (In fact, the formulation can be given without the y-variables; they are included only for convenience in presentation.) Constraints (4–3d) ensure that for each $m \in M$, at least one element of C is visited that covers m. Constraints (4–3e) are subtour elimination constraints (see[50]). Finally, (4–3f) and (4–3g) state logical restrictions and bounds on the variables.

Problem (5–21) is a special case of the CTP [42]. The CTP is defined on a graph $G = (V \cup W, E)$ where V is the set of vertices that can be visited and W is the set of targets that must be covered by vertices in V. For every $w \in W$, there exists a nonempty subset $V^w \subseteq V$ of vertices that cover w. There is also a subset $\overline{V} \subseteq V$ containing the vertices that must be visited by any feasible tour. The goal in the CTP is to find a shortest tour on a subset of V that visits all vertices of $\overline{V}$ as well as at least one

vertex of V^w, for every $w \in W$. Therefore, problem (5–21) is a special case of the CTP on a complete directed graph with $V = C \cup \{p_0\}$, $W = M$, $\overline{V} = \{p_0\}$, and $V^m = N(m)$.

The following proposition establishes a relationship between the optimal objective function value of problem (5–21) and the optimal CETSP tour length.

Proposition 4.3. *Suppose that the optimal CETSP tour length is l^*, and consider an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (5–21) with objective function value z_{LB1}. Define $I = \{i : y_i^* = 1\}$, which indicates the set of turn cells associated with this optimal solution, and let $N_i = |\{m \in M : i \in N(m)\}|$ be the number of neighborhood sets intersected by C_i. Then*

$$z_{LB1} \leq l^* \leq z_{LB1} + \sum_{i \in I} h_i, \tag{4–4}$$

where

$$h_i = \begin{cases} 2N_i r_i \sin\left(\frac{\alpha_i}{2}\right) & \text{if } N_i \leq 2, \\ 2r_i \left(2\sin\left(\frac{\alpha_i}{4}\right) + (N_i - 1)\sin\left(\frac{\alpha_i}{2(N_i-1)}\right)\right) & \text{if } N_i \geq 3, \end{cases} \tag{4–5}$$

for an arc-based CETSP-partitioning, and

$$h_i = \begin{cases} 2\sqrt{{w_{1i}}^2 + {w_{2i}}^2} & \text{if } N_i = 1, \\ w_{1i} + w_{2i} + \sqrt{{w_{1i}}^2 + {w_{2i}}^2} & \text{if } N_i = 2, \\ 2(w_{1i} + w_{2i}) & \text{if } N_i = 3, \\ 2(w_{1i} + w_{2i}) + \sqrt{{w_{1i}}^2 + {w_{2i}}^2} & \text{if } N_i \geq 4, \end{cases} \tag{4–6}$$

for a grid-based CETSP-partitioning.

Proof. Suppose that an optimal CETSP tour T is characterized by an ordered set of turn points $(p_0, \ldots, p_k)$. Let $C'_t = C_{i_t} \in C$ be an element of the CETSP-partitioning that contains p_t, for $t = 1, \ldots, k$ and define $C'_{k+1} = C'_0 = C_0$ and $i_{k+1} = i_0 = 0$. Because cells might contain multiple turn points, some elements of $C' = \{C'_1, \ldots, C'_k\}$ may be identical. Suppose that

- $y^R_{i_t} = 1$, for $t = 0, \ldots, k$,

- $x^R_{0,i_1} = 1$, and
- $x^R_{i_t,i_s} = 1$, for $t = 1, \dots, k$ and $s > t$, if and only if $x^R_{i_u,i_t} = 1$ for some $u < t$ and s is the first index strictly greater than t such that $C'_s \notin \{C'_1, \dots, C'_t\}$.

By construction, $(\mathbf{x}^R, \mathbf{y}^R)$ defines a tour over a subset $\{C_{i_0}, \dots, C_{i_k}\} \subseteq C$ that covers all the targets in M. Hence, $(\mathbf{x}^R, \mathbf{y}^R)$ is feasible to (5–21). Let $z^R = \sum_{(i,j)\in A^R} l_{i,j}$ denote the objective function value of $(\mathbf{x}^R, \mathbf{y}^R)$, where $A^R = \{(i,j) : 0 \le i,j \le n, x^R_{i,j} = 1\}$. Now suppose $x^R_{i_t,i_s} = 1$. Because l_{i_t,i_s} is defined as the minimum distance between C_{i_t} and C_{i_s} and $p_t \in C_{i_t}$ and $p_s \in C_{i_s}$, we have

$$l_{i_t,i_s} \le \sum_{j=t}^{s-1} |p_j p_{j+1}|, \tag{4–7}$$

where $|p_i p_{i+1}|$ is the length of line segment between p_i and p_{i+1}, with $p_{k+1} = p_0$. Aggregating (4–7) over all elements of A^R, we obtain $z^R \le l^*$. Therefore, $z_{LB1} \le z^R \le l^*$.

To prove the validity of the upper bound in (4–4), we form a feasible CETSP tour $\hat{T}$ whose length is bounded by $z_{LB1} + \sum_{i\in I} h_i$. Recall that an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (5–21) consists of a collection of (possibly disconnected) links between turn cells in I (see Figure 4-7 for example). Suppose $x^*_{ji} = x^*_{ik} = 1$. That is, the corresponding optimal solution to (5–21) contains an incoming link (with length l_{ji}) from C_j to a point p^i_1 in C_i. Similarly, there is a point p^i_2 in C_i from which there exists a link to C_k. (Note that p^i_2 may not be the same point as p^i_1.) One can construct $\hat{T}$ from this optimal solution to (5–21) by connecting p^i_1 to p^i_2 for each $i \in I$ via a shortest trajectory that visits all N_i targets that are covered by C_i.

We first consider an arc-based CETSP-partitioning where each C_i is an arc of radius r_i and central angle α_i. The connecting trajectory in this case starts from p^i_1, visits (at most) $N_i - 1$ *middle points* on the arc to cover all the other targets that are covered by C_i, and then moves to p^i_2. Because this trajectory interconnects a collection of points on the underlying arc, it consists of several chords. The chord length of an arc with radius r and central angle α equals $2r\sin(\alpha/2)$. When $N_i \le 2$, a shortest connecting trajectory

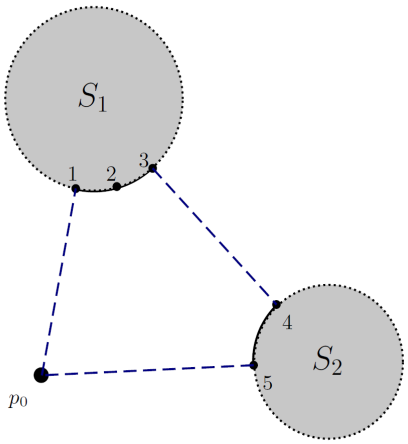


Figure 4-7. An instance of the lower bound problem (5–21) and its optimal solution

consists of at most N_i chords and therefore (4–5) holds. For $N_i \geq 3$, we first compute an upper bound on the shortest total length of $N_i - 1$ chords that interconnect p_1^i and the remaining middle points. This bound is given by the optimal objective function value of the following problem.

$$\text{Max} \quad \sum_{k=1}^{N_i-1} 2r_i \sin\left(\frac{\beta_k}{2}\right) \tag{4–8}$$

$$\text{s.t.} \quad \sum_{k=1}^{N_i-1} \beta_k \leq \alpha_i \tag{4–9}$$

$$\beta_k \geq 0, \quad \forall k = 1, \ldots, N_i - 1. \tag{4–10}$$

Since $\alpha_i < \pi$, the objective function is concave and so the unique optimal solution to the above problem is $\beta_1 = \cdots = \beta_{N_i-1} = \alpha_i/(N_i - 1)$, and the maximum length of these $N_i - 1$ chords is given by $2r_i(N_i - 1)\sin(\alpha_i/2(N_i - 1))$. However, we possibly need additional chords to complete this connecting trajectory. When p_1^i is an endpoint of the underlying arc, the connecting section starts from p_1^i, visits all the middle points, and then traverses to p_2^i. An upper bound on the length of the trajectory in this case is given by $2r_i \sin(\alpha_i/2) + 2r_i(N_i - 1)\sin(\alpha_i/2(N_i - 1))$. Now suppose p_1^i is not an endpoint of the underlying arc, and let P_1 be the endpoint that is closer to p_1^i (compared to p_2^i)

and P_2 be the other endpoint. One possible way to connect p_1^i and p_2^i is to start from p_1^i, visit all the middle points on the way to P_1, come back to p_1^i, visit the remaining middle points, and finally move to p_2^i. In this case, two chords ($P_1p_1^i$, and $P_2p_2^i$) are added to the previous $N_i - 1$ chords. Let γ_1 and γ_2 be the central angles corresponding to the two new chords. Because $\gamma_1 + \gamma_2 \leq \alpha_i$, the total length of these two new chords cannot exceed $4r_i \sin(\alpha_i/4)$ (which is greater than $2r_i \sin(\alpha_i/2)$ for all values of α_i). Hence, an upper bound on the length of the shortest connecting trajectory is given by $4r_i \sin(\alpha_i/4) + 2r_i(N_i - 1)\sin(\alpha_i/2(N_i - 1))$.

Now consider a grid-based CETSP-partitioning. The worst-case scenario for each value of N_i occurs when p_1^i exists at a vertex of the grid cell. When $N_i = 1$, the length of a minimal connecting segment is clearly no more than $2\sqrt{{w_{1i}}^2 + {w_{2i}}^2}$, which happens when $p_1^i = p_2^i$ but a diagonal of the grid cell needs to be traversed once to cover a target, and once to return to p_2^i. When $N_i = 2$, the connecting trajectory consists of three connected line segments. It can be verified that the maximum length of the minimal trajectory in the underlying rectangle is given by $w_{1i} + w_{2i} + \sqrt{{w_{1i}}^2 + {w_{2i}}^2}$, which happens when $p_1^i = p_2^i$ and two of the other grid vertices need to be visited to cover the corresponding targets. When $N_i = 3$, the maximum length happens when $p_1^i = p_2^i$ and the other three grid vertices of the rectangle need to be visited to cover all the targets. (The details showing that the situations above are indeed the worst cases when $N_i = 2$ or 3 are intuitive and are omitted for brevity.) Finally, when $N_i \geq 4$, one can ensure that all N_i targets are covered by starting from p_1^i, traversing the perimeter of the rectangle and then moving to p_2^i. (No interior point of the grid cell needs to be visited by our assumption that no S_m is contained within the interior of any cell.) The length of the connecting trajectory in this case is bounded by $2(w_{1i} + w_{2i}) + \sqrt{{w_{1i}}^2 + {w_{2i}}^2}$. □

Remark 4.1. Note that the upper bounds in (4–4) are obtained by performing a worst-case analysis. In practice, one can design a postprocessing subroutine that uses the solution of (5–21) as well as the problem's geometry to build a feasible CETSP tour,

which would typically yield a smaller upper bound than the one given in Proposition 4.3. A heuristic algorithm (such as those in [51] and [13]) can also be used for this purpose. Alternatively, one can obtain an upper bound on the optimal CETSP tour length by solving (5–21) over a set P of grid points, where $p_0 \in P$ and $P \cap S_m \neq \emptyset$, $\forall m \in M$, with respect to the Euclidean distance between the corresponding points. □

Proposition 4.3 implies that the set of links in an optimal solution to (5–21) can be used to form a feasible CETSP tour whose length can become arbitrarily close to the optimal CETSP tour length. To that end, one can refine the underlying partitioning in such a way that the grid cell sizes belonging to I are small enough to yield an acceptably small optimality gap. One possible strategy is to solve (5–21) and obtain I, subdivide all cells in I into smaller cells, and solve the revised instance of (5–21) in a repeated fashion.

4.3.2 Cutting-Plane Generation

Subtour Elimination Constraints. Because there are an exponential number of subtour elimination constraints (4–3e), we initially relax these constraints and add those that are violated (with respect to a pre-determined violation threshold) at each node of the branch-and-bound tree. Here, we discuss the corresponding separation procedures.

To find a subtour elimination constraint (4–3e) that is violated by a solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, we use a well known separation procedure (see [52, 53]) as follows. Define a complete directed graph with the node set $\{0, \dots, n\}$ in which the capacity of an arc (i, j) equals $\overline{x}_{ij}$. For each $v \in \{1, \dots, n\}$ with $\overline{y}_v > 0$, we find the maximum flow from 0 to v, which will generate a minimum cut $(\overline{S}, S)$ where $0 \in \overline{S}$ and $v \in S$. If the capacity of this cut is less than $\overline{y}_v$, that is, if $\sum_{i \in \overline{S}} \sum_{j \in S} \overline{x}_{ij} < \overline{y}_v$, then S and v define a violated inequality (4–3e). When this procedure does not find a violated subtour inequality, $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ satisfies all the subtour elimination constraints (4–3e).

Model Tightening Inequalities. Recall that the objective function coefficients in (5–21) are pairwise minimum distances of the corresponding cells. In general, the

triangle inequality does not hold for these cost coefficients, which may result in a weak lower bound. To illustrate this point, note that there exists an optimal CETSP tour in which no two turn points cover an identical set of targets (or otherwise, bypassing either of them results in a CETSP tour that is no longer than the original one). However, it is possible that two cells on the tour cover the same set of targets in a unique optimal solution to (5–21). Consider the CETSP instance in Figure 4-7, where C_1 is the arc from point 1 to 2, and C_2 is the arc from point 2 to 3. Here, $x_{12} = 1$ at optimality. This optimal solution contains both C_1 and C_2, which cover the same target, and allows the tour to go from 1 to 3 at a distance of zero (noting that $l_{12} = 0$ because point 2 belongs to C_1 and C_2). The following proposition, which is a generalization of similar inequalities for the CTP [42], can be added to (5–21) to strengthen the obtained bound.

Proposition 4.4. *Consider two nonempty subsets C^1 and C^2 of C such that $C^1 \cap C^2 = \emptyset$. Define*

$$V(C^p) = \cup_{C_i \in C^p} \{m \in M : i \in N(m)\}, \tag{4–11}$$

for $p = 1, 2$. If $V(C^1) \subseteq V(C^2)$, then the following inequality is valid:

$$\sum_{i \in C^1 \cup C^2} y_i \leq \min\{|C^1| + |C^2| - 1, |V(C^2)|\}, \tag{4–12}$$

in the sense that solving model (5–21) augmented with (4–12) still yields a valid lower bound on the optimal CETSP tour length.

Proof. To cover all the target locations in $V(C^2)$, a shortest CETSP tour needs at most $|V(C^2)|$ turn points. Moreover, an optimal CETSP tour will not contain a turn point in every cell of $C^1 \cup C^2$ because in that case, a shorter feasible tour can be constructed by excluding any of the turn points in C^1. As a result, $(\mathbf{x}^R, \mathbf{y}^R)$ in the proof of Proposition 4.4 satisfies (4–12), which completes the proof. □

Remark 4.2. In Proposition 4.4, (4–12) can be strengthened in the following form when $|C^1| \geq 2$, $|C^2| \geq 2$, and $V(C^1) = V(C^2)$:

$$\sum_{i \in C^1 \cup C^2} y_i \leq \min\{|C^1| + |C^2| - 2, |V(C^2)|\}. \tag{4–13}$$

The reduction in the first term is due to the fact that if an optimal CETSP tour contains a turn point in every cell in C^1 (or C^2), then no cells in C^2 (or C^1) are visited by this optimal tour or otherwise, omitting those turn points in C^2 (or C^1) results in a shorter tour. Since $|C^1|$ and $|C^2|$ are both at least two, $(\mathbf{x}^R, \mathbf{y}^R)$ in the proof of Proposition 4.4 satisfies (4–13) and so the inequality obtains a valid bound in this case. Else, there exists at least one cell in both C^1 and C^2 that does not contain any turn point of an optimal CETSP tour, and (4–13) holds in this case as well. □

Given a (possibly fractional) vector $\overline{\mathbf{y}}$, a CETSP-partitioning C, a set M of targets, and the covering set $V_i = V(\{C_i\})$ for each $C_i \in C$, the separation problem of (4–12) seeks to find nonempty disjoint subsets C^1 and C^2 of C such that $V(C^1) \subseteq V(C^2)$ and $\sum_{i \in C^1 \cup C^2} \overline{y}_i > |C^1| + |C^2| - 1$. Let this decision problem be denoted by SP1. We prove that SP1 is strongly NP-complete by a reduction from the exact cover by 3-sets problem (X3C), defined as follows [54].

Definition 4.2. Problem X3C: Let $G = \{g_1, \ldots, g_{3q}\}$ and consider a collection $F = \{F_1, \ldots, F_p\}$ of 3-element subsets of G. Is there a subset $F' \subseteq F$ such that every element of G occurs in exactly one member of F'?

Theorem 4.1. *Problem SP1 is strongly NP-complete.*

Proof. SP1 clearly belongs to NP, because an input guess can be verified to satisfy the SP1 conditions in $O(n)$ time. To show that SP1 is NP-complete, we reduce any arbitrary instance (G, F) of X3C into an instance of SP1 as follows. Let $M = \{m_1, \ldots, m_{3q+p}\}$ and $C = \{C_0, \ldots, C_{p+1}\}$. For each C_i, $i = 1, \ldots, p$, define $V_i = \{m_j : g_j \in F_i\} \cup \{m_{3q+i}\}$ and let $V_{p+1} = \{m_1, \ldots, m_{3q}\}$. Also, let $\overline{y}_i = \frac{q+\epsilon}{q+1}$ for $i = 1, \ldots, p+1$, where $0 < \epsilon < \frac{1}{q+2}$. We

assert that there exists an exact cover by 3-sets of G if and only if there exist subsets C^1 and C^2 of C that solve the above instance of SP1.

$\Longrightarrow$ Suppose there exists a subset $F' \subseteq F$ such that $|F'| = q$ and every element of G occurs in exactly one member of F'. Let $C^1 = \{C_{p+1}\}$ and $C^2 = \{C_i \in C : F_i \in F'\}$. Then,

$$\sum_{i \in C^1 \cup C^2} \overline{y}_i - |C^1| - |C^2| = (q+1)\left(\frac{q+\epsilon}{q+1}\right) - q - 1 \tag{4–14}$$

$$= \epsilon - 1 > -1. \tag{4–15}$$

$\Longleftarrow$ Suppose that there exist disjoint subsets C^1 and C^2 that solve SP1. Then we have that $\sum_{i \in C^1 \cup C^2} \overline{y}_i > |C^1| + |C^2| - 1$, i.e. $\sum_{i \in C^1 \cup C^2} \frac{1-\epsilon}{q+1} < 1$. Note that C^1 cannot contain any of $C_1, \ldots, C_p$ or otherwise $V(C^1) \nsubseteq V(C^2)$. Therefore, $C^1 = \{C_{p+1}\}$ and C^2 must cover all elements in $\{m_1, \ldots, m_{3q}\}$, which requires $|C^2|$ to be at least q. Hence we have that $(|C^2| + 1)\frac{1-\epsilon}{q+1} < 1$, or equivalently, $|C^2| < \frac{q+\epsilon}{1-\epsilon}$. Because $\epsilon < \frac{1}{q+2}$, we conclude that $|C^2| \leq q$. Consequently, $|C^2| = q$ and elements of C^2 correspond to an exact cover by 3-sets of G. This completes the proof. □

We can state a similar result for the separation problem of (4–13), which we denote SP2.

Theorem 4.2. *Problem SP2 is strongly NP-complete.*

Proof. We establish a reduction from X3C to SP2 as follows. Define $M = \{m_1, \ldots, m_{3q+p+1}\}$ and $C = \{C_0, \ldots, C_{p+3}\}$. For each C_i, $i = 1, \ldots, p$, define $V_i = \{m_j : g_j \in F_i\} \cup \{m_{3q+i}\}$ and let

$$IV_{p+1} = \{m_1, \ldots, m_{3q+p}\}, \tag{4–16}$$

$$V_{p+2} = \{m_{3q+1}, \ldots, m_{3q+p+1}\}, \tag{4–17}$$

$$V_{p+3} = \{m_{3q+p+1}\}. \tag{4–18}$$

Also, let $\overline{y}_i = \frac{q+1+\epsilon}{q+3}$ for $i = 1, \ldots, p+3$, where $0 < \epsilon < \frac{2}{q+4}$. There exists an exact cover by 3-sets of G if and only if there exist two non-empty disjoint subsets C^1 and C^2 of C that solve the above instance of SP2, where $C^1 = \{C_{p+1}, C_{p+3}\}$ and C^2 consists of C_{p+2} and the elements of C corresponding to an element F_i in an exact cover by 3-sets for G. $\Longrightarrow$ Suppose there exists a subset $F' \subseteq F$ such that $|F'| = q$ and every element of G occurs in exactly one member of F'. Let $C^1 = \{C_{p+1}, C_{p+3}\}$ and $C^2 = \{C_i \in C : F_i \in F'\} \cup \{C_{p+2}\}$. Then,

$$\sum_{i \in C^1 \cup C^2} \overline{y}_i - |C^1| - |C^2| = (q+3)\left(\frac{-2+\epsilon}{q+3}\right) > -2. \tag{4–19}$$

$\Longleftarrow$ Suppose there exist disjoint subsets C^1 and C^2 of C such that $|C^1| \geq 2$, $|C^2| \geq 2$, $V(C^1) = V(C^2)$ and $\sum_{i \in C^1 \cup C^2} \overline{y}_i > |C^1| + |C^2| - 2$. The latter condition is equivalent to $\sum_{i \in C^1 \cup C^2} \frac{2-\epsilon}{q+3} < 2$. One possibility for sets C^1 and C^2 that satisfies $|C^1| \geq 2$, $|C^2| \geq 2$, and $V(C^1) = V(C^2)$ is given by $C^1 = \{C_{p+1}, C_{p+2}\}$ and $C^2 = \{C_1, \ldots, C_p\} \cup \{C_{p+3}\}$. However, this solution does not violate (4–13) unless when $p = q$, in which case the answer to the X3C instance is trivial. The only other choices for C^1 and C^2 that satisfy the above conditions are those in which $C^1 = \{C_{p+1}, C_{p+3}\}$ and $C_{p+2} \in C^2$ (or vice versa). Therefore, $C^2 \setminus \{C_{p+2}\}$ covers all elements of $\{m_1, \ldots, m_{3q}\}$, and so $|C^2| \geq q+1$. Hence, $(|C^2|+2)\frac{2-\epsilon}{q+3} < 2$, or equivalently, $|C^2| < \frac{2q+2\epsilon+2}{2-\epsilon}$. Because $\epsilon < \frac{2}{q+4}$, we have that $|C^2| < q+2$. Hence, $|C^2| = q+1$ and elements of $C^2 \setminus \{C_{p+2}\}$ establish an exact cover by 3-sets of G. □

Given the worst-case complexity of separation routines for these inequalities, we add these inequalities to the formulations only when $|C^1| = |C^2| = 1$.

4.4 Alternative Model Formulations and Algorithms

In this section, we present an alternative model in Section 4.4.1, which captures a tighter bound on minimum distances traveled in a tour, at the expense of creating additional binary decision variables. We explore subtour elimination constraints for this

model in Section 4.4.2. We then formulate a third lower-bounding model in Section 4.4.3, and demonstrate that this model is solvable by a Benders decomposition strategy.

4.4.1 Expanded Formulation

Another way of reducing the conservativeness of formulation (5–21) is to redefine the decision variables so that they capture some of the travel distances inside the cells, and hence, contribute to a tighter lower bound. To that end, we need to add sequence-related binary variables to the problem. Let $s_{ijk} = x_{ij}x_{jk}$ be a binary variable that equals one if and only if the solution to the lower bound problem consecutively visits C_i, C_j, and C_k. Moreover, let e_{ijk} $(\geq l_{ij} + l_{jk})$ denote the length of a shortest path that goes from a point in C_i to some point in C_j, and then from the same point in C_j to a point in C_k. (Section 4.5 discusses details pertaining to the calculation of these e_{ijk}-values.) Figure 4-8 illustrates the definition of e_{ijk} for an arc-based partitioning scheme. Consider the following integer programming problem.

$$\text{Min} \quad \frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} e_{ijk}s_{ijk} \tag{4–20a}$$

$$\text{s.t.} \quad \sum_{l=0}^{n} s_{lij} = \sum_{k=0}^{n} s_{ijk}, \quad \forall i = 0, \dots, n,\ j = 0, \dots, n \tag{4–20b}$$

$$y_j = \sum_{i=0}^{n}\sum_{k=0}^{n} s_{ijk}, \quad \forall j = 0, \dots, n \tag{4–20c}$$

$$\sum_{i \in N(m)} y_i \geq 1, \quad \forall m \in M \tag{4–20d}$$

$$\sum_{i \in S}\sum_{j \notin S}\sum_{k=0}^{n} s_{ijk} \geq y_v, \quad \forall \mathcal{S} \subset \{1, \dots, n\} : 3 \leq |\mathcal{S}| \leq |C| - 3 \text{ and } v \in \mathcal{S} \tag{4–20e}$$

$$s_{ijk} \in \{0, 1\}, \quad \forall i = 0, \dots, n,\ j = 0, \dots, n,\ k = 0, \dots, n \tag{4–20f}$$

$$0 \leq y_i \leq 1,\ \forall i = 1, \dots, n; \quad y_0 = 1. \tag{4–20g}$$

Proposition 4.5. *Let l^* be the optimal CETSP tour length and suppose z_{LB1} and z_{LB2} are the optimal objective function values of problems (5–21) and (4–20), respectively.*

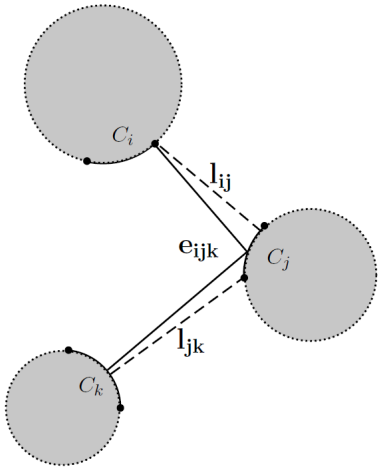


Figure 4-8. Illustration of the objective function coefficients e_{ijk} in (4–20)

Then,

$$z_{LB1} \leq z_{LB2} \leq l^*. \tag{4–21}$$

Proof. Suppose again that an optimal CETSP tour T visits (in order) the set of turn points $(p_0, \ldots, p_k)$. Similar to the proof of Proposition 4.3, let $C'_t = C_{i_t} \in C$ be an element of the CETSP-partitioning that contains p_t, for $t = 1, \ldots, k$ and define $C'_{k+1} = C'_0 = C_0$ and $i_{k+1} = i_0 = 0$. Consider $(\mathbf{x}^R, \mathbf{y}^R)$ as defined in the proof of Proposition 4.3 and let $s^R_{ijk} = x^R_{ij} x^R_{jk}$, for all $0 \leq i, j, k \leq n$. Note that $(\mathbf{s}^R, \mathbf{y}^R)$ defines a feasible solution to (4–20). Let z^R_{LB1} and z^R_{LB2} denote the objective function value of $(\mathbf{x}^R, \mathbf{y}^R)$ and $(\mathbf{s}^R, \mathbf{y}^R)$, respectively. By definition, if $s^R_{i_t, i_u, i_v} = 1$, we have

$$l_{i_t, i_u} + l_{i_u, i_v} \leq e_{i_t, i_u, i_v} \leq \sum_{j=t}^{v-1} |p_j p_{j+1}|, \tag{4–22}$$

where $p_{k+1} = p_0$. Aggregating all of the above inequalities yields $2z^R_{LB1} \leq 2z^R_{LB2} \leq 2l^*$. Therefore, $z_{LB2} \leq z^R_{LB2} \leq l^*$. Now let $(\hat{\mathbf{s}}, \hat{\mathbf{y}})$ be any feasible solution to (4–20). Corresponding to $(\hat{\mathbf{s}}, \hat{\mathbf{y}})$, we define a unique feasible solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ to (5–21), where $\hat{x}_{ij} = \sum_{k=1}^{n} \hat{s}_{ijk}$, for all $0 \leq i, j \leq n$. Using $l_{ij} + l_{jk} \leq e_{ijk}$, it is straightforward to show that

$\hat{z}_{LB1} \leq \hat{z}_{LB2}$, where $\hat{z}_{LB1}$ and $\hat{z}_{LB2}$ denote the objective function value of $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ and $(\hat{\mathbf{s}}, \hat{\mathbf{y}})$, respectively. Therefore, $z_{LB1} \leq z_{LB2}$, which concludes the proof. □

Proposition 4.6. *Denote by P^1 and P^2 the LP relaxation polytopes of formulation (5–21) and formulation (4–20), respectively. Then, there exists a mapping t such that $P^1 = proj_t(P^2)$, where $proj_t(P^2) = \{(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) = t(\mathbf{x}, \mathbf{s})$ for some $(\mathbf{x}, \mathbf{s}) \in P^2\}$.*

Proof. Suppose t is defined to map any point (**s**,**y**) in P^2 onto a point (**x**,**y**) such that $x_{ij} = \sum_{k=0}^{n} s_{ijk}$. It is then straightforward to check that any point (**x**,**y**) in the codomain of t satisfies all the constraints of (5–21). Therefore, $proj_t(P^2) \subseteq P^1$. We next prove that $P^1 \subseteq proj_t(P^2)$. Let (**x**,**y**) represent a solution in P_1. To show that the above solution is in $proj_t(P^2)$, we build a vector **s** such that (**s**,**y**) is feasible to the LP relaxation of (4–20) and $x_{ij} = \sum_{k=0}^{n} s_{ijk}$ hold true for each pair i and j. For each $C_j \in C$, define $A_j = \{i : x_{ij} > 0\}$ and $B_j = \{k : x_{jk} > 0\}$. Take $i \in A_j$ and $k \in B_j$ and let $s_{ijk} = \min\{x_{ij}, x_{jk}\}$. Then revise the values of x_{ij} and x_{jk} by subtracting s_{ijk} from them. Note that after this step, at least one of x_{ij} and x_{jk} will become zero, in which case we eliminate either i or k from the corresponding set A_j or B_j and perform the above procedure again. We can continue this procedure until both sets A_j and B_j are empty, which will eventually happen since $\sum_i x_{ij} = \sum_k x_{jk}$. This procedure results in vectors **s** and **y** that define a feasible solution to (4–20). This is true because

- $\sum_{l=0}^{n} s_{lij} = x_{ij} = \sum_{k=0}^{n} s_{ijk}$, by construction;
- $y_j = \sum_{i \in C} x_{ij} = \sum_{i \in C} \sum_{k \in C} s_{ijk}$; and
- Because $\sum_{k=0}^{n} s_{ijk} = x_{ij}$, **s** and **y** satisfy the subtour elimination constraints (4–20e).

This completes the proof. □

4.4.2 Subtour Elimination

To separate (4–20e), we can use the same procedure specified for the separation of (4–3e), with the modification of setting the capacity of arc (i, j) as $\sum_{k=0}^{n} \overline{s}_{ijk}$. For the

expanded model, though, we also consider an alternative form of subtour elimination constraints as follows.

Proposition 4.7. *Consider the following set of subtour elimination constraints for* (4–20)*.*

$$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} s_{ijk} - \sum_{i\notin\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ivk} \leq \sum_{j\in\mathcal{S}\setminus\{v\}} y_j - y_v,$$
$$\forall \mathcal{S} \subset \{1,\dots,n\} : 3 \leq |\mathcal{S}| \leq |C| - 3 \text{ and } v \in \mathcal{S}. \tag{4–23}$$

Let

$$Y^1 = \{(\mathbf{s},\mathbf{y}) : (\mathbf{s},\mathbf{y}) \text{ is feasible to (4–20)}\},$$
$$Y^2 = \{(\mathbf{s},\mathbf{y}) : (\mathbf{s},\mathbf{y}) \text{ is feasible to (4–20) with (4–20e) substituted by (4–23)}\}.$$

Moreover, let $\overline{Y}^1$ *be the LP relaxation polytope of* (4–20) *and define* $\overline{Y}^2$ *as the LP relaxation polytope of* (4–20) *with* (4–23) *instead of* (4–20e)*. Then,* $Y^1 = Y^2$ *and* $\overline{Y}^2 \subseteq \overline{Y}^1$*.*

Proof. We first prove that $\overline{Y}^2 \subseteq \overline{Y}^1$ by showing that every (possibly fractional) solution to $\overline{Y}^2$ satisfies (4–20e) as well. By rearranging (4–23) we obtain

$$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} s_{ijk} + 2y_v \leq \sum_{j\in\mathcal{S}} y_j + \sum_{i\notin\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ivk}. \tag{4–24}$$

Note that $\sum_{i\notin\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ivk} \leq \sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk}$ and $\sum_{j\in\mathcal{S}} y_j = \sum_{i=0}^{n}\sum_{j\in\mathcal{S}}\sum_{k=0}^{n} s_{ijk}$. Therefore, we have

$$2y_v \leq \sum_{i=0}^{n}\sum_{j\in\mathcal{S}}\sum_{k=0}^{n} s_{ijk} + \sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk} - \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} s_{ijk}. \tag{4–25}$$

The above can be restated as

$$2y_v \leq \sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} s_{ijk} + \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk} + 2\sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk} \tag{4–26a}$$
$$= \sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k=0}^{n} s_{ijk} + \sum_{i=0}^{n}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk}. \tag{4–26b}$$

Note that the first term of (4–26b) represents the number of times a (possibly fractional) tour enters set $\mathcal{S}$, and the second term represents the number of times a tour exits set $\mathcal{S}$. By (4–20b), these values must be equal, and so (4–26b) reduces to $2\sum_{i=0}^{n}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk}$. But since

$$\sum_{i=0}^{n}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ijk} = \sum_{i\in\mathcal{S}}\sum_{j\notin\mathcal{S}}\sum_{k=0}^{n} s_{ijk}, \tag{4–27}$$

we have that (4–26) implies (4–20e).

We now prove that $Y^1 = Y^2$. We showed above that if a solution satisfies (4–23), it also satisfies (4–20e). Therefore, $Y^2 \subseteq Y^1$. Now consider an integer solution $(\overline{\mathbf{s}}, \overline{\mathbf{y}})$ in Y^1, which represents a tour over a subset of C that intersects p_0. We show that $(\overline{\mathbf{s}}, \overline{\mathbf{y}})$ satisfies (4–23) as well. Suppose $\mathcal{S} \subset C$ is chosen such that $0 \notin \mathcal{S}$ and $v \in \mathcal{S}$. Consider the following cases.

- $\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} \geq 1$: Note that

  $$\sum_{j\in\mathcal{S}} \overline{y}_j = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} + \sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} + \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} \overline{s}_{ijk} + \sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} \overline{s}_{ijk}. \tag{4–28}$$

  Moreover, we have

  $$\sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} \geq 1 \tag{4–29}$$

  $$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} \overline{s}_{ijk} \geq 1 \tag{4–30}$$

  $$\sum_{i\notin\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\notin\mathcal{S}} \overline{s}_{ijk} \geq 0. \tag{4–31}$$

  Therefore,

  $$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} \leq \sum_{j\in\mathcal{S}} \overline{y}_j - 2 \leq \sum_{j\in\mathcal{S}} \overline{y}_j - 2y_v. \tag{4–32}$$

  and (4–23) holds.

- $\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} = 0$ and $\overline{y}_v = 0$: (4–23) clearly holds in this case.
- $\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{S}} \overline{s}_{ijk} = 0$ and $\overline{y}_v = 1$: In this case, if $\sum_{j\in\mathcal{S}\setminus\{v\}} y_j \geq 1$ then (4–23) is valid. If $\sum_{j\in\mathcal{S}\setminus\{v\}} y_j = 0$, then $\sum_{i\notin\mathcal{S}}\sum_{k\notin\mathcal{S}} s_{ivk} = 1$, and (4–23) remains valid.

□

Note that since $Y_1 = Y_2$, the same separation procedure that generates (4–20e) can also be used to generate (4–23).

We conclude this section by presenting stronger subtour elimination constraints for (4–20). Suppose $\mathcal{S} \subset \{1, \dots, n\}$ and let $\overline{\mathcal{S}} = \{0, \dots, n\} \setminus \mathcal{S}$. Define $\widetilde{M}(\mathcal{S}) = M \setminus \{m \in M : \overline{\mathcal{S}} \cap N(m) \neq \emptyset\}$, i.e., $\widetilde{M}(\mathcal{S})$ is the set of targets not covered by a cell in $\overline{\mathcal{S}}$. Let $\widetilde{C}(\mathcal{S})$ be a smallest cardinality subset of $\mathcal{S}$ that covers all targets in $\widetilde{M}(\mathcal{S})$. Then

$$\sum_{i \in \mathcal{S}} y_i \geq |\widetilde{C}(\mathcal{S})| \tag{4–33}$$

is valid for (4–20). If $|\widetilde{C}(\mathcal{S})| \geq 1$, then (4–33) implies the validity of the following stronger subtour elimination constraint for (4–20):

$$\sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} \sum_{k=0}^{n} s_{ijk} \geq 1. \tag{4–34}$$

Note that (4–34) is a modified version of similar connectivity inequalities for the CTP [42]. When $|\widetilde{C}(\mathcal{S})| \geq 2$, (4–34) can be further strengthened as follows.

$$\sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} \sum_{k \in \mathcal{S}} s_{ijk} + \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \sum_{k \notin \mathcal{S}} s_{ijk} + \sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} \sum_{k \notin \mathcal{S}} s_{ijk} \geq 2. \tag{4–35}$$

Therefore, when an inequality (4–23) is generated in our separation routine, we first check to see if can be strengthened into the form (4–34) or (4–35) before adding the cut to the model.

4.4.3 Alternative Formulation

Formulations (5–21) and (4–20) contain $O(n^2)$ and $O(n^3)$ binary variables, respectively, which possibly makes them intractable for CETSP-partitionings having a large number of cells. We develop an alternative MIP formulation whose number of binary variables does not depend on n.

Note that any feasible solution to (5–21) can be characterized by two sets of decisions: (a) the order in which the neighborhood sets S_m are visited, and (b) for each S_m, which cell represents target m on the tour. Given any particular order of visiting the neighborhood sets, the problem of optimally identifying a representative cell for each target can be solved by solving a shortest path problem as follows. For a CETSP-partitioning $C = \{C_0, \ldots, C_n\}$ define set Q as

$$Q = \{(i, m) : C_i \in C,\ m \in M,\ i \in N(m)\} \cup \{\psi, \tau\}, \tag{4–36}$$

where ψ and τ respectively serve as source and destination nodes in the graph. An ordered pair (i, m) belongs to Q for each cell $C_i \in C$ and every $m \in M$ such that $C_i \cap S_m \neq \emptyset$. For convenience, we also describe the elements of Q by Greek letters, with components $\hat{i}(\bullet)$ and $\hat{m}(\bullet)$, where $\hat{i}(\psi) = \hat{i}(\tau) = 0$ and $\hat{m}(\psi) = \hat{m}(\tau) = p_0$. The distance between any two elements δ and σ in Q is defined as $d_{\delta\sigma} = l_{\hat{i}(\delta),\hat{i}(\sigma)}$. Let u_{lm} be a binary variable that indicates whether target l is visited immediately before (or simultaneously with) target m. We seek a shortest tour that starts from the depot, visits a cell in every neighborhood set (in the order determined by the u-variables), and returns to the depot. This is equivalent to finding a shortest path (with respect to arc costs d) from ψ to τ in a graph $G(\mathbf{u}) = (Q, A(\mathbf{u}))$, where $A(\mathbf{u}) = \{(\delta, \sigma) : \delta, \sigma \in Q,$ and $u_{\hat{m}(\delta),\hat{m}(\sigma)} = 1\}$. To model this shortest path, we introduce nonnegative flow variables $f_{\delta\sigma}$ along with the necessary flow-balance constraints. The corresponding formulation is stated below, where $M' = M \cup \{p_0\}$.

$$\text{Min} \quad \sum_{\delta \in Q} \sum_{\sigma \in Q} d_{\delta\sigma} f_{\delta\sigma} \tag{4–37a}$$

$$\text{s.t.} \quad \sum_{m \in M'} u_{ml} = \sum_{m \in M'} u_{lm} = 1 \quad \forall l \in M' \tag{4–37b}$$

$$\sum_{l \in \mathcal{S}} \sum_{m \in \mathcal{S}} u_{lm} \leq |\mathcal{S}| - 1, \quad \forall \mathcal{S} \subset M',\ 2 \leq |\mathcal{S}| \leq |M'| - 2 \tag{4–37c}$$

$$f_{\delta\sigma} \leq u_{\hat{m}(\delta),\hat{m}(\sigma)} \quad \forall \delta \in Q,\ \sigma \in Q \tag{4–37d}$$

$$\sum_{\sigma \in Q} f_{\sigma\delta} - \sum_{\sigma \in Q} f_{\delta\sigma} = 0, \quad \forall \delta \in Q \setminus \{\psi, \tau\} \tag{4–37e}$$

$$\sum_{\sigma \in Q} f_{\sigma\tau} = 1, \tag{4–37f}$$

$$f_{\delta\sigma} \geq 0, \quad \forall \delta \in Q,\ \sigma \in Q \tag{4–37g}$$

$$u_{lm} \in \{0, 1\}, \quad \forall l \in M',\ m \in M'. \tag{4–37h}$$

In the above formulation, constraints (4–37b), (4–37c), and (4–37h) impose a TSP tour over set M', while (4–37d)–(4–37g) model the corresponding CETSP tour as a shortest path from ψ to τ in $G(\mathbf{u})$. Note that when $|C^1| = |C^2| = 1$, an equivalent form of (4–12) can be applied to formulation (4–37) by adding $f_{\delta\sigma} = f_{\sigma\delta} = 0$ to the model, where $\hat{i}(\delta) \in C^1$ and $\hat{i}(\sigma) \in C^2$.

Proposition 4.8. *Suppose that the optimal CETSP tour length is l^*, and consider an optimal solution $(\mathbf{u}^*, \mathbf{f}^*)$ to (4–37) with objective function value z_{LB3}. Define the set of turn cells as $l' = \{\delta \in Q \setminus \{\psi, \tau\} : \sum_{\sigma \in Q} f^*_{\delta\sigma} = 1\}$. Then,*

$$z_{LB3} \leq l^* \leq z_{LB3} + \sum_{\delta \in l'} h'_\delta, \tag{4–38}$$

where

$$h'_\delta = \begin{cases} 2r_{\hat{i}(\delta)} \sin\left(\frac{\alpha_{\hat{i}(\delta)}}{2}\right) & \text{if } C_{\hat{i}(\delta)} \subseteq S_{\hat{m}(\delta)}, \\ 4r_{\hat{i}(\delta)} \sin\left(\frac{\alpha_{\hat{i}(\delta)}}{2}\right) & \text{if } C_{\hat{i}(\delta)} \nsubseteq S_{\hat{m}(\delta)}, \end{cases} \tag{4–39}$$

for an arc-based CETSP-partitioning, and

$$h'_\delta = 2\sqrt{w_{1,\hat{i}(\delta)}{}^2 + w_{2,\hat{i}(\delta)}{}^2}, \tag{4–40}$$

for a grid-based CETSP-partitioning.

Proof. Consider an optimal CETSP tour characterized by an ordered set of turn points $(p_0, \ldots, p_k)$ with a corresponding solution $(\mathbf{x}^R, \mathbf{y}^R)$ as constructed in the proof of Proposition 4.3. Suppose that this solution corresponds to a tour $C_0 \rightarrow \ldots \rightarrow$

$C_r \rightarrow C_0$. For each $i = 1, \ldots, r$ define $M^i = \{m \in M : m \notin \cup_{h=1}^{i-1} M^h$ and $\exists j \in \{1, \ldots, k\}$ such that $p_j \in S_m \cap C_i\}$. Note that $M^i \neq \emptyset$ for all $i = 1, \ldots, r$ or otherwise, a shorter CETSP tour can be obtained by excluding all turn points in C_i from T. Now suppose $M^i = \{m_1^i, \ldots, m_{|M^i|}^i\}$, $\forall i = 1, \ldots, r$, and consider the following feasible solution to (4–37).

- $f_{\psi,(1,m_1^1)} = 1$,
- $f_{(i,m_j^i),(i,m_{j+1}^i)} = 1$ for all $1 \leq i \leq r, j = 1, \ldots, |M^i| - 1$,
- $f_{(i,m_{|M^i|}^i),(i+1,m_1^{i+1})} = 1$ for $1 \leq i \leq r - 1$,
- $f_{(r,m_{|M^r|}^r),\tau} = 1$,
- $u_{lm} = 1$ for $l, m \in M'$, if and only if $l \neq m$ and there exist δ and σ in Q such that $\hat{m}(\delta) = l$, $\hat{m}(\sigma) = m$, and $f_{\delta\sigma} = 1$.

The objective function value of this solution equals that of $(\mathbf{x}^R, \mathbf{y}^R)$. Therefore, $z_{LB3} \leq z^R \leq l^*$. Validity of the upper bound follows from a similar discussion to that of Proposition 4.3, noting that in this case every visit to C_i requires a visit to at most one middle point. □

Proposition 4.8 implies that the optimal objective function value of (4–37) can be used as a lower bound on the optimal CETSP tour length. Note that regardless of the size n of the underlying CETSP-partitioning, problem (4–37) contains $O(|M|^2)$ binary variables.

We next describe a Benders decomposition algorithm for solving (4–37). To that end, suppose $\overline{d}_{lm}$ equals the minimum distance between two neighborhood sets S_l and S_m, for $l, m \in M'$. Formulation (4–37) can be equivalently stated as

$$\text{Min} \quad \sum_{l \in M'} \sum_{m \in M'} \overline{d}_{lm} u_{lm} + \sum_{\delta \in Q} \sum_{\sigma \in Q} (d_{\delta\sigma} - \overline{d}_{\hat{m}(\delta),\hat{m}(\sigma)}) f_{\delta\sigma} \tag{4–41a}$$

$$\text{s.t.} \quad \text{Constraints (4–37b)–(4–37h).} \tag{4–41b}$$

We reformulate (4–41) as:

$$\text{Min} \quad \sum_{l \in M'} \sum_{m \in M'} \overline{d}_{lm} u_{lm} + \theta(\mathbf{u}) \tag{4–42a}$$

$$\text{s.t.} \quad \text{Constraints (4–37b), (4–37c), and (4–37h),} \tag{4–42b}$$

where

$$\theta(\mathbf{u}) = \text{Min} \quad \sum_{\delta \in Q} \sum_{\sigma \in Q} (d_{\delta\sigma} - \overline{d}_{\hat{m}(\delta),\hat{m}(\sigma)}) f_{\delta\sigma} \tag{4–43a}$$

$$\text{s.t.} \quad \text{Constraints (4–37d)–(4–37g).} \tag{4–43b}$$

In the Benders decomposition framework, we let (4–42) serve as a *master problem* in which $\theta(\mathbf{u})$ is replaced by a value function variable θ. Then, given a fixed value of $\mathbf{u}$, (4–43) serves as the *subproblem*. Solving (4–43) either proves the optimality of $\mathbf{u}$, or yields a Benders (optimality) inequality. Given $\overline{\mathbf{u}}$, define π as a $|Q|$-vector that contains the dual values associated with this shortest path problem, i.e., π_δ denotes the dual value associated with constraint (4–37e) for $\delta \in Q \setminus \{\psi, \tau\}$, π_τ is the dual value associated with constraint (4–37f), and $\pi_\psi \equiv 0$. Such dual values indicate the length of a shortest path from ψ to any other node in $G(\overline{\mathbf{u}})$. For any $\delta \in Q$ and $\sigma \in Q$, define $-\gamma_{\delta\sigma}$ as the dual value associated with constraint (4–37d). Hence, $\gamma_{\delta\sigma} = \max\{0, \pi_\sigma - \left(d_{\delta\sigma} - \overline{d}_{\hat{m}(\delta),\hat{m}(\sigma)}\right) - \pi_\delta\}$, and let

$$\eta_{lm} = \sum_{\delta \in Q : \hat{m}(\delta) = l} \quad \sum_{\sigma \in Q : \hat{m}(\sigma) = m} \gamma_{\delta\sigma}. \tag{4–44}$$

Note that η_{lm} is nonnegative and can take positive values only for $(l, m) \in U^0 = \{(l, m) : l \in M', m \in M', \text{ and } \overline{u}_{lm} = 0\}$. Hence, the corresponding Benders inequality can be written as

$$\theta + \sum_{(l,m) \in U^0} \overline{\eta}_{lm} u_{lm} \geq \pi_\tau, \tag{4–45}$$

where $\overline{\eta}_{lm} = \min\{\eta_{lm}, \pi_\tau\}$. These reduced coefficients are valid due to the nonnegativity of η_{lm}, $\forall (l, m) \in U^0$, and the fact that $\theta \geq 0$. This concept can be used to further tighten (4–45). Noting that (4–45) is valid at $\mathbf{u} = \overline{\mathbf{u}}$, regardless of the coefficients of the u-variables in U^0, we need to ensure that (4–45) is valid for $\mathbf{u}' \neq \overline{\mathbf{u}}$. Note that $\mathbf{u}'$ contains at least two variables in U^0 that equal one. Let $\rho = \min_{(l,m)\in U^0}\{\overline{\eta}_{lm}\}$. Suppose U^{01} denotes a subset of U^0 whose elements satisfy $\overline{\eta}_{lm} \geq \pi(\tau) - \rho$, and define $U^{02} = U^0 \setminus U^{01}$. If $\rho \leq \frac{\pi_\tau}{2}$, then (4–45) can be strengthened as

$$\theta + \sum_{(l,m)\in U^{01}} (\pi_\tau - \rho) u_{lm} + \sum_{(l,m)\in U^{02}} \eta_{lm} u_{lm} \geq \pi_\tau; \tag{4–46}$$

otherwise, the following inequality is valid for (4–42).

$$\theta + \sum_{(l,m)\in U^0} \frac{\pi_\tau}{2} u_{lm} \geq \pi_\tau. \tag{4–47}$$

4.5 Calculation of Distance Values

We describe how to calculate the cost coefficients l_{ij} and e_{ijk} for the cells of a given CETSP-partitioning. Two cells C_i and C_j are said to be *regularly placed* with respect to each other if both of the following conditions hold:

1. $a_i + w_{1i} \leq a_j$ or $a_i \geq a_j + w_{1j}$, and
2. $b_i + w_{2i} \leq b_j$ or $b_i \geq b_j + w_{2j}$.

If C_i and C_j are not regularly placed with respect to each other and $a_j - w_{1i} < a_i < a_j + w_{1j}$, then $l_{ij} = \max\{b_i - b_j - w_{2j}, b_j - b_i - w_{2i}\}$. If $b_j - w_{2i} < b_i < b_j + w_{2j}$, then $l_{ij} = \max\{a_i - a_j - w_{1j}, a_j - a_i - w_{1i}\}$. On the other hand, when C_i and C_j are regularly placed with respect to each other, it can be easily shown that there exists a vertex v of C_i and a vertex v' of C_j such that the line segment (vv') is the shortest line segment that connects C_i and C_j and so $l_{ij} = |vv'|$. As a result, calculating each l_{ij}-value can be done in $O(1)$ time.

Now let $(v_1 v_2)$ be the shortest line segment connecting two regularly placed cells C_i and C_j and $(v_3 v_4)$ be the shortest line segment connecting cells C_j and C_k, which are also regularly placed with respect to each other. Hence, $l_{ij} = |v_1 v_2|$ and $l_{jk} = |v_3 v_4|$. To evaluate e_{ijk}, we consider the following three cases.

Case 1.: v_2 and v_3 are the same vertex of C_j. In this case, $e_{ijk} = l_{ij} + l_{jk}$.

Case 2.: v_2 and v_3 are on the same edge of C_j. Let $v_i = (\omega_{1i}, \omega_{2i})$ for $i = 1, \dots, 4$. Since v_2 and v_3 are on the same side of cell C_j, either $\omega_{12} = \omega_{13}$ or $\omega_{22} = \omega_{23}$. Without loss of generality, suppose $\omega_{22} = \omega_{23} = \overline{\omega}$. The minimum distance from any point in C_i to any point in C_j and then to any point in C_k is obtained by going from v_1 to some point $(\mu, \overline{\omega})$ on the line segment between v_2 and v_3 and from there to v_4. Therefore, the corresponding optimization problem is as follows.

$$\min_{\omega_{12} \le \mu \le \omega_{13}} \left((\mu - \omega_{11})^2 + (\overline{\omega} - \omega_{21})^2\right)^{\frac{1}{2}} + \left((\mu - \omega_{14})^2 + (\overline{\omega} - \omega_{24})^2\right)^{\frac{1}{2}} \tag{4–48}$$

Since the objective function is strictly convex, the optimal value of μ is either ω_{12}, ω_{13}, or where μ is the unconstrained minimizer of (4–48), such that

$$\frac{\mu - \omega_{11}}{\omega_{14} - \mu} = \left| \frac{\overline{\omega} - \omega_{21}}{\overline{\omega} - \omega_{24}} \right|, \text{ assuming } \overline{\omega} \neq \omega_{24}.$$

Case 3.: v_2 and v_3 are diagonally opposite each other in C_j. In this case, if the line segment (v_1, v_4) passes through C_j, then $e_{ijk} = |v_1 v_4|$. Otherwise, $e = |v_1 v'| + |v' v_4|$, where v' is one of two vertices of C_j other than v_2 and v_3 (whichever is a minimizer of distance), because this path dominates all other paths that do not pass through a vertex of C_j. Therefore, calculating e_{ijk}-coefficients can be done in $O(1)$ time.

Unlike the grid-based CETSP-partitioning, computing objective function coefficients for an arc-based CETSP-partitioning does not appear to be easy. This is because the corresponding distance minimization problem in general is not convex. One way to deal with this difficulty is to numerically calculate all the minimum distances between any two arcs prior to solving the mathematical programming problems. We can also

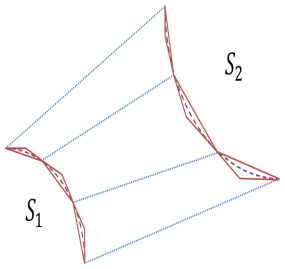


Figure 4-9. Obtaining a lower bound on the distance between two arcs

use an easily-computable valid lower bound on the minimum distance. To obtain this lower bound, one can calculate the minimum distance between two minimal triangles (outer-approximations) that contain the two arcs. This is equivalent to solving a convex optimization problem. The corresponding triangles can be obtained using the tangent lines at the two endpoints, plus the arc's chord. Moreover, we can divide each of the two arcs into several smaller portions to obtain a collection of smaller triangles, as illustrated in Figure 4-9. The minimum distance between any pair of triangles representing different arcs provides a valid lower bound on the minimum distance between the two arcs. This lower bound can become arbitrarily tight by increasing the number of triangles.

4.6 Computational Experiments

To examine the efficiency of the proposed formulations and algorithms, we generate random CETSP instances on which we obtain upper and lower bounds for the optimal CETSP tour length. All algorithms are implemented in C++ programming language and compiled using Microsoft Visual Studio 2008. Integer programming instances are solved using CPLEX version 12.2 via ILOG Concert Technology 2.9 on a PC with an Intel Core2 Quad processor Q9500 and 4 GB of memory, running Windows 7.

All CETSP instances are generated according to the following procedure. The $|M|$ target locations and the depot are chosen randomly on a rectangle of length 16 and width 10. The neighborhood set for all targets are discs of identical radius r (which we vary in our computational tests). For each disc, we specify the intersection of its boundary and the convex hull of the target locations and the depot, and divide this

intersection into N' smaller arcs, which are initial elements of the partitioning set C. Therefore, the initial CETSP-partitioning contains $|M|N' + 1$ cells.

We first provide a comparison between the performance of different methods of obtaining a lower bound on the optimal CETSP tour length. For brevity, we will refer to formulations (5–21), (4–20) (with subtour elimination constraints (4–23)), and (4–41) as LB1, LB2, and LB3, respectively. The subtour elimination constraints for all three models are initially relaxed. At each node of the branch-and-bound tree, we use the separation procedure explained in Section 4.3 to find violated subtour inequalities. If the violation magnitude is greater than or equal to 0.5 for a subtour elimination constraint, we add that inequality to the linear programming relaxation of the corresponding node.

The results of running all three models over a set of ten small test instances are reported in Tables 4-1 and 4-2. These instances contain six target locations with the neighborhood set of each location defined as a disc of radius $r = 0.25$, and each arc is partitioned into $N' = 4$ smaller ones. A 1500-second limit is imposed on the running time of all methods. The **Lower Bound** columns report the lower bound that is obtained via each method, while the **Upper Bound** columns report the upper bound provided by (4–4) (for LB1 and LB2) and (4–38) (for LB3). We also report the average absolute gap (difference) between the lower and upper bounds of all methods in the **Avg. Gap** row. We observe that while LB2 always provides the best lower bound, it generally requires a significantly longer time to solve than the other two methods. Note that LB1 consumes roughly 30 times the CPU time required to solve LB3, and it provides a lower bound that is dominated by that of LB3. The fast running time of LB3 is evidently due at least in part to the low number of subtour cuts that are required to solve the problem.

Table 4-3 compares the running time of formulations LB1 and LB3 for ten 8- and 10-node instances for which LB2 is too large to be practically useful. Overall, these results demonstrate that LB3 is significantly easier to solve than LB1, and once again, there appears to be direct correlation between the number of subtour cuts generated

(many fewer for LB3 than for LB1). The lower bounds obtained by these algorithms are not displayed in the table, because all lower bounds were the same except for three instances for $r = 0.25$ and six for $r = 0.5$. For eight out of these nine instances we found that LB3 provides a slightly better lower bound than LB1. In one instance, the lower bound obtained from LB1 was better than that of LB3 with a difference of 0.052.

Table 4-1. Comparing lower bound formulations on six-node instances

	LB1		LB2		LB3	
Instance	Subtour Cuts	Time (s)	Subtour Cuts	Time (s)	Subtour Cuts	Time (s)
CETSP-6-01	261	4.6	145	263.3	27	0.8
CETSP-6-02	357	22.7	205	442.3	21	0.8
CETSP-6-03	448	26.3	195	378.4	14	1.6
CETSP-6-04	127	15.4	0	21.6	14	0.5
CETSP-6-05	156	17.5	0	22.6	12	1.2
CETSP-6-06	173	20.2	0	22.5	7	0.8
CETSP-6-07	233	23.7	216	364.7	10	0.7
CETSP-6-08	269	21.2	226	429.0	23	1.5
CETSP-6-09	115	12.5	0	25.9	19	0.8
CETSP-6-10	496	28.8	372	767.2	28	0.9
Averages	**263.5**	**19.2**	**135.9**	**273.7**	**17.5**	**0.9**

Table 4-4 demonstrates the efficiency of the Benders decomposition algorithm for LB3 in solving 80 random instances. For each value of $|M|$, we solve two instances associated with setting $r = 0.25$ and $r = 0.5$. The number of arcs per neighborhood set is set to be $N' = 4$ for each instance. We present the running time (in CPU seconds) required by the Benders decomposition algorithm, and for solving formulation (4–41) directly using CPLEX. We report the number of added subtour elimination constraints for both methods. For the Benders decomposition algorithm, we also report the number of Benders inequalities needed to solve the master problem to optimality. (Note that the average CPU times factor in 1500 seconds for the case in which a problem instance is not solved within the time limit.) The results demonstrate that using Benders decomposition dramatically improves the model's solvability. Using (4–39), we can also obtain upper bounds on the optimal CETSP tour length for all instances. The average difference between the upper and lower bounds for different instance sizes are specified below, where the first, second, and third components denote $|M|$, r and the average

Table 4-2. Lower and upper bounds obtained on six-node instances

	LB1		LB2		LB3	
Instance	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound
CETSP-6-01	33.1604	34.2768	33.4994	34.6158	33.1604	34.2768
CETSP-6-02	27.0611	27.8568	27.3521	28.1474	27.0615	27.8568
CETSP-6-03	24.7048	25.6439	25.0509	25.99	24.7048	25.6439
CETSP-6-04	36.1009	37.0838	36.4295	37.4124	36.1009	37.0838
CETSP-6-05	21.3865	22.5598	21.5978	22.771	21.3865	22.9133
CETSP-6-06	27.1195	28.1387	27.5114	28.5306	27.1195	28.1387
CETSP-6-07	33.9453	35.0102	34.3707	35.4329	33.948	35.0102
CETSP-6-08	23.4709	24.514	23.8133	24.8564	23.4709	24.514
CETSP-6-09	27.8231	29.0243	28.137	29.3381	27.8231	29.0243
CETSP-6-10	33.6742	34.9533	34.1952	35.4743	33.6742	34.9533
Avg. Gap	**1.0614**		**1.0611**		**1.0965**	

Table 4-3. Comparing formulations LB1 and LB3

		$r = 0.25$				$r = 0.5$			
		LB1		LB3		LB1		LB3	
Instance	$\|M\|$	Subtour Cuts	Time (s)	Subtour Cuts	Time (s)	Subtour Cuts	Time (s)	Subtour Cuts	Time (s)
CETSP-8-01	8	603	25.9	35	1.2	346	14.7	38	1.9
CETSP-8-02	8	700	37.1	49	2.5	760	31.9	83	6.5
CETSP-8-03	8	210	8.1	15	0.5	314	11.4	15	0.7
CETSP-8-04	8	619	19.4	46	2.7	189	3.5	34	3.0
CETSP-8-05	8	1596	64.4	73	2.5	663	25.0	67	4.6
CETSP-8-06	8	232	9.0	33	1.0	341	12.9	65	3.7
CETSP-8-07	8	249	11.7	54	1.0	196	5.6	61	1.2
CETSP-8-08	8	387	22.3	52	1.2	917	35.2	60	1.7
CETSP-8-09	8	254	12.7	42	0.8	137	6.1	44	1.2
CETSP-8-10	8	379	17.3	36	2.1	374	21.3	62	8.3
Averages		**522.9**	**22.8**	**43.5**	**1.5**	**423.7**	**16.8**	**52.9**	**3.3**
CETSP-10-01	10	1231	92.2	30	1.6	1906	192.3	41	3.7
CETSP-10-02	10	972	97.1	93	5.7	1698	215.3	122	12.5
CETSP-10-03	10	938	52.5	68	5.6	419	45.0	79	12.7
CETSP-10-04	10	1812	231.5	50	4.1	1008	78.2	51	4.8
CETSP-10-05	10	890	73.7	89	1.9	638	53.3	166	19.6
CETSP-10-06	10	687	56.7	59	1.6	1887	227.0	68	13.2
CETSP-10-07	10	415	24.5	38	2.5	564	63.3	38	6.3
CETSP-10-08	10	2598	196.3	50	7.2	530	53.5	104	18.4
CETSP-10-09	10	279	15.4	34	1.4	385	24.6	77	4.7
CETSP-10-10	10	509	52.3	24	1.2	309	23.7	90	18.2
Averages		**1033.1**	**89.2**	**53.5**	**3.3**	**934.4**	**97.6**	**83.6**	**11.4**

absolute gap, respectively: $(14, 0.25, 3.80)$, $(14, 0.5, 7.78)$, $(16, 0.25, 4.38)$, $(16, 0.5, 9.20)$, $(18, 0.25, 5.20)$, $(18, 0.5, 11.04)$, $(20, 0.25, 5.79)$, and $(20, 0.5, 12.68)$. Note that a few instances were not solved within the time limit by either algorithm. The lower bound that we report for these instances is given by the lower bound on z_{LB3} after 1500 seconds. The upper bound for these instances employs Proposition 4.8, with z_{LB3} given by the objective corresponding to the best feasible solution found for LB3 after 1500 seconds.

Figure 4-10 illustrates the convergence of lower and upper bounds for an instance of size $|M| = 12$ and $r = 0.25$. In each iteration, we obtain lower and upper bounds on the optimal CETSP tour length using the Benders decomposition algorithm that was proposed in Section 4.4.3. Figure 4-10 also shows the solution time of each iteration for the upper and lower bound problems.

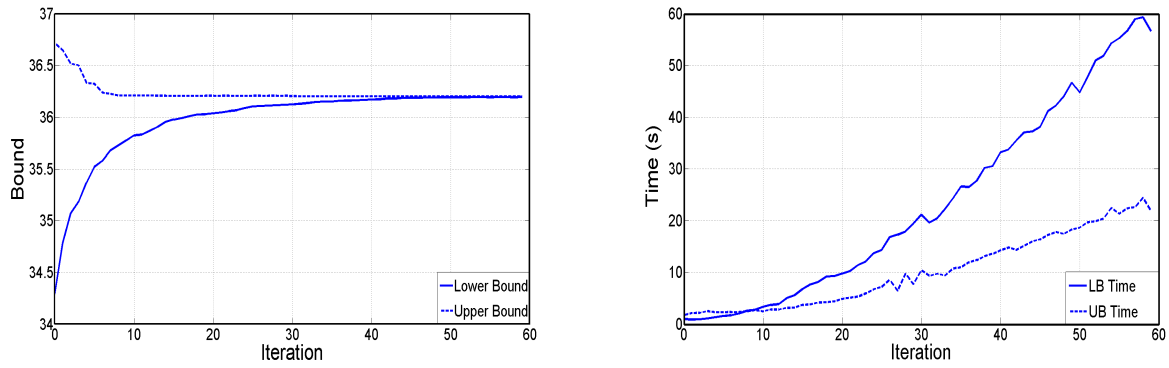


Figure 4-10. Results for the iterative solution method

Table 4-4. Performance of Benders Decomposition

		$r = 0.25$						$r = 0.5$						
		Monolithic			Benders			Monolithic			Benders			
Instance	$\|M\|$	Time (s)	Gap	Subtour Cuts	Time (s)	Benders Cuts	Subtour Cuts	Time (s)	Gap	Subtour Cuts	Time (s)	Gap	Benders Cuts	Subtour Cuts
CETSP-14-01	14	49.7	0.0%	148	2.5	88	150	353.7	0.0%	216	5.3	0.0%	257	202
CETSP-14-02	14	10.7	0.0%	80	1.5	21	92	24.1	0.0%	160	2.2	0.0%	49	88
CETSP-14-03	14	10.5	0.0%	209	6.4	48	140	76.4	0.0%	199	11.7	0.0%	223	213
CETSP-14-04	14	10.0	0.0%	139	2.6	28	154	103.9	0.0%	360	4.5	0.0%	137	243
CETSP-14-05	14	5.6	0.0%	130	2.1	20	98	70.3	0.0%	211	3.3	0.0%	69	174
CETSP-14-06	14	17.2	0.0%	115	3.7	7	141	77.4	0.0%	183	2.5	0.0%	52	213
CETSP-14-07	14	5.3	0.0%	66	2.4	8	65	71.0	0.0%	251	2.8	0.0%	37	106
CETSP-14-08	14	21.3	0.0%	110	6.3	26	167	71.0	0.0%	277	7.4	0.0%	242	196
CETSP-14-09	14	11.8	0.0%	94	2.9	14	97	21.1	0.0%	106	1.9	0.0%	42	111
CETSP-14-10	14	7.9	0.0%	168	3.4	21	148	27.1	0.0%	177	6.0	0.0%	154	174
Averages		**15.0**	**0.0**%	**125.9**	**3.4**	**28.1**	**125.2**	**89.6**	**0.0**%	**214.0**	**4.8**	**0.0**%	**126.2**	**172.0**
CETSP-16-01	16	89.3	0.0%	355	5.0	166	231	709.4	0.0%	323	8.3	0.0%	294	239
CETSP-16-02	16	22.7	0.0%	228	1.8	18	193	1060.8	0.0%	652	8.7	0.0%	238	302
CETSP-16-03	16	11.7	0.0%	153	6.7	12	175	111.2	0.0%	198	9.7	0.0%	168	244
CETSP-16-04	16	23.5	0.0%	258	3.0	25	289	250.3	0.0%	460	13.8	0.0%	384	460
CETSP-16-05	16	61.0	0.0%	311	3.2	47	209	577.4	0.0%	554	8.4	0.0%	202	255
CETSP-16-06	16	17.1	0.0%	173	4.0	47	251	802.6	0.0%	480	14.2	0.0%	337	334
CETSP-16-07	16	40.2	0.0%	272	7.5	30	172	213.3	0.0%	297	15.6	0.0%	450	248
CETSP-16-08	16	7.8	0.0%	149	2.0	8	145	29.3	0.0%	176	7.2	0.0%	50	196
CETSP-16-09	16	218.0	0.0%	262	11.6	203	246	1500.9	73.7%	1046	520.1	0.0%	5365	521
CETSP-16-10	16	52.0	0.0%	162	5.2	71	158	331.4	0.0%	357	15.8	0.0%	339	199
Averages		**54.3**	**0.0**%	**232.3**	**5.0**	**62.7**	**206.9**	**558.7**	**7.4**%	**454.3**	**62.2**	**0.0**%	**782.7**	**299.8**
CETSP-18-01	18	296.0	0.0%	556	8.1	145	458	1500.8	15.2%	745	35.6	0.0%	790	545
CETSP-18-02	18	269.2	0.0%	353	12.8	321	265	1500.8	4.2%	467	202.0	0.0%	2597	413
CETSP-18-03	18	20.7	0.0%	182	9.1	66	230	1500.8	17.4%	622	102.9	0.0%	1536	550
CETSP-18-04	18	49.9	0.0%	201	2.7	15	127	382.5	0.0%	291	11.0	0.0%	132	146
CETSP-18-05	18	121.7	0.0%	205	4.7	65	165	862.7	0.0%	427	28.0	0.0%	483	230
CETSP-18-06	18	1500.9	10.0%	596	13.4	259	444	1501.5	80.1%	884	173.9	0.0%	2142	718
CETSP-18-07	18	55.7	0.0%	301	4.2	39	292	140.4	0.0%	283	11.4	0.0%	108	211
CETSP-18-08	18	75.7	0.0%	293	22.0	533	295	1501.0	96.1%	912	1501.0	6.7%	7068	693
CETSP-18-09	18	85.0	0.0%	252	5.7	61	198	442.1	0.0%	294	38.4	0.0%	680	274
CETSP-18-10	18	31.8	0.0%	272	4.7	68	306	176.5	0.0%	392	36.2	0.0%	506	307
Averages		**250.7**	**1.0**%	**321.1**	**8.8**	**157.2**	**278.0**	**950.9**	**21.3**%	**531.7**	**214.0**	**0.7**%	**1604.2**	**408.7**
CETSP-20-01	20	625.0	0.0%	286	11.7	221	230	1500.9	7.4%	367	556.4	0.0%	4397	656
CETSP-20-02	20	36.8	0.0%	209	11.8	175	320	1500.9	4.6%	493	135.4	0.0%	1211	609
CETSP-20-03	20	18.4	0.0%	339	8.3	98	772	1501.0	17.7%	948	41.5	0.0%	586	814
CETSP-20-04	20	36.9	0.0%	254	9.2	122	252	1501.4	13.6%	528	84.6	0.0%	992	311
CETSP-20-05	20	709.7	0.0%	309	8.2	95	309	1501.7	81.4%	842	716.9	0.0%	4628	987
CETSP-20-06	20	366.6	0.0%	464	15.0	120	306	1501.2	3.1%	567	177.3	0.0%	1734	707
CETSP-20-07	20	73.0	0.0%	238	15.8	247	251	1501.2	76.9%	1071	1501.3	3.3%	6872	741
CETSP-20-08	20	1501.4	2.5%	417	39.4	632	481	1500.8	10.0%	719	1501.0	4.3%	6100	1233
CETSP-20-09	20	91.0	0.0%	322	17.4	169	441	476.5	0.0%	515	57.0	0.0%	746	387
CETSP-20-10	20	20.4	0.0%	299	10.1	33	292	359.4	0.0%	623	21.4	0.0%	289	346
Averages		**347.9**	**0.2**%	**313.7**	**14.7**	**191.2**	**365.4**	**1284.5**	**21.5**%	**667.3**	**479.3**	**0.8**%	**2755.5**	**679.1**

CHAPTER 5
EXTENSIONS TO THE LIFETIME MAXIMIZATION PROBLEM

5.1 The Mobile Sink Model with Finite Sink Speed

In this chapter, we address an important extension to the lifetime maximization problem in wireless sensor networks (WSNs). Similar to the models presented in Chapter 2, here we examine a WSN consisting of a set of limited-power sensors that are deployed in an area for data collection purposes. The collected information needs to be transmitted for further processing to a data-collecting node called the *sink*. Sensors are often deployed randomly, but in high quantities to prevent coverage breach. This dense deployment of the sensor nodes allows multihop delivery of the collected information: Each sensor can remotely communicate with other nearby sensors via wireless links and use them to relay its collected information to the sink.

As we discussed in previous chapters, sensor energy is a scarce resource in WSNs. However, if sensors are deployed in not-easily-accessible or hostile environments, the task of replacing their batteries becomes impractical. An important optimization problem is to devise balanced communication schemes between the sensors and the sink to prolong WSN lifetime. Various lifetime maximization problems for WSNs have been studied recently [5, 6, 24–26], especially with respect to the task of exploiting sink mobility [7–10, 15–19, 26, 55, 56]. When the sink is static (i.e., located at one fixed position), the sensor nodes closer to the sink are burdened with traffic aggregation because they need to relay other nodes' traffic. Therefore, these nodes exhaust their battery energy sooner than the rest of the sensor nodes, disconnecting the rest of the network from the sink. This phenomenon is known as the *energy hole* problem [20–22]. Moving the sink in the sensor field can mitigate the energy hole problem, which results in an extended network lifetime. In this chapter, we focus on situations in which there exists one sink that moves over a subset of pre-specified sink locations in the sensor field.

In WSNs having a static sink, the main decision affecting network lifetime is how to route the data from each sensor node to the sink. On the other hand, when the sink is mobile, one must also determine how long the sink should stay at each sink location. We will refer to these decisions as *scheduling decisions*. When the sink can move arbitrarily fast between its possible locations, scheduling only refers to the amount of time that the sink stays at each location. (Later in this chapter, we will also include the *order* in which those locations are visited.) The problem is then to find a set of routing and scheduling decisions that maximize the network lifetime, where the lifetime is defined as the time until the first sensor node expends all of its battery power [5, 6].

At higher discharge rates, batteries are typically less efficient at converting their chemically stored energy into available electrical energy (see, for example, [57]). Therefore, it is energy-efficient to use routing and scheduling schemes that require frequent changes in the sensors' energy consumption rates. In this chapter, we consider a cyclic model in which the sink is required to finish some (positive integer) C cycles during the network lifetime (see [26, 58] for similar cyclic models). Our lifetime maximization problem is then focused on finding an optimal trajectory for the sink as well as optimal routing schemes for the sensor nodes during one cycle.

As mentioned in Chapter 2, Papadimitriou and Georgiadis [7] propose a linear programming formulation for the problem of maximizing the lifetime of a wireless sensor network with one mobile sink. Their formulation integrates both routing and scheduling decisions. (See also [55] for related work on this problem that employs column-generation techniques.) A major assumption in these papers is that the sink's travel time from one location to another is negligible, or equivalently, that there are many stationary sinks, one at each possible location. However, when the sink's travel times are significant compared to its dwelling times at each sink location, the order in which the sink visits these locations becomes important.

Keskin et al. [58] formulate a lifetime maximization problem that assumes nonzero sink travel times, and propose heuristic algorithms to solve it. Their formulation requires a shortest path routing of the collected data from sensor nodes to the sink, which may result in a suboptimal network lifetime as discussed in [7]. Unlike [58], the models we propose in this chapter incorporate the routing of data from sensor nodes to the sink.

We formulate the WSN lifetime maximization problem as a mixed-integer program (MIP) to capture the order in which the sink locations are visited during each cycle. We assume that the sink cannot gather information from the sensor nodes while traveling between different sink locations. As a result, the underlying application must be *delay-tolerant* with a maximum tolerable delay D, i.e., data can be delivered to the sink with a maximum delay of D time units. In some delay-tolerant WSN applications, sensor nodes are capable of postponing data transmission to the sink by storing the data locally. Alternatively, if the sensor nodes' computational power and storage capabilities are limited, the sensors may instead continue data transmission to their target sink locations, where the transmitted data is stored until a subsequent visit by the sink. While our models and solution methods in this chapter can be applied to both cases, we take the perspective of the latter case, where sink locations store data. In addition to an optimal set of routing schemes from the sensor nodes to each sink location, our formulations find an optimal trajectory for the sink that satisfies the maximum tolerable delay restriction for the underlying application.

The rest of this chapter is organized as follows. In Section 5.2 we present MIP formulations for the lifetime maximization problems examined in this chapter. In Section 5.3, we describe a cutting-plane approach for more effectively solving one variant of the problem. Next, we recast the problem as a two-stage optimization problem in Section 5.4, and prescribe a Benders decomposition algorithm for its solution. We conclude the chapter by presenting some computational results in Section 5.5.

5.2 Problem Statement

We begin in Section 5.2.1 by introducing notation and discussing several key characteristics of the lifetime maximization problem. We then formulate the lifetime maximization problem in Section 5.2.2 and present an alternative problem variation in Section 5.2.3. We provide a comparison of the presented models in Section 5.2.4.

5.2.1 Preliminaries

Let N be the set of sensor nodes ("nodes" for short) and L be the set of sink locations of a WSN. The sink periodically visits sink locations in a subset of L and collects data from the nodes. When the sink is at location $l \in L$, node i can send its data (or relay other nodes' data) only to a set of downstream neighbors S_i^l. This set is often given as $S_i^l = \{j \in N \cup \{l\} : \mu_{ij} \leq \overline{\mu}\}$, where μ_{ij} is the Euclidean distance between i and j and $\overline{\mu}$ is the maximum transmission range of any node. Denote by E_i the initial energy level of node i, and suppose that the sink is required to finish C cycles during the network lifetime. Therefore, maximizing the network lifetime is equivalent to maximizing the cycle duration, while ensuring that each sensor node's energy expenditure in a cycle does not exceed $\overline{E}_i = E_i/C$. The required energy for transmitting one unit of data from node i to a point $j \in S_i^l$ while the sink is at location $l \in L$ is denoted by $e_{ij}^l > 0$. Similarly, $\gamma > 0$ is the required energy for receiving one unit of data at every node. Finally, $d_i > 0$ represents the data generation rate of node i.

Under the commonly-used assumption of zero sink travel times [7, 55], the lifetime maximization problem in WSNs with a mobile sink can be formulated as a linear program. Let variable z_l determine how long the sink stays at location $l \in L$ during each cycle, and y_{ij}^l represent the volume of data transmitted from sensor node i to $j \in S_i^l$ while the sink is at location l. The lifetime maximization problem can be formulated as follows [7].

$$\text{Max} \sum_{l \in L} z_l \tag{5–1a}$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} e_{ij}^l y_{ij}^l + \sum_{j:i \in S_j^l} \gamma y_{ji}^l \right) \leq \overline{E}_i, \qquad \forall i \in N \tag{5–1b}$$

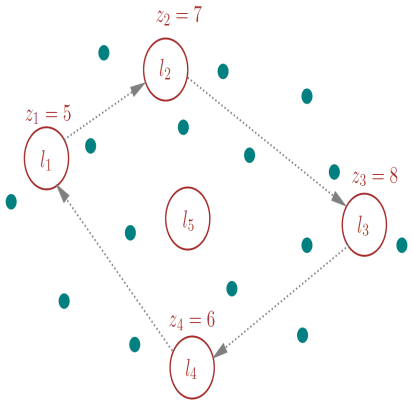
$$\sum_{j \in S_i^l} y_{ij}^l - \sum_{j:i \in S_j^l} y_{ji}^l = d_i z_l, \qquad \forall i \in N,\ l \in L \tag{5–1c}$$

$$z_l \geq 0, \qquad \forall l \in L \tag{5–1d}$$

$$y_{ij}^l \geq 0, \qquad \forall i \in N,\ j \in S_i^l,\ l \in L. \tag{5–1e}$$

The objective function optimizes the WSN lifetime by maximizing the sum of time that the sink spends staying over all possible sink locations. (The actual lifetime would equal this value multiplied by C.) Constraints (5–1b) ensure that the total energy expenditure of each sensor node i during each cycle does not exceed $\overline{E}_i$. Constraints (5–1c) are flow conservation constraints and ensure that for each sink location l, every sensor node's out-flow equals its in-flow plus the volume of data generated at the sensor node during the sink's stay at l. Constraints (5–1d) and (5–1e) state the nonnegativity requirements for z- and y-variables. We refer to this model as the *Mobile Sink Model* (MSM). Observe that this model implicitly assumes that data is routed to the sink locations in the same manner each time the sink traverses a cycle. (Here, routing includes both path selection and rate allocation.) This assumption is made for the sake of simplifying routing and data collection, and all models we present here employ this cyclic routing assumption.

When the sink moves with a finite speed, Formulation (5–1) may not necessarily yield a feasible trajectory for the sink. Consider the WSN in Figure 5-1A where the sink visits sink locations $\{l_1, l_2, l_3, l_4\}$. Figure 5-1B contains the distances between each pair of sink locations. Suppose that in some solution to (5–1), the sink stays at location l_1 for $z_1 = 5$ time units, during which time the sensor nodes route data to the sink according to a communication pattern specified by vector $\mathbf{y}^1$. Also, suppose that the solution requires a $z_2 = 7$, $z_3 = 8$, and $z_4 = 6$ time-unit stay at sink locations l_2, l_3, and l_4 with communication schemes $\mathbf{y}^2$, $\mathbf{y}^3$, and $\mathbf{y}^4$, respectively. If the sink moves arbitrarily fast,



A Illustration of the network

	l_1	l_2	l_3	l_4	l_5
l_1	0	4	12	8	5
l_2	4	0	9	9	4
l_3	12	9	0	7	6
l_4	8	9	7	0	3
l_5	5	4	6	3	0

B Pairwise distances between the sink locations

Figure 5-1. A wireless sensor network with five sink locations

as assumed in MSM, then this solution need not prescribe a specific order of visit to the sink locations.

Suppose instead that the sink travels with a speed of 2 distance units per time unit. We can build a feasible trajectory for the sink based on the foregoing optimal z- and y-values as follows:

- The sink leaves l_1 toward l_2 at time $t = 0$. During the sink's travel from l_1 to l_2, the sensor nodes transmit their collected data to l_2 according to the communication pattern specified by $\mathbf{y}^2$. The transmitted data is stored at l_2 until the sink arrives there at time $t = 2$.
- The sink stays at l_2 until $t = 7$ and collects data from the sensor nodes (according to $\mathbf{y}^2$), and then heads to l_3.
- The sink reaches l_3 at time $t = 11.5$, collects all stored data immediately, continues collecting data until $t = 15$ (according to $\mathbf{y}^3$), and then leaves l_3 to go to l_4.
- The sink then reaches l_4 at $t = 18.5$, and leaves there at $t = 21$.

- Finally, the sink reaches l_1 at $t = 25$, and stays there until the end of the cycle at $t = 26$.

Clearly, the sink can only travel on an arc if its travel time is no more than the dwelling time at the destination. Now, suppose that the sink only moves 1 distance unit per time unit. There exists no feasible trajectory for the sink that spans $\{l_1, l_2, l_3, l_4\}$, given $z_1, \dots, z_4$: The fact that $z_1 + \cdots + z_4 = 26$ implies that the sink's tour cannot take longer than 26 time units, but the shortest tour spanning these four nodes is 28 time units.

5.2.2 Problem Formulation

In this section, we consider the WSN lifetime maximization problem with a finite sink speed. Suppose that it takes $t_{lm} > 0$ time units for the sink to go from location $l \in L$ to location $m \in L$. Data generated while the sink travels from l to m is transmitted and stored at location m and is collected by the sink once it reaches m. The maximum tolerable delay is assumed to be D, so the sink can only travel on arcs that satisfy $t_{lm} \leq D$. We will refer to the directed graph $G = (L, A)$ with arc set $A = \{(l, m) : t_{lm} \leq D\}$ as the *sink graph*. In the sink graph define $N_l^+ = \{m \in L : (l, m) \in A\}$ and $N_l^- = \{m \in L : (m, l) \in A\}$. Our goal is to maximize the WSN lifetime by finding a tour for the sink over a subset of L, as well as routing patterns from each sensor node to each visited sink location. We refer to this problem as *MSM-FS1*.

To formulate the problem as a MIP, we introduce binary variables u_{lm} that equal 1 if and only if the sink travels along arc (l, m) in the underlying sink graph. Recall that the main constraint in this model requires the sink's dwelling time at l to be at least as large as t_{ml} if $u_{ml} = 1$. Our formulation satisfies this requirement because the transmission time to each sink location is now given by $z_l + \sum_{m \in N_l^-} t_{ml} u_{ml}$, where z_l is defined as the sink's stop length at l. We also introduce auxiliary binary variables v_l that equal 1 if and only if location l is visited by the sink. Problem MSM-FS1 can be formulated as follows,

where constants M_l, $\forall l \in L$, are large values that we specify later in this chapter.

$$\text{Max} \sum_{l \in L} \left(z_l + \sum_{m \in N_l^-} t_{ml} u_{ml} \right) \tag{5–2a}$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} e_{ij}^l y_{ij}^l + \sum_{j : i \in S_j^l} \gamma y_{ji}^l \right) \leq \overline{E}_i, \qquad \forall i \in N \tag{5–2b}$$

$$\sum_{j \in S_i^l} y_{ij}^l - \sum_{j : i \in S_j^l} y_{ji}^l = d_i \left(z_l + \sum_{m \in N_l^-} t_{ml} u_{ml} \right), \qquad \forall i \in N,\ l \in L \tag{5–2c}$$

$$z_l \leq M_l v_l, \qquad \forall l \in L \tag{5–2d}$$

$$\sum_{m \in N_l^+} u_{lm} = v_l, \qquad \forall l \in L \tag{5–2e}$$

$$\sum_{m \in N_l^-} u_{ml} = \sum_{m \in N_l^+} u_{lm}, \qquad \forall l \in L \tag{5–2f}$$

$$\sum_{l \in S} \sum_{m \in \overline{S}} u_{lm} \geq v_k + v_r - 1, \qquad \forall S \subset L :\ 2 \leq |S| \leq |L| - 2,\ k \in S,\ r \in \overline{S} \tag{5–2g}$$

$$v_l \in \{0, 1\}, \qquad \forall l \in L \tag{5–2h}$$

$$u_{lm} \in \{0, 1\}, \qquad \forall l \in L,\ m \in N_l^+ \tag{5–2i}$$

$$z_l \geq 0, \qquad \forall l \in L \tag{5–2j}$$

$$y_{ij}^l \geq 0, \qquad \forall i \in N,\ j \in S_i^l,\ l \in L. \tag{5–2k}$$

The objective maximizes the time that the sink dwells at sink locations, plus the time it spends traveling between these locations. Constraints (5–2b) and (5–2c) are the energy and flow conservation constraints, respectively, similar to (5–1b) and (5–1c). Constraints (5–2d) force $v_l = 1$ whenever $z_l > 0$, $\forall l \in L$. Constraints (5–2e) define the relationship between u- and v-variables and ensure that at most one arc enters each sink location, while constraints (5–2f) ensure that for each $l \in L$, the number of incoming and outgoing arcs in a feasible tour of sink locations are equal. Finally, constraints (5–2g) are the generalized subtour elimination constraints [40]. These constraints ensure that for any

subset $S \subset L$ such that $2 \leq |S| \leq |L| - 2$, there exists an arc going from S to $\overline{S}$ if both S and $\overline{S}$ contain a sink location that is on the tour.

Remark 1. Note that when $C = 1$, the sink does not need to return to its initial location and therefore, one seeks a feasible path for the sink instead of a cycle. In this case, let binary variable δ_l (τ_l) indicate whether the sink's path starts (ends) at sink location l. We would then substitute constraints (5–2e) and (5–2f) with the following constraints.

$$\delta_l + \sum_{m \in N_l^+} u_{lm} = v_l, \qquad \forall l \in L \tag{5–3a}$$

$$\sum_{m \in N_l^-} u_{ml} - \sum_{m \in N_l^+} u_{lm} = \tau_l - \delta_l, \qquad \forall l \in L \tag{5–3b}$$

$$\sum_{l \in L} \delta_l = 1 \tag{5–3c}$$

$$\delta_l \in \{0, 1\}, \qquad \forall l \in L \tag{5–3d}$$

$$\tau_l \in \{0, 1\}, \qquad \forall l \in L. \tag{5–3e}$$

Moreover, because we seek a directed path, the left-hand-side of subtour elimination constraints (5–2g) becomes $\sum_{l \in S} \sum_{m \in \overline{S}} (u_{lm} + u_{ml})$. Note that the equality $\sum_{l \in L} \tau_l = 1$ is implied by the aggregation of (5–3b) and (5–3c). □

For simplicity, we preprocess the cases in which the sink visits only a single sink location $l \in L$. To accomplish this, we solve the MSM in which L is restricted to the singleton $\{l\}$. Let M_l' be the optimal MSM objective to this problem. From this point forward, we assume that the sink visits at least two locations in L. Our "ultimate" solution is one having the largest objective among the single-sink-location solutions and the multiple-sink-location solution we obtain.

We next address the derivation of the M_l-values used in (5–2d). Noting that M_l must be at least as large as the maximum dwelling time at any location $l \in L$, we consider the scenario in which the sink spends (almost) the entire cycle duration at location l. Recall

that M'_l is the maximum amount of time the sink can spend at $l \in L$, and the sink must spend some time in transit to l, due to the assumption that the sink visits at least two locations. Hence, M'_l is an upper bound on $z_l + t_{ml}$, where m is whichever sink location visited immediately before l. Therefore,

$$M_l = M'_l - \min_{m \in N_l^-} t_{ml} \tag{5–4}$$

can be used in (5–2d). Alternatively, we can determine the minimum amount of energy that a node must expend while the sink is at l. Dividing the node's energy by this minimum expenditure yields a valid bound for M'_l, and enables us to use

$$M_l = \min_{i \in N} \left\{ \frac{\overline{E}_i}{d_i \min_{j \in S_i^l} \{e_{ij}^l\}} \right\} - \min_{m \in N_l^-} t_{ml} \tag{5–5}$$

as a valid upper bound on z_l. Although calculating M_l via (5–4) is more computationally expensive than calculating (5–5), it generally results in a stronger bound, and hence we compute M_l via (5–4) in the remainder of this chapter.

5.2.3 Alternative Delay Model

Problem MSM-FS1 assumes real-time transmission of data whenever possible, i.e., a sink location stores data only when the sink is en route to the location, and data transmission happens in real time while the sink dwells at a sink location. This model is appropriate for applications such as surveillance and real-time monitoring where unnecessary delay must be avoided, except, e.g., when the sink is traveling between the locations. In other more delay-tolerant WSN applications, it may be more appropriate to allow sensors to transmit data to some location l, even when the sink is not currently present at, or en route to, location l. However, we still honor the maximum delay restriction stating that once information begins transmission to location l, the sink must arrive at node l no more than D time units later.

To illustrate this alternative delay model, consider again the wireless sensor network in Figure 5-1A and suppose that the sink's trajectory is $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_1$. The cycle

time equals $\sum_{m=1}^{3} r_m$, where r_m is the length of the period during which the sensor nodes send their data to sink location l_m according to the data transmission scheme given by $\mathbf{y}^m$. Data is transmitted to these sink locations in the same order in which they are later visited by the sink. Unlike (5–2), it is now possible, for instance, that while the sink is moving from l_1 to l_2, the sensor nodes start transmitting data to l_3.

First, note that the the sink's tour length cannot exceed the cycle time. Therefore, there must exist at least one sink location l on the sink's tour, such that sensors are still sending data to l when the sink arrives at l. (This characterization includes the case in which sensors stop transmitting to l immediately when the sink arrives at l.) We will refer to one such sink location as the tour's *origin*. This assumption is necessary to ensure the uniformity of data routing and collection during each cycle performed by the sink.

In our example, if l_1 serves as the origin, delay restrictions impose the following constraints.

$$t_{12} \leq D, \tag{5–6a}$$

$$\max\{t_{12} - r_2, 0\} + t_{23} \leq D, \tag{5–6b}$$

$$\max\{\max\{t_{12} - r_2, 0\} + t_{23} - r_3, 0\} + t_{31} \leq D. \tag{5–6c}$$

Here inequality (5–6a) ensures that transmission delay along the path $l_l \to l_2$ does not exceed D, while (5–6b) and (5–6c) ensure that this requirement is met along $l_l \to l_2 \to l_3$ and $l_l \to l_2 \to l_3 \to l_1$, respectively. Moreover, because l_1 serves as the origin, the sensor nodes must still be sending data to l_1 when the sink arrives there, and so the following constraint must be satisfied:

$$r_1 \geq \max\{\max\{t_{12} - r_2, 0\} + t_{23} - r_3, 0\} + t_{31}.$$

To formulate these constraints when the tour's origin is not determined *a priori*, let variables u_{lm} and v_l determine the sink's trajectory (as modeled by (5–2e)–(5–2i)). Also, define binary variable s_l, for $l \in L$, which equals 1 if location l serves as the tour's

origin. We impose constraints analogous to (5–6) for every combination of origin node $l \in L$ and sink location $m \in L$. These constraints are made inactive (by weakening the right-hand-side with a big-M term) whenever $s_l = 0$ or $v_m = 0$, and are otherwise active. To model these delay constraints, let variable q_{lm}, $\forall l, m \in L$, represent the amount of time that sensors transmit data to m before the sink begins moving toward m from its preceding sink location, given that l serves as the tour's origin. Therefore, data is transmitted to m starting from $q_{lm} + \sum_{k \in N_m^-} t_{km} u_{km}$ time units before the sink arrives at m, and the following set of constraints must hold:

$$q_{lm} + \sum_{k \in N_m^-} t_{km} u_{km} \leq D, \qquad \forall l \in L,\ m \in L. \tag{5–7}$$

Constraints (5–8a) and (5–8b) below ensure the existence of a designated origin from among those nodes visited by the tour.

$$\sum_{l \in L} s_l = 1 \tag{5–8a}$$

$$s_l \leq v_l, \qquad \forall l \in L. \tag{5–8b}$$

Without loss of generality, if the sink's tour traverses sink locations $1, \ldots, p$ in order with origin node 1, then $q_{1,w+1} = \max\{q_{1w} + t_{w-1,w} - r_w, 0\}$, where all index addition and subtraction is performed modulo p, for each $w = 1, \ldots, p$. Moreover, because location 1 is the origin (and location 2 should not collect data before the sink leaves 1), $q_{12} = 0$. All other q-variables should equal 0. The following constraints properly define the q-variables as such in terms of the r-, s-, and u-variables.

$$q_{lm} \geq \sum_{k \in N_m^-} \left[q_{lk} + \sum_{h \in N_k^-} (t_{hk} - r_k) u_{hk} \right] u_{km} - D(1 - s_l), \qquad \forall l \in L,\ m \in L \tag{5–9a}$$

$$D(1 - s_l) + r_l \geq q_{ll} + \sum_{m \in N_l^-} t_{ml} u_{ml}, \qquad \forall l \in L. \tag{5–9b}$$

To see that (5–9a) and (5–9b), along with (5–7), correctly define the q-variables, we first claim that there exists a solution to these constraints in which each q-variable takes the smallest value allowed by (5–9a) and (5–9b) (due to q_{lm} being on the left-hand-side of (5–7), which is of the $\leq$ sense, and because q-variables are not present elsewhere in the model). If $s_l = 0$, then note that setting $q_{lk} = 0, \forall k \in L$, is feasible: The right-hand-side of (5–9a) would be no more than $t_{hk} - D \leq 0$ (for any $h \in N_k^-$), and (5–9b) permits $q_{ll} = 0$ due to the fact that $t_{ml} \leq D$ for all $m \in N_l^-$. Now consider the $l \in L$ such that $s_l = 1$. If $v_m = 0$, then $\sum_{k \in N_m^-} u_{km} = \sum_{k \in N_m^+} u_{mk} = 0$, and thus setting $q_{lm} = 0$ is feasible to (5–9a). Finally, consider $m \in L : v_m = 1$. Reindex the locations so that the tour visits locations $1, \ldots, p$ in order, with origin $l = 1$ (where $p \geq 2$). To begin, note that (5–9b) restricts $r_1 \geq q_{11} + t_{p1}$, and so (5–9a) for $m = 2$ reduces to $q_{12} \geq q_{11} + t_{p1} - r_1$, which is thus dominated by $q_{12} \geq 0$. Hence, $q_{12} = 0$ as desired. By induction, assume that $q_{12}, \ldots, q_{1w}$ are correctly defined for some $2 \leq w \leq p$. Constraint (5–9a) defines $q_{l,w+1} \geq q_{lw} + t_{w-1,w} - r_w$ (where $w+1 \equiv 1$ if $w = p$, and $w-1 \equiv p$ if $w = 1$), which along with the nonnegativity of the q-variables forces $q_{l,w+1} \geq \max\{q_{lw} + t_{w-1,w} - r_w, 0\}$ as desired. Repeating this argument shows that q_{lw} is correctly defined for all $w = 1, \ldots, p$.

To linearize (5–9a), we introduce nonnegative variables $\alpha_{lkm} = q_{lk} u_{km}$, $\beta_{hkm} = u_{hk} u_{km}$, and $\lambda_{hkm} = r_k u_{hk} u_{km}$, and add the following constraints:

$$D(1 - u_{km}) + \alpha_{lkm} \geq q_{lk}, \qquad \forall l \in L,\ m \in L,\ k \in N_m^- \tag{5–10}$$

$$\beta_{hkm} \geq u_{hk} + u_{km} - 1, \qquad \forall h \in L,\ k \in N_h^+,\ m \in N_k^+ \tag{5–11}$$

$$\lambda_{hkm} \leq r_k, \qquad \forall h \in L,\ k \in N_h^+,\ m \in N_k^+ \tag{5–12}$$

$$\lambda_{hkm} \leq D u_{hk}, \qquad \forall h \in L,\ k \in N_h^+,\ m \in N_k^+ \tag{5–13}$$

$$\lambda_{hkm} \leq D u_{km}, \qquad \forall h \in L,\ k \in N_h^+,\ m \in N_k^+, \tag{5–14}$$

where (5–9a) is now revised as

$$q_{lm} \geq \sum_{k \in N_m^-} \left[\alpha_{lkm} + \sum_{h \in N_k^-} (t_{hk}\beta_{hkm} - \lambda_{hkm}) \right] - D(1 - s_l).$$

Again, because α- and β-variables appear on the right-hand-side of (5–9a) (a $\geq$-constraint), there always exists an optimal solution in which these variables take on their smallest possible value permitted by (5–10) and (5–11), and so the upper-bounding constraints required to linearize α- and β-variables are unnecessary in this formulation. By a similar argument, the lower-bounding constraints for linearizing λ-variables are not required in the formulation, because there always exists an optimal solution in which these variables assume their largest possible values. Hence, problem MSM-FS2 can be formulated as follows.

$$\text{Max} \sum_{l \in L} r_l \tag{5–15a}$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} e_{ij}^l y_{ij}^l + \sum_{j:i \in S_j^l} \gamma y_{ji}^l \right) \leq \overline{E}_i, \qquad \forall i \in N \tag{5–15b}$$

$$\sum_{j \in S_i^l} y_{ij}^l - \sum_{j:i \in S_j^l} y_{ji}^l = d_i r_l, \qquad \forall i \in N,\ l \in L \tag{5–15c}$$

$$r_l \leq M_l' v_l, \qquad \forall l \in L \tag{5–15d}$$

Constraints (5–2e)–(5–2i) (5–15e)

Constraints (5–7)–(5–14) (5–15f)

$$q_{lm} \geq 0, \qquad \forall l \in L,\ m \in L \tag{5–15g}$$

$$r_l \geq 0, \qquad \forall l \in L \tag{5–15h}$$

$$\alpha_{lkm},\ \beta_{hkm},\ \lambda_{hkm} \geq 0, \qquad \forall l \in L,\ k \in N_l^+,\ m \in N_k^+. \tag{5–15i}$$

Formulation (5–15) contains an exponential number of subtour elimination constraints. In practice, one can relax the subtour elimination constraints, and add those inequalities that are violated at each node of the branch-and-bound tree by a

polynomially solvable separation routine [52]. We next formulate an alternative MIP model for MSM-FS2 that is polynomially sized. To that end, we decompose each dwelling time r_l into variables r_l^1 and r_l^2, where r_l^1 represents the amount of time that sensors send data to l before the sink arrives at l, and r_l^2 represents the amount of time that the sink stays at l. Next, define the start of a sink's cycle as the time the sink departs the origin. We also introduce variables π_l as the duration from the start of each cycle until the sensors start transmitting data to l, and ρ_l as the duration from the start of each cycle until the sink reaches l.

Given these new variable definitions, consider the following alternative formulation for MSM-FS2, where the ρ- and π-variables serve the dual roles of enforcing delay limits and preventing subtours.

$$\text{Max} \sum_{l \in L} r_l^1 + \sum_{l \in L} r_l^2 \tag{5–16a}$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} e_{ij}^l y_{ij}^l + \sum_{j : i \in S_j^l} \gamma y_{ji}^l \right) \leq \overline{E}_i, \qquad \forall i \in N \tag{5–16b}$$

$$\sum_{j \in S_i^l} y_{ij}^l - \sum_{j : i \in S_j^l} y_{ji}^l = d_i (r_l^1 + r_l^2), \qquad \forall i \in N,\ l \in L \tag{5–16c}$$

$$r_l^1 + r_l^2 \leq M_l' v_l, \qquad \forall l \in L \tag{5–16d}$$

Constraints (5–2e), (5–2f), (5–2h), (5–2i), (5–8a), and (5–8b) (5–16e)

$$\rho_m \geq \sum_{l \in N_m^-} t_{lm} u_{lm}, \qquad \forall m \in L \tag{5–16f}$$

$$\pi_m \leq M(2 - s_l - u_{lm}), \qquad \forall l \in L, m \in L \tag{5–16g}$$

$$\rho_m \geq \rho_l + r_l^2 + t_{lm} u_{lm} - M(1 - u_{lm} + s_l), \qquad \forall l \in L,\ m \in L \tag{5–16h}$$

$$r_l^1 + r_l^2 \geq \pi_m - \pi_l - M(1 - u_{lm} + s_l), \qquad \forall l \in L,\ m \in L \tag{5–16i}$$

$$r_l^1 + r_l^2 \geq \rho_l - \pi_l - D(1 - s_l), \qquad \forall l \in L \tag{5–16j}$$

$$r_l^1 + r_l^2 \geq t_{ml}(u_{ml} + s_l - 1), \qquad \forall l \in L,\ m \in L \tag{5–16k}$$

$$r_l^1 \leq \rho_l - \pi_l \leq D, \qquad \forall l \in L \tag{5–16l}$$

$$y_{ij}^{l} \geq 0, \qquad \forall i \in N,\ j \in S_i^l,\ l \in L \tag{5–16m}$$

$$r_l^1, r_l^2 \geq 0, \qquad \forall l \in L. \tag{5–16n}$$

The big-M values in constraints (5–16g), (5–16h), and (5–16i) can be set to the optimal objective function value of the corresponding MSM instance. The next two propositions formally establish the equivalence of formulations (5–15) and (5–16).

Proposition 5.1. *A feasible solution to* (5–16) *does not contain any subtours.*

Proof. Let $(\mathbf{r}^1, \mathbf{r}^2, \mathbf{y}, \mathbf{u}, \mathbf{s}, \mathbf{v}, \rho, \pi)$ denote a feasible solution to (5–16). Suppose that there exists a subtour, T, that does not contain the tour's origin. Without loss of generality, assume that T contains arcs $\{(1,2),(2,3),\ldots,(p,1)\}$, and $s_l = 0,\ \forall l = 1,\ldots,p$. Aggregating constraints (5–16h) for all arcs in T yields $t_{12} + t_{23} + \cdots + t_{p1} + \sum_{l=1}^{p} r_l^2 \leq 0$, which is a contradiction because all t-values are positive and all r^2-variables are nonnegative. This completes the proof. □

Proposition 5.2. *Formulations* (5–15) *and* (5–16) *are equivalent.*

Proof. We prove that each solution to formulation (5–15) corresponds to a solution to formulation (5–16) having the same objective function value, and vice versa. First, let $\mathbf{a} = (\mathbf{r}^1, \mathbf{r}^2, \mathbf{y}, \mathbf{u}, \mathbf{s}, \mathbf{v}, \rho, \pi)$ denote a feasible solution to (5–16). Define $r_l = r_l^1 + r_l^2$, for all $l \in L$. For $l, m \in L$, let $q_{lm} = \max\{0, \sum_{k \in N_m^-}(\rho_k - \pi_m)u_{km}\}$ if $s_l v_m (1 - u_{lm}) = 1$, and $q_{lm} = 0$ otherwise. Solution $\mathbf{b} = (\mathbf{r}, \mathbf{y}, \mathbf{q}, \mathbf{u}, \mathbf{s}, \mathbf{v})$ has the same objective function value for (5–15) as $\mathbf{a}$ does for (5–16). We next show that $\mathbf{b}$ is feasible to (5–15). Using the foregoing definitions and Proposition 5.1, $\mathbf{b}$ satisfies constraints (5–15b)–(5–15e), (5–15g), (5–15h), (5–8a), and (5–8b). Also, we compute the α-, β-, and λ-values according to their desired linearizations, and hence the solution satisfies (5–10)–(5–14) and (5–15i). Hence, we must show that $\mathbf{b}$ satisfies (5–7), (5–9a), and (5–9b).

If $q_{lm} = 0$ for $l, m \in L$, then $\mathbf{b}$ satisfies (5–7) because $t_{km} \leq D$ for all $k \in N_m^-$. Now suppose that $q_{lm} > 0$ for $l, m \in L$. Then, by definition, we must have $s_l = v_m = 1$ and

$u_{lm} = 0$. Let $k \in N_m^- \setminus l$ be the location for which $u_{km} = 1$. Note that

$$\begin{aligned} D &\geq \rho_m - \pi_m && \text{(by (5–16l))} \\ &\geq \rho_k + r_k^2 + t_{km} - \pi_m && \text{(by (5–16h), written for } (k, m)) \\ &\geq \rho_k + t_{km} - \pi_m. \end{aligned}$$

Because $D \geq t_{km}$ as well, we conclude that **b** satisfies (5–7) because

$$D \geq \max\{\rho_k - \pi_m, 0\} + t_{km} = q_{lm} + t_{km}.$$

We next prove that **b** satisfies (5–9a) for $l, m \in L$. Note that if $v_m = 0$, then (5–9a) holds because $u_{km} = 0$ for all $k \in N_m^-$. Also, if $v_m = 1$ and $s_l = 0$, then because l does not serve as the origin, we have that $q_{lk} = 0$, $\forall k \in L$, and (5–9a) holds because $r_k \geq 0$ for $k \in N_m^-$ and $t_{hk} \leq D$ for $k \in L$ and $h \in N_k^-$. Hence, we focus on the case in which $v_m = s_l = 1$. Suppose that $k \in N_m^-$, and $h \in N_k^-$ are chosen such that $u_{hk} = u_{km} = 1$. We consider three different cases for the sequence $h \to k \to m$. In case (i), assume that $k = l$. We have that $q_{lm} = 0$ and (5–9a) reduces to $0 \geq q_{ll} + t_{hl} - r_l$. If $q_{ll} = 0$, the inequality is satisfied because of (5–16k) written for (l, h), and otherwise if $q_{ll} = \rho_h - \pi_l > 0$, then the inequality is satisfied because

$$r_l = r_l^1 + r_l^2 \geq \rho_l - \pi_l \geq (\rho_h + r_h^2 + t_{hl}) - \pi_l = q_{ll} + r_h^2 + t_{hl} \geq q_{ll} + t_{hl},$$

where the first two inequalities hold because of (5–16j) and (5–16h) (written for (h, l)), respectively. In case (ii), assume that $h = l$. Hence, $q_{lk} = 0$ and (5–9a) reduces to $q_{lm} \geq t_{lk} - r_k$. Using the definition of q_{lm}, (5–16i) (for (k, m)), and (5–16f), we have that

$$q_{lm} \geq \rho_k - \pi_m \geq \rho_k - r_k - \pi_k \geq t_{lk} - \pi_k - r_k.$$

Also, (5–16g) implies that $\pi_k = 0$, and hence (5–9a) holds. Finally, in case (iii), assume that $h \neq l$ and $k \neq l$. The following inequalities hold:

$$\begin{aligned} q_{lm} &\geq \rho_k - \pi_m && \text{(definition of } q_{lm}\text{)} \\ &\geq \rho_h + r_h^2 + t_{hk} - \pi_m && \text{(5–16h) for } (h, k) \\ &\geq \rho_h + r_h^2 + t_{hk} - (\pi_k + r_k^1 + r_k^2) && \text{(5–16i) for } (k, m) \\ &\geq \rho_h - \pi_k + t_{hk} - r_k, && \end{aligned} \tag{5–17}$$

Also, the inequality $q_{lm} \geq \rho_h + r_h^2 + t_{hk} - \pi_m$ leads to the following:

$$\begin{aligned} q_{lm} &\geq \rho_k - \pi_m \\ &\geq \rho_h + r_h^2 + t_{hk} - \pi_m \\ &\geq \pi_h + r_h^1 + r_h^2 + t_{hk} - \pi_m \\ &\geq \pi_k + t_{hk} - \pi_m \\ &\geq \pi_m - r_k + t_{hk} - \pi_m \\ &= t_{hk} - r_k. \end{aligned} \tag{5–18}$$

Using (5–17) and (5–18), we obtain

$$q_{lm} \geq \max\{0, \rho_h - \pi_k\} + t_{hk} - r_k = q_{lk} + t_{hk} - r_k.$$

Therefore **b** satisfies (5–9a).

We next prove that **b** satisfies (5–9b). If $s_l = 0$, then (5–9b) holds because $q_{ll} = 0$, $t_{ml} \leq D$, for all $m \in N_l^-$, and $r_l \geq 0$. Now suppose that l and $k \in N_l^-$ are chosen such that $s_l = u_{kl} = 1$. If $q_{ll} = \rho_k - \pi_l$, then using (5–16j) and (5–16h) for (k, l) we have

$$r_l = r_l^1 + r_l^2 \geq \rho_l - \pi_l \geq \rho_k + r_k^2 + t_{kl} - \pi_l \geq \rho_k + t_{kl} - \pi_l.$$

Hence, $r_l \geq q_{ll} + t_{kl}$ and (5–9b) is satisfied. Otherwise we must have $q_{ll} = 0$, and (5–16k) implies that $r_k \geq t_{kl}$, which again implies that (5–9b) is satisfied.

Conversely, let $\mathbf{b} = (\mathbf{r}, \mathbf{y}, \mathbf{q}, \mathbf{u}, \mathbf{s}, \mathbf{v})$ denote a feasible solution to (5–15). Let l and m be two sink locations for which $s_l = u_{lm} = 1$. Define:

- $r_k^1 = q_{lk} + \sum_{h \in N_k^-} t_{hk} u_{hk}$ and $r_k^2 = r_k - r_k^1$, for $k \in L$.
- $\pi_m = 0$ and $\rho_m = t_{lm}$.
- $\pi_k = \pi_h + r_h$ if $u_{hk} = 1$ and $h \neq l$.
- $\rho_k = \rho_h + r_h^2 + t_{hk}$ if $u_{hk} = 1$ and $h \neq l$.

Then $\mathbf{a} = (\mathbf{r}^1, \mathbf{r}^2, \mathbf{y}, \mathbf{u}, \mathbf{s}, \mathbf{v}, \rho, \pi)$ has the same objective function value as does $\mathbf{b}$ in (5–15), and satisfies (5–16b)–(5–16e) and (5–16n) due to the fact that $\mathbf{a}$ is feasible to (5–15). Solution $\mathbf{b}$ satisfies all remaining constraints to (5–16) by construction (the details of this analysis are straightforward and are omitted for brevity). □

5.2.4 Comparison of MSM, MSM-FS1, and MSM-FS2

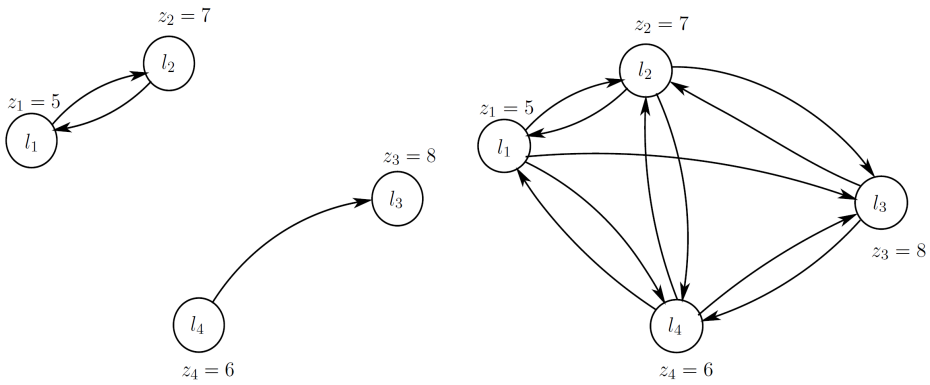
We conclude this section by stating the relationship between the optimal objective function values of the three lifetime maximization problems.

Let $(\bar{\mathbf{z}}, \bar{\mathbf{y}})$ denote an optimal solution to (5–1). From this solution, we build a directed graph $G' = (L', A')$ with $L' = \{l \in L : \bar{z}_l > 0\}$ as the node set and $A' = \{(l, m) \in A : \bar{z}_m \geq t_{lm}\}$ as the arc set. Figure 5-2 illustrates G' for the examples discussed in Section 5.2.1. Note that when the sink moves slower, G' becomes sparser (Figure 5-2A) and as we showed earlier, there exists no feasible trajectory for the sink. However, with a faster sink, G' is Hamiltonian and as we observed earlier, a feasible trajectory for the sink exists.

Proposition 5.3. *Let Z_{MSM}, $Z_{MSM\text{-}FS1}$, and $Z_{MSM\text{-}FS2}$ denote the optimal objective function value of problems MSM, MSM-FS1, and MSM-FS2, respectively. Then*

$$Z_{MSM\text{-}FS1} \leq Z_{MSM\text{-}FS2} \leq Z_{MSM}. \tag{5–19}$$

Proof. Suppose that $(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{v})$ represents an optimal solution to MSM-FS1 and let $r_l = z_l + \sum_{m \in N_l^-} t_{ml} u_{ml}$ and $q_{lm} = 0$ for all $l \in L$ and $m \in L$. Also, suppose that $s_l = 1$ for



A G' when sink speed = 1 B G' when sink speed = 2

Figure 5-2. Illustration of G' for the example in Section 5.2.1

an arbitrary l on the tour that is represented by $(\mathbf{u}, \mathbf{v})$, and $s_m = 0, \forall m \in L \backslash \{l\}$. We prove that $Z_{MSM\text{-}FS1} \leq Z_{MSM\text{-}FS2}$ by showing that $(\mathbf{r}, \mathbf{y}, \mathbf{q}, \mathbf{u}, \mathbf{v}, \mathbf{s})$ is feasible to MSM-FS2. (The two solutions clearly provide the same objective function value.) Note that if $u_{ml} = 1$, then $t_{ml} \leq r_l$ and therefore (5–9b) holds for all $l \in L$. The sink can only travel on an arc whose travel time is not more than D, and so (5–7) is satisfied. Finally, constraints (5–8a), (5–8b), and (5–9a) hold by definition. Also, because MSM-FS2 is a restriction of MSM, we have that $Z_{MSM\text{-}FS2} \leq Z_{MSM}$, which completes the proof. □

Proposition 5.4 establishes a sufficient condition under which $Z_{MSM\text{-}FS1} = Z_{MSM\text{-}FS2} = Z_{MSM}$.

Proposition 5.4. *$Z_{MSM\text{-}FS1} = Z_{MSM\text{-}FS2} = Z_{MSM}$ if there exists an optimal solution to MSM for which the corresponding graph G' is Hamiltonian.*

Proof. Because $Z_{MSM\text{-}FS1} \leq Z_{MSM\text{-}FS2} \leq Z_{MSM}$, we prove the claim by showing that an optimal solution to MSM satisfying the assumptions of the proposition corresponds to a feasible solution to MSM-FS1 having the same objective function value. Consider an optimal solution $(\bar{\mathbf{z}}, \bar{\mathbf{y}})$ to MSM satisfying the assumptions of the proposition, and define T as the set of arcs in a Hamiltonian cycle in graph G' induced by $\bar{\mathbf{z}}$. Let $u^*_{lm} = 1$ if (l, m) exists in T, and set $z^*_m = \bar{z}_m - \sum_{l \in N^-_m} t_{lm} u^*_{lm}$. By construction, $(\mathbf{u}^*, \mathbf{z}^*, \bar{\mathbf{y}})$ is feasible to

(5–2), and has the same objective function value as does the optimal MSM solution. This completes the proof. □

5.3 Cutting-plane Algorithm for MSM-FS1

In this section, we focus on the development of a cutting-plane algorithm for solving MSM-FS1. In (5–2), we revise the definition of r_l as $r_l = z_l + \sum_{m \in N_l^-} t_{ml} u_{ml}$, i.e., r_l denotes the sink's dwelling time at location l plus the time that it takes for the sink to reach l from its previous sink location, and thus represents the total amount of time data is sent to location l. Using this revised definition of r_l, we can reformulate MSM-FS1 as follows.

$$\text{Max} \sum_{l \in L} r_l \tag{5–20a}$$

$$\text{s.t.} \sum_{l \in L} \left(\sum_{j \in S_i^l} e_{ij}^l y_{ij}^l + \sum_{j : i \in S_j^l} \gamma y_{ji}^l \right) \leq \overline{E}_i, \qquad \forall i \in N \tag{5–20b}$$

$$\sum_{j \in S_i^l} y_{ij}^l - \sum_{j : i \in S_j^l} y_{ji}^l = d_i r_l, \qquad \forall i \in N,\ l \in L \tag{5–20c}$$

$$r_l \leq M_l v_l, \qquad \forall l \in L \tag{5–20d}$$

$$r_m \geq \sum_{l \in N_l^-} t_{lm} u_{lm}, \qquad \forall l \in L,\ m \in N_l^+ \tag{5–20e}$$

$$\text{Constraints (5–2e)–(5–2i), and (5–2k)} \tag{5–20f}$$

$$r_l \geq 0, \qquad \forall l \in L. \tag{5–20g}$$

Now consider the following *master problem* relaxation of (5–20).

$$\text{Max} \sum_{l \in L} r_l \tag{5–21a}$$

$$\text{s.t. Constraints (5–20b)–(5–20d), (5–20g), and (5–2k)} \tag{5–21b}$$

$$t_{lm} w_{lm} \leq r_m, \qquad \forall l \in L,\ m \in N_l^+ \tag{5–21c}$$

$$w_{lm} \leq v_l, \qquad \forall l \in L,\ m \in N_l^+ \tag{5–21d}$$

$$v_l \in \{0,1\}, \qquad \forall l \in L \tag{5–21e}$$

$$w_{lm} \in \{0,1\}, \qquad \forall l \in L,\ m \in N_l^+. \tag{5–21f}$$

Here, auxiliary binary variable w_{lm} can equal one only when $t_{lm} \le r_m\ (\le D)$ and $v_l = v_m = 1$. Because the delay constraints present in MSM-FS1 are absent in (5–21), the optimal objective function value to (5–21) matches that of MSM, and therefore provides an upper bound on the optimal objective function value of (5–20). Moreover, any feasible solution to (5–21) induces a directed graph $G' = (L', A')$ with $L' = \{l \in L : v_l = 1\}$ as its node set and $A' = \{(l,m) : w_{lm} = 1\}$ as its arc set. (More precisely, after (5–21) is solved, one should postprocess the solution and set $w_{lm} = 1$ if permitted by (5–21c), for each $l \in L$ and $m \in N_l^+$.) By Proposition 5.4, if G' is Hamiltonian, then any optimal solution to (5–21) can be used to construct an optimal solution to (5–2). Let us define the following sets with respect to an optimal solution $(\overline{\mathbf{r}}, \overline{\mathbf{y}}, \overline{\mathbf{v}}, \overline{\mathbf{w}})$ to (5–21).

$$\mathcal{V}^q = \{l \in L : \overline{v}_l = q\}, \qquad \text{for } q = 0, 1; \tag{5–22}$$

$$\mathcal{W}^1 = \{(l,m) : l \in L,\ m \in N_l^+, \text{ and } \overline{w}_{lm} = 1\} \tag{5–23}$$

$$\mathcal{W}^0 = \{(l,m) : l \in L,\ m \in N_l^+, \text{ and } \overline{w}_{lm} = 0\} \cup \{(l,m) : l \in L,\ m \in L \setminus N_l^+\}. \tag{5–24}$$

The following *subproblem* seeks a Hamiltonian cycle over $\mathcal{V}^1$, where edge (l,m) has an objective function cost coefficient of $d_{lm} = 0$ if $(l,m) \in \mathcal{W}^1$, and a cost of $d_{lm} = 1$ if $(l,m) \in \mathcal{W}^0$ (including the case when $m \notin N_l^+$). If the optimal objective function value to subproblem (5–25) below equals 0, then any of its optimal solutions represents a Hamiltonian cycle in G', which in turn indicates that the current optimal solution to the master problem corresponds to an optimal solution to MSM-FS1.

$$\text{Min} \sum_{l \in \mathcal{V}^1} \sum_{m \in \mathcal{V}^1} d_{lm} x_{lm} \tag{5–25a}$$

$$\text{s.t.} \sum_{m \in \mathcal{V}^1} x_{ml} = 1, \qquad \forall l \in \mathcal{V}^1 \tag{5–25b}$$

$$\sum_{m \in \mathcal{V}^1} x_{lm} = 1, \qquad \forall l \in \mathcal{V}^1 \tag{5–25c}$$

$$\sum_{l \in S} \sum_{m \in S} x_{lm} \leq |S| - 1, \qquad \forall S \subset \mathcal{V}^1,\ 2 \leq |S| \leq |\mathcal{V}^1| - 2 \tag{5–25d}$$

$$x_{lm} \in \{0, 1\}, \qquad \forall l \in \mathcal{V}^1,\ m \in \mathcal{V}^1. \tag{5–25e}$$

Note that (5–25) is an instance of the *Traveling Salesman Problem (TSP)* [14]. Let n^* denote the optimal objective function value of (5–25). If n^* is positive, then any Hamiltonian cycle in G' (if one exists) uses at least n^* arcs from $\mathcal{W}^0$, and we must add an appropriate cutting plane to the master problem (5–21) to ensure the existence of a Hamiltonian cycle in G' either by modifying r-values (which possibly modifies $\mathcal{W}^0$ and $\mathcal{W}^1$) or the composition of $\mathcal{V}^1$. We next describe our method of generating cutting planes for the master problem.

Let $(\mathbf{r}, \mathbf{y}, \mathbf{v}, \mathbf{u})$ be a feasible solution to (5–20); this solution is also feasible to (5–21) by letting $w_{lm} = u_{lm}$. The following proposition presents a class of cutting planes for the master problem (5–21) that are valid for all feasible solutions of (5–20).

Proposition 5.5. *Suppose that the optimal objective function value of the subproblem* (5–25) *given* $\overline{\mathbf{v}}$ *and* $\overline{\mathbf{w}}$ *is* $n^* > 0$. *The following inequality for* (5–21) *cuts off the solution containing* $\overline{v}$ *and* $\overline{w}$*, and is valid for all feasible solutions to* (5–20)*:*

$$2 \sum_{l \in \mathcal{V}^1} (1 - v_l) + \sum_{(l,m) \in \mathcal{W}^0} w_{lm} \geq n^* + \sum_{l \in \mathcal{V}^0} v_l. \tag{5–26}$$

Proof. First note that (5–26) cuts off $(\overline{\mathbf{v}}, \overline{\mathbf{w}})$ when $n^* > 0$, because the left-hand-side of (5–26) evaluates to 0 for this solution while the right-hand-side equals n^*. We say that location l is *active* if $v_l = 1$, and is *inactive* otherwise. Suppose that a feasible solution $(\widehat{\mathbf{r}}, \widehat{\mathbf{y}}, \widehat{\mathbf{v}}, \widehat{\mathbf{u}})$ to (5–20) induces a Hamiltonian cycle H in which there exists $p \geq 0$ inactive sink locations in $\mathcal{V}^1$, and $r \geq 0$ active sink locations in $\mathcal{V}^0$. Note that if $p = |\mathcal{V}^1|$, then

(5–26) reduces to

$$\sum_{(l,m)\in\mathcal{W}^0} w_{lm} \geq n^* - 2|\mathcal{V}^1| + r,$$

which is valid because $n^* \leq |\mathcal{V}^1|$ and $\sum_{(l,m)\in\mathcal{W}^0} w_{lm} = r$. (The equality holds because in this case H contains exactly r arcs with both endpoints in $\mathcal{V}^0$.) Therefore, we will assume in the remainder of this proof that $p \leq |\mathcal{V}^1| - 1$, i.e., H visits at least one sink location from $\mathcal{V}^1$.

Suppose by contradiction that the solution associated with H violates (5–26). Then we must have

$$2p + \sum_{(l,m)\in\mathcal{W}^0} \widehat{u}_{lm} \leq n^* - 1 + r. \tag{5–27}$$

Hence, H contains at most $n^* - 1 + r - 2p$ arcs from $\mathcal{W}^0$. Index these nodes as $i^1, j_1^1, \ldots, j_{q_1}^1, \ldots, i^t, j_1^t, \ldots, j_{q_t}^t, i^1$, where $i^1, i^2, \ldots, i^t$ belong to $\mathcal{V}^1$ and all other nodes belong to $\mathcal{V}^0$. (If $q_e = 0$ for some $e = 1, \ldots, t$, then the tour proceeds from i^e to i^{e+1}, where $i^{t+1} \equiv i^1$.) We build a new cycle H' by first removing all nodes in $\mathcal{V}^0$ from H, so that H' initially consists of $i^1, i^2, \ldots, i^t, i^1$ (as shown in Figure 5-3). Note that in removing nodes $j_1^e, \ldots, j_{q_e}^e$ from H, for some $e \in \{1, \ldots, t\}$ with $q_e \geq 1$, a total of $q_e + 1$ arcs in $\mathcal{W}^0$ are removed from H' while adding at most one arc (i^e, i^{e+1}) in $\mathcal{W}^0$ to H'. Hence, H' now contains at most $n^* - 1 - 2p$ arcs in $\mathcal{W}^0$. Next, suppose that we expand H' to include all p nodes in $\mathcal{V}^1$ that were not contained in H. These p nodes can be inserted to H' in any arbitrary order, with each insertion requiring at most two inactive arcs. Now, H' consists of at most $n^* - 1$ arcs in $\mathcal{W}^0$, which contradicts the assumption that the optimal objective function value of (5–25) is n^*. This completes the proof. □

Remark 2. The last step in the proof of Proposition 5.5 can alternatively be stated in the following way: If $p > 0$, include all sink locations $i^{t+1}, \ldots, i^{t+p}$ in $\mathcal{V}^1$ not present in H' (and H) by simply extending H' to visit all locations in the order $i^1, i^2, \ldots, i^{t+p}, i^1$, which

in the worst case adds $p+1$ more arcs in $\mathcal{W}^0$ to H'. Then, H' spans $\mathcal{V}^1$ and contains at most $n^* - p$ arcs in $\mathcal{W}^0$ when $p > 0$, and at most $n^* - 1$ arcs in $\mathcal{W}^0$ when $p = 0$. This suggests the following alternative valid inequality for master problem (5–21):

$$1 + \sum_{l \in \mathcal{V}^1} (1 - v_l) + \sum_{(l,m) \in \mathcal{W}^0} w_{lm} \geq n^* + \sum_{l \in \mathcal{V}^0} v_l. \tag{5–28}$$

Note that while (5–28) is valid and cuts off solutions that are not cut off by (5–26), it does not cut off an infeasible master problem solution when the subproblem yields an optimal objective function value of $n^* = 1$. However, we can add both (5–26) and (5–28) to the master problem in our cutting-plane scheme, or add (5–26) when $n^* = 1$ and (5–28) otherwise. A brief computational study reveals that the use of (5–26) by itself in the cutting-plane algorithm is the most effective implementation. □

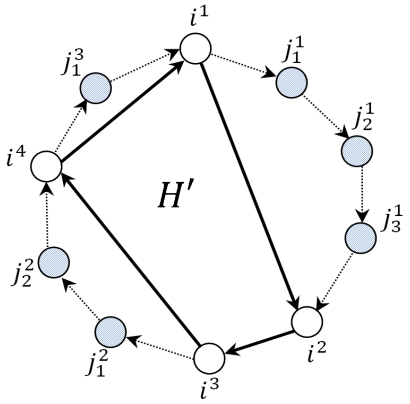


Figure 5-3. Construction of H' in the proof of Proposition 5.5

Now, we present a method that permits us to set certain w-values equal to zero in a preprocessing phase. Let Z_{LB} denote a lower bound on the optimal objective function value of (5–2), e.g., as computed via the objective function value of a feasible solution to (5–2). To determine if a given arc $(l, m) \in A$ could be traversed by the sink in an optimal solution to (5–2), we check a necessary condition for it to be used in at least one feasible solution having an objective function value that is no less than Z_{LB}. Specifically,

if

$$\left(\sum_{i\in N} d_i\right)\left(\min_{i:m\in S_i^m}\{e_{im}^m\}\right)t_{lm} + \left(\sum_{i\in N\setminus\{i:m\in S_i^m\}} d_i\right)\gamma t_{lm} + (Z_{LB} - t_{lm})\sum_{i:m\in S_i^m}\left(d_i \min_{l\in L, j\in S_i^l}\{e_{ij}^l\}\right) > \sum_{i:m\in S_i^m}\overline{E}_i, \quad (5\text{–}29)$$

then (l, m) cannot be included in any optimal solution to (5–2) and hence $w_{lm} = 0$ is valid for (5–21). The first two terms in (5–29) provide a lower bound on the total energy expended by sensor nodes that can communicate with the sink at location m during the sink's travel on arc (l, m), while the third term is a lower bound on the combined energy expended by these sensors during the rest of the network lifetime.

Remark 3. We can also use the information from an optimal solution to MSM to add valid inequalities to MSM-FS1. Let $L' \subset L$ and suppose that $Z_{TSP(L')} > Z_{MSM}$, where $Z_{TSP(L')}$ is the minimum TSP tour length over the sink locations in L'. Then, any TSP tour over a subset of L that visits all sink locations in L' is not associated with a feasible to (5–2) because any such solution would have an objective function value greater than Z_{MSM}. Therefore,

$$\sum_{l\in L'} v_l \leq |L'| - 1 \quad (5\text{–}30)$$

is valid for (5–2). In particular, if $t_{lm} > \frac{Z_{MSM}}{2}$, then $u_{lm} = 0$ is valid for (5–2). □

5.4 Benders Decomposition

The cutting-plane algorithm presented in Section 5.3 for the MSM-FS1 starts from a relaxation of the original lifetime maximization problem whose optimal solution may not admit a Hamiltonian cycle of the active sink locations. The algorithm then induces the existence of a Hamiltonian cycle by iteratively adding inequalities of form (5–26) to the master problem. In this section, we propose a decomposition approach that starts from a tour over a subset of the sink locations and reaches an optimal solution by adding appropriate Benders cuts [59]. Our decomposition approach makes the sink's routing

decisions in the following master problem.

$$\text{Max} \sum_{l\in L}\sum_{m\in N_l^-} t_{ml}u_{ml} + \theta \tag{5–31a}$$

$$\text{s.t.} \sum_{m\in N_m^-} u_{ml} = v_l, \qquad \forall l \in L \tag{5–31b}$$

$$\sum_{m\in N_m^-} u_{ml} = \sum_{m\in N_l^+} u_{lm}, \qquad \forall l \in L \tag{5–31c}$$

$$\sum_{l\in S}\sum_{m\in \overline{S}} u_{lm} \geq v_k + v_r - 1, \qquad \forall S \subset L, k \in S, r \in \overline{S} \tag{5–31d}$$

$$v_l \in \{0,1\}, \qquad \forall l \in L \tag{5–31e}$$

$$u_{lm} \in \{0,1\}, \qquad \forall l \in L,\ m \in N_l^+, \tag{5–31f}$$

where θ is defined as follows, given values $\mathbf{u} = \bar{\mathbf{u}}$ and $\mathbf{v} = \bar{\mathbf{v}}$

$$\theta = \text{Max} \sum_{l\in L} z_l \tag{5–32a}$$

$$\text{s.t.} \sum_{l\in L}\left(\sum_{j\in S_i^l} e_{ij}^l y_{ij}^l + \sum_{j:i\in S_j^l} \gamma y_{ji}^l\right) \leq \overline{E}_i, \qquad \forall i \in N \tag{5–32b}$$

$$\sum_{j\in S_i^l} y_{ij}^l - \sum_{j:i\in S_j^l} y_{ji}^l = d_i\left(z_l + \sum_{m\in N_l^-} t_{ml}\bar{u}_{ml}\right), \qquad \forall i \in N,\ l \in L \tag{5–32c}$$

$$z_l \leq M_l \bar{v}_l, \qquad \forall l \in L \tag{5–32d}$$

$$z_l \geq 0, \qquad \forall l \in L \tag{5–32e}$$

$$y_{ij}^l \geq 0, \qquad \forall i \in N,\ j \in S_i^l,\ l \in L. \tag{5–32f}$$

Problems (5–31) and (5–32) together provide an equivalent reformulation for (5–2). Let f_i, g_{il}, and h_l denote the dual variables associated with constraints (5–32b), (5–32c), and (5–32d), respectively. Taking the dual of (5–32), we get

$$\theta = \text{Min} \sum_{i\in N} \overline{E}_i f_i + \sum_{i\in N}\sum_{l\in L} d_i\left(\sum_{m\in N_l^-} t_{ml}\bar{u}_{ml}\right) g_{il} + \sum_{l\in L} M_l \bar{v}_l h_l \tag{5–33a}$$

$$\text{s.t.} \sum_{i \in N} -d_i g_{il} + h_l \geq 1, \qquad \forall l \in L \tag{5–33b}$$

$$g_{il} - g_{jl} + e_{ij}^l f_i + \gamma f_j \geq 0, \qquad \forall i \in N,\ j \in S_i^l \setminus \{s\},\ l \in L \tag{5–33c}$$

$$g_{il} + e_{is}^l f_i \geq 0, \qquad \forall i \in N,\ s \in S_i^l,\ l \in L \tag{5–33d}$$

$$h_l \geq 0, \qquad \forall l \in L \tag{5–33e}$$

$$f_i \geq 0, \qquad \forall i \in N. \tag{5–33f}$$

If problem (5–33) (which serves as the Benders dual subproblem) has an unbounded direction $(\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \widehat{\mathbf{h}})$, we add the following feasibility cut to the Benders master problem (5–31).

$$\sum_{i \in N} \overline{E}_i \widehat{f}_i + \sum_{i \in N} \sum_{l \in L} \sum_{m \in N_l^-} (d_i t_{ml} \widehat{g}_{il}) u_{ml} + \sum_{l \in L} (M_l \widehat{h}_l) v_l \geq 0. \tag{5–34}$$

When (5–33) has an optimal solution $(\mathbf{f}^*, \mathbf{g}^*, \mathbf{h}^*)$, we add the following optimality cut to the master problem. Note that (5–33) is never infeasible, because setting $h_l = 1$, for all $l \in L$, and all other variables equal to zero, is always feasible.

$$\theta \leq \sum_{i \in N} \overline{E}_i f_i^* + \sum_{i \in N} \sum_{l \in L} \sum_{m \in N_l^-} (d_i t_{ml} g_{il}^*) u_{ml} + \sum_{l \in L} (M_l h_l^*) v_l. \tag{5–35}$$

5.5 Computational Results

We examine the computational effectiveness of the proposed algorithms in this chapter on a collection of randomly generated instances. We generated these instances by randomly placing sensor nodes and sink locations in a circular area of radius 25. A sensor node can communicate with sensor nodes and sink locations in a disk of radius 15 around it. We assume an initial energy and fixed data generation rate of $\overline{E}_i = 500$ and $d_i = 50$ for all sensor nodes. To calculate the required energy for transmission of a unit of data along the link (i, j), we use an energy model similar to that of [8]. More specifically, we use $e_{ij}^l = 0.0005 + 0.00013\mu_{ij}^4$ and $\gamma = 0.0005$, where μ_{ij} denotes the

distance between two nodes i and j that can communicate with one another while the sink is at a sink location $l \in L$.

We have implemented all mixed-integer programs in C++ calling CPLEX version 11.0 via ILOG Concert Technology 2.5. All codes are compiled using Microsoft Visual C++ 2008 and the computational experiments are carried out on a PC having an Intel Core2 Quad processor Q9500 and 4 GB of memory, running Windows 7. A 1000-second time limit is imposed on all running times.

We first provide a comparison between solving MSM-FS1 by the cutting-plane algorithm of Section 5.3 and by solving problem (5–2) directly using CPLEX. For this experiment, we generate random instances of two different sizes $(|N|, |L|) = (50, 15)$ and $(|N|, |L|) = (60, 20)$ as well as two different levels $D = 0.2$ and $D = 0.4$ for the maximum tolerable delay. For each combination of problem size and delay tolerance level, ten random networks are generated. We then generate two instances of MSM-FS1 associated with the two different sink speeds, and solve these instances both directly via CPLEX (which we call "CPLEX" in our computational tables), and also by the cutting-plane algorithm in Section 5.3. Our CPLEX implementation for MSM-FS1 uses a special callback function inside the branch-and-bound tree to add the violated subtour elimination constraints at each node. This callback function is an implementation of the following separation procedure [52], where $\overline{u}_{lm}$ and $\overline{v}_l$ denote current values of variables u_{lm} and v_l, respectively.

- Build a complete directed graph G having L as its node set, and fix the capacity of each arc (l, m) in G to $\overline{u}_{lm}$.
- For every pair of sink locations l and m such that $\overline{v}_l + \overline{v}_m > 1$, solve a maximum flow problem from l and m on G.
- If the capacity of the resulting minimum cut $(S, \overline{S})$ is less than $\overline{v}_l + \overline{v}_m - 1$, that is, if $\sum_{l \in S} \sum_{m \in \overline{S}} \overline{u}_{lm} < \overline{v}_k + \overline{v}_r - 1$, then add the violated inequality to the linear programming relaxation at the current node.

Valid inequalities (5–26) are also added to the master problem (5–21) using a callback function. We add these inequalities only at branch-and-bound nodes having integer solutions. We do not include inequalities based on (5–29) in our preprocessing step because efficient deployment of these inequalities requires a heuristic procedure for obtaining good lower-bound solutions, which is beyond the scope of this chapter. Similarly, separation of inequalities (5–30) for a given subset of L requires solving an instance of the TSP. Therefore, we do not add these inequalities to our formulation. Table 5-1 contains the results of running the two algorithms on the generated instances for $D = 0.2$. Table 5-2 contains the results for $D = 0.4$. These results suggest that the cutting-plane algorithm is advantageous only for instances in which a few cuts are needed to solve the problem to optimality.

Table 5-3 presents the results of running the Benders decomposition algorithm of Section 5.4 for MSM-FS1 versus a direct solve by CPLEX for two different sink speeds and two problem sizes $(|N|, |L|) = (50, 15)$ and $(|N|, |L|) = (60, 20)$. Here, we assume a maximum tolerable delay of $D = 0.2$ time units. All cutting planes, including the subtour elimination and Benders feasibility and optimality cuts, are added using a callback routine. The Benders feasibility and optimality cuts are added only at branch-and-bound nodes having integer solutions. The results clearly favor the use of Benders decomposition for both sink speeds, as the average time for Benders decomposition is significantly shorter than that of CPLEX. Table 5-4 provides a comparison of the performance of the two algorithms for the same instances and the same sink speeds but with an increased maximum tolerable delay of $D = 0.4$ time units for all instances. In this case, the results for $v = 25$ still suggest using the Benders decomposition algorithm. However, results for instances with $v = 50$ suggest using a direct solve by CPLEX rather than the Benders decomposition algorithm. The combination of lower travel times due to increased sink speed and the increase in maximum tolerable delay makes the underlying sink graphs for $v = 50$ denser than

their counterparts for lower sink speeds and shorter maximum tolerable delay levels. We conclude that a direct solve of (5–2) by CPLEX tends to outperform Benders decomposition on instances having relatively dense sink graphs. One explanation for the effectiveness of Benders decomposition on sparse sink graphs is that sparse graphs contain fewer cycles, which in turn requires fewer Benders decomposition iterations.

We conclude this section by providing a comparison between the running times of formulations (5–15) and (5–16) in solving MSM-FS2 in Table 5-5. These results are obtained by solving MSM-FS2 on the same networks used in Table 5-4. The maximum tolerable delay is set to $D = 0.4$. The subtour elimination constraints in (5–15) are initially relaxed and added inside a callback function using the foregoing separation procedure. In addition to the running times for both formulations (in CPU seconds), we report the number of subtour elimination constraints added to (5–15) for each instance. These results suggest that solving (5–15) is generally faster, despite the fact that it is not polynomially sized. Also, the results suggest that MSM-FS2 tends to be more challenging and computationally complex than MSM-FS1.

Table 5-1. Comparison of the direct solve and cutting-plane algorithms for MSM-FS1

	$v = 25$				$v = 50$			
	CPLEX		Cutting-Plane		CPLEX		Cutting-Plane	
	Time	Subtours	Time	Cuts	Time	Subtours	Time	Cuts
MSM-FS1-50-15-01	16.6	8	14.5	1	6.4	40	59.8	59
MSM-FS1-50-15-02	7.0	8	14.3	1	12.5	76	22.9	22
MSM-FS1-50-15-03	0.6	0	0.8	0	1.2	0	2.8	0
MSM-FS1-50-15-04	0.3	0	0.3	0	27.2	72	127.0	38
MSM-FS1-50-15-05	5.6	16	3.4	2	33.0	72	78.0	73
MSM-FS1-50-15-06	0.9	0	0.9	0	7.3	16	8.0	1
MSM-FS1-50-15-07	4.7	8	5.2	1	11.9	84	36.7	58
MSM-FS1-50-15-08	0.6	0	0.9	0	1.9	8	2.4	0
MSM-FS1-50-15-09	0.5	0	0.5	0	7.3	32	19.5	21
MSM-FS1-50-15-10	3.8	8	3.1	1	35.7	156	323.0	211
Average	**4.1**	**4.8**	**4.4**	**0.6**	**14.4**	**55.6**	**68.0**	**48.3**
MSM-FS1-60-20-01	18.5	44	66.5	17	42.5	216	> 1000.0	> 1274
MSM-FS1-60-20-02	2.9	0	3.4	0	3.0	0	3.2	0
MSM-FS1-60-20-03	19.6	32	24.4	5	37.7	160	448.3	918
MSM-FS1-60-20-04	31.2	36	33.1	8	64.2	172	919.7	916
MSM-FS1-60-20-05	2.4	0	8.2	0	98.0	104	> 1000.0	> 352
MSM-FS1-60-20-06	25.8	16	23.4	2	55.4	100	284.1	256
MSM-FS1-60-20-07	7.2	0	20.7	2	53.7	306	> 1000.0	> 1320
MSM-FS1-60-20-08	47.4	48	77.0	59	25.1	52	> 1000.0	> 1070
MSM-FS1-60-20-09	223.5	52	355.8	31	415.0	168	> 1000.0	> 95
MSM-FS1-60-20-10	25.3	56	21.4	14	70.3	406	> 1000.0	> 654
Average	**40.4**	**28.4**	**63.4**	**13.8**	**86.5**	**168.4**	$> \mathbf{765.6}$	$> \mathbf{685.5}$

Table 5-2. Performance of the cutting-plane algorithm for MSM-FS1 under increased delay tolerance

	$v = 25$				$v = 50$			
	CPLEX		Cutting-Plane		CPLEX		Cutting-Plane	
	Time	Subtours	Time	Cuts	Time	Subtours	Time	Cuts
MSM-FS1-50-15-01	4.6	52	25.0	51	7.5	12	13.5	42
MSM-FS1-50-15-02	12.3	68	17.6	19	5.4	78	6.7	8
MSM-FS1-50-15-03	1.5	0	2.6	0	1.8	0	2.0	0
MSM-FS1-50-15-04	23.7	80	89.2	27	11.1	252	$>$ 1000.0	$>$ 1030
MSM-FS1-50-15-05	25.4	56	137.2	57	19.1	210	$>$ 1000.0	$>$ 1352
MSM-FS1-50-15-06	5.5	16	7.4	5	1.6	8	2.2	0
MSM-FS1-50-15-07	16.7	56	30.5	42	23.5	380	$>$ 1000.0	$>$ 2006
MSM-FS1-50-15-08	1.6	0	1.3	0	32.4	108	200.5	48
MSM-FS1-50-15-09	5.0	32	15.8	9	15.7	176	$>$ 1000.0	$>$ 783
MSM-FS1-50-15-10	22.8	94	343.9	134	20.8	280	$>$ 1000.0	$>$ 218
Average	**11.9**	**45.4**	**67.1**	**34.4**	**13.9**	**150.4**	$>$ **522.5**	$>$ **548.7**
MSM-FS1-60-20-01	44.1	250	$>$ 1000.0	$>$ 1030	25.2	208	$>$ 1000.0	$>$ 1021
MSM-FS1-60-20-02	3.6	8	3.3	0	4.6	14	10.4	0
MSM-FS1-60-20-03	20.3	92	$>$ 1000.0	$>$ 366	133.1	468	$>$ 1000.0	$>$ 366
MSM-FS1-60-20-04	52.4	166	$>$ 1000.0	$>$ 280	30.1	278	$>$ 1000.0	$>$ 1046
MSM-FS1-60-20-05	154.0	170	$>$ 1000.0	$>$ 128	22.4	150	46.7	6
MSM-FS1-60-20-06	47.4	166	292.7	108	66.7	688	$>$ 1000.0	$>$ 544
MSM-FS1-60-20-07	39.4	276	$>$ 1000.0	$>$ 682	37.7	310	138.8	101
MSM-FS1-60-20-08	18.4	48	$>$ 1000.0	$>$ 45	18.3	112	382.8	90
MSM-FS1-60-20-09	326.9	168	998.4	78	355.3	658	$>$ 1000.0	$>$ 362
MSM-FS1-60-20-10	73.5	518	$>$ 1000.0	$>$ 462	54.7	820	$>$ 1000.0	$>$ 216
Average	**78.0**	**186.2**	$>$ **829.6**	$>$ **317.9**	**74.8**	**370.6**	$>$ **657.9**	$>$ **375.2**

Table 5-3. Comparison of the direct solve and Benders decomposition algorithms for MSM-FS1

	$v = 25$					$v = 50$				
	CPLEX		Benders			CPLEX		Benders		
	Time	Subtours	Time	Benders Cuts	Subtour Cuts	Time	Subtours	Time	Benders Cuts	Subtour Cuts
MSM-FS1-50-15-1	16.6	8	2.0	2	8	6.4	40	2.8	5	96
MSM-FS1-50-15-2	7.0	8	2.3	2	8	12.5	76	2.9	5	80
MSM-FS1-50-15-3	0.6	0	3.3	2	8	1.2	0	4.7	9	134
MSM-FS1-50-15-4	0.3	0	1.7	1	0	27.2	72	3.2	8	142
MSM-FS1-50-15-5	5.6	16	2.0	3	24	33.0	72	2.5	5	126
MSM-FS1-50-15-6	0.9	0	3.5	1	0	7.3	16	4.3	5	48
MSM-FS1-50-15-7	4.7	8	1.7	2	8	11.9	84	2.7	7	90
MSM-FS1-50-15-8	0.6	0	1.3	1	0	1.9	8	2.3	7	92
MSM-FS1-50-15-9	0.5	0	1.7	1	0	7.3	32	2.4	6	60
MSM-FS1-50-15-10	3.8	8	1.9	2	8	35.7	156	3.0	6	172
Average	**4.1**	**4.8**	**2.1**	**1.7**	**6.4**	**14.4**	**55.6**	**3.1**	**6.3**	**104.0**
MSM-FS1-60-20-1	18.5	44	7.5	6	96	42.5	216	11.0	12	406
MSM-FS1-60-20-2	2.9	0	7.9	6	74	3.0	0	7.4	4	432
MSM-FS1-60-20-3	19.6	32	7.7	4	48	37.7	160	6.7	5	392
MSM-FS1-60-20-4	31.2	36	6.0	4	56	64.2	172	8.5	10	378
MSM-FS1-60-20-5	2.4	0	5.5	2	0	98.0	104	7.7	8	516
MSM-FS1-60-20-6	25.8	16	4.8	3	24	55.4	100	6.6	8	312
MSM-FS1-60-20-7	7.2	0	5.4	3	16	53.7	306	8.2	11	424
MSM-FS1-60-20-8	47.4	48	6.8	6	80	25.1	52	7.1	6	346
MSM-FS1-60-20-9	223.5	52	7.6	7	140	415.0	168	7.5	5	248
MSM-FS1-60-20-10	25.3	56	5.8	3	80	70.3	406	7.6	6	514
Average	**40.4**	**28.4**	**6.5**	**4.4**	**61.4**	**86.5**	**168.4**	**7.8**	**7.5**	**396.8**

Table 5-4. Performance of the Benders decomposition algorithm under increased delay tolerance

	$v = 25$					$v = 50$				
	CPLEX		Benders			CPLEX		Benders		
	Time	Subtours	Time	Benders Cuts	Subtour Cuts	Time	Subtours	Time	Benders Cuts	Subtour Cuts
MSM-FS1-50-15-01	4.6	52	3.073	6	96	7.5	12	15.054	39	1068
MSM-FS1-50-15-02	12.3	68	3.057	5	80	5.4	78	16.068	41	860
MSM-FS1-50-15-03	1.5	0	4.352	9	148	1.8	0	5.726	18	396
MSM-FS1-50-15-04	23.7	80	3.213	8	142	11.1	252	133.097	240	1834
MSM-FS1-50-15-05	25.4	56	2.808	7	126	19.1	210	32.199	51	1562
MSM-FS1-50-15-06	5.5	16	4.322	5	48	1.6	8	8.033	17	758
MSM-FS1-50-15-07	16.7	56	2.714	7	90	23.5	380	6.458	18	754
MSM-FS1-50-15-08	1.6	0	2.121	6	92	32.4	108	5.381	21	756
MSM-FS1-50-15-09	5.0	32	2.668	6	60	15.7	176	20.733	54	1104
MSM-FS1-50-15-10	22.8	94	3.556	10	180	20.8	280	104.285	145	2366
Average	**11.9**	**45.4**	**3.2**	**6.9**	**106.2**	**13.9**	**150.4**	**34.7**	**64.4**	**1145.8**
MSM-FS1-60-20-01	44.1	250	9.109	9	412	25.2	208	NA	NA	NA
MSM-FS1-60-20-02	3.6	8	9.127	9	496	4.6	14	133.339	164	3442
MSM-FS1-60-20-03	20.3	92	7.12	6	400	133.1	468	34.58	56	1980
MSM-FS1-60-20-04	52.4	166	8.63	13	438	30.1	278	NA	NA	NA
MSM-FS1-60-20-05	154.0	170	9.062	14	518	22.4	150	NA	NA	NA
MSM-FS1-60-20-06	47.4	166	6.32	6	312	66.7	688	147.747	134	2798
MSM-FS1-60-20-07	39.4	276	7.652	9	424	37.7	310	NA	NA	NA
MSM-FS1-60-20-08	18.4	48	6.806	7	346	18.3	112	NA	NA	NA
MSM-FS1-60-20-09	326.9	168	7.55	5	236	355.3	658	NA	NA	NA
MSM-FS1-60-20-10	73.5	518	14.947	16	566	54.7	820	50.529	62	1990
Average	**78.0**	**186.2**	**8.6**	**9.4**	**414.8**	**74.8**	**370.6**	**NA**	**NA**	**NA**

NA: Instance not solved due to memory limitations.

Table 5-5. Comparison of solution times for MSM-FS2 models (5–15) and (5–16)

	$v = 25$			$v = 50$		
	Formulation (5–15)		Formulation (5–16)	Formulation (5–15)		Formulation (5–16)
	Time	Subtours	Time	Time	Subtours	Time
MSM-FS2-50-15-01	5.3	50	8.8	7.1	68	143.1
MSM-FS2-50-15-02	8.9	80	16.5	30.0	534	13.6
MSM-FS2-50-15-03	3.3	0	1.3	3.9	84	6.4
MSM-FS2-50-15-04	19.8	108	21.0	$>$ 1000.0	$>$ 3010	$>$ 1000.0
MSM-FS2-50-15-05	21.5	78	46.7	$>$ 1000.0	$>$ 2870	$>$ 1000.0
MSM-FS2-50-15-06	1.7	0	2.0	1.8	0	6.4
MSM-FS2-50-15-07	13.7	84	24.5	35.7	442	$>$ 1000.0
MSM-FS2-50-15-08	1.6	18	1.6	21.1	328	397.6
MSM-FS2-50-15-09	6.8	32	16.7	$>$ 1000.0	$>$ 1932	$>$ 1000.0
MSM-FS2-50-15-10	32.9	120	138.8	$>$ 1000.0	$>$ 2542	$>$ 1000.0
Average	**11.5**	**57.0**	**27.8**	$>$ **410.0**	$>$ **1181.0**	$>$ **556.8**
MSM-FS2-60-20-01	61.6	258	815.8	$>$ 1000.0	$>$ 3574	$>$ 1000.0
MSM-FS2-60-20-02	4.9	0	3.3	2.9	0	4.1
MSM-FS2-60-20-03	35.7	104	493.3	$>$ 1000.0	$>$ 616	$>$ 1000.0
MSM-FS2-60-20-04	60.8	202	522.8	NA	NA	$>$ 1000.0
MSM-FS2-60-20-05	96.1	174	$>$ 1000.0	$>$ 1000.0	$>$ 2256	319.3
MSM-FS2-60-20-06	26.7	86	117.7	$>$ 1000.0	$>$ 2004	999.4
MSM-FS2-60-20-07	54.0	398	179.7	$>$ 1000.0	$>$ 4222	$>$ 1000.0
MSM-FS2-60-20-08	18.7	94	126.6	$>$ 1000.0	$>$ 2356	$>$ 1000.0
MSM-FS2-60-20-09	408.2	194	634.5	NA	NA	$>$ 1000.0
MSM-FS2-60-20-10	45.0	378	$>$ 1000.0	$>$ 1000.0	$>$ 4308	$>$ 1000.0
Average	**81.2**	**188.8**	$>$ **489.4**	**NA**	**NA**	$>$ **832.4**

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this dissertation, we considered several mathematical programming models for maximizing the lifetime of a wireless sensor network (WSN). In Chapter 2, we studied two different models for the lifetime maximization in WSNs with mobile sinks. One model prohibits delay tolerance (the MSM), and the other allows it (the queue-based DT-MSM). For each of these models, we described linear programming formulations and proposed a column generation algorithm. We showed that in both models, the subproblem of the corresponding column generation algorithm can be solved by Dijkstra's algorithm. Also, for the MSM, it is shown that the problem is separable to several smaller problems. The computational results showed the efficiency of the proposed column generation algorithm for the MSM, but revealed that LP is quite efficient for the queue-based DT-MSM instances that we generated.

One of the keys to the successful application of an optimal routing and/or rate allocation algorithm to the real-world networks, including our column generation algorithms, is whether the algorithm can be implemented in a distributed manner. In general, the subproblems arising from our column generation algorithms exhibit a shortest path structure, which are amenable to decentralized implementation. However, the master problems for these algorithms are not readily solvable by a distributed algorithm. To apply the column generation algorithms, a tentative arrangement may be to let the sink solve the master problems. Despite this shortcoming, our column generation algorithms are still much more open to distributed implementation and applications to real-world networks than a standard, monolithic LP algorithm. Finding a distributed implementation of our column generation algorithms is one important future research direction. In addition, we will continue to explore decomposition approaches for related WSN lifetime problems, and determine cases in which they outperform simple linear programming approaches.

In Chapter 3, we proposed an algorithm for maximizing the WSN lifetime when there is a mobile sink and the underlying application can tolerate some degree of delay in delivering the data to the sink. Our main contribution was that the algorithm is distributed, and in addition, mostly uses local information. Such an algorithm can be implemented by parallel and/or distributed execution and the overhead of message passing is low. It is also possible to embed the algorithm into a network protocol so that the senor nodes and the sink can run it directly as part of the network operation. We also gave a proof of the algorithm's optimality and the boundedness of the queue sizes. The proof was based on analyzing a Lyapunov drift. The results and the analytical technique are both substantially different from the standard optimization theory.

In Chapter 4, we presented several methods of obtaining lower and upper bounds on the optimal objective function value of an important geometric variant of the traveling salesman problem, called the close-enough traveling salesman problem (CETSP). The ability to obtain tight lower bounds on the optimal CETSP tour length is vital in evaluating the quality of any heuristic solution to the problem. We proved several properties of an optimal CETSP tour, and used them to establish a way of partitioning the continuous solution space. This partitioning scheme is then used in formulating three different integer programming problems that yield lower and upper bounds on the optimal CETSP tour length. In particular, we formulated a lower bounding problem that is a special instance of the covering tour problem, and described an alternative formulation that yields a tighter lower bound.

We also described a way of reformulating the underlying integer program that makes it amenable to Benders decomposition. The subproblem in the decomposition scheme finds a shortest path in a special directed graph with an expanded node set. We observed that this reformulation yields a lower bound that is not dominated by that of the original formulation, while greatly enhancing the running time via Benders decomposition. An iterative refining of the underlying partitioning scheme ensures that

the lower bounds obtained from the three methods converge to the optimal CETSP tour length.

The framework presented in Chapter 4 can be extended to many other problems with continuous solution space. Candidates include the lawn mowing problem [60] and the the polygon exploration problem [61]. In some applications, a turning cost may be present based on the angle that is formed at each turn point of the tour (see, e.g., [62]). Formulation (4–20) may, in particular, be extended to address practical instances of geometric tour problems with turn costs. Another interesting line of future research would perform a comprehensive polyhedral study on the proposed integer programming problems formulated in this chapter.

In Chapter 5, we proposed exact algorithms for solving a version of the lifetime maximization problem in WSNs with a mobile sink. Our approach differs from those in the literature as it tries to find an optimal cyclic trajectory for the sink in the sensor field when the sink's travel times between its different locations are not negligible. We formulated several mixed-integer linear programs to model different variants of this problem. We then focused on one of these formulations and provided cutting-plane and decomposition algorithms to solve it. Our computational experiments indicated that the proposed algorithms can potentially improve the solvability of the underlying model.

An important future research direction would focus on developing efficient exact algorithms for MSM-FS2, which proves to be a computationally challenging problem. MSM-FS2 generally yields a higher network lifetime and seems to be more appropriate for applications in which real-time transmission of data is less crucial. Another problem may investigate the case in which the sink trajectory does not need to be a cycle and can take on more general forms, e.g., a figure-eight trajectory. Furthermore, we also suggest that our research can be extended to the case of a delay tolerant mobile sink model as formulated in [26, 56].

From a practical point of view, our models in Chapter 5 are primarily intended for finding an optimal sink trajectory. Given a specific trajectory for the sink, another future research problem would devise a distributed algorithm for finding a set of optimal routing decisions for the sensor nodes in a decentralized manner. A distributed routing algorithm might be more valuable than a centralized one in certain applications, e.g., when the routing algorithm can be built into the network protocol. While there exist several distributed algorithms for MSM (see, e.g., [9]), none of them can be directly applied to the models proposed in Chapter 5. Hence, the development of distributed algorithms for solving the models presented in Chapter 5 poses another future research challenge.

REFERENCES

[1] C. Raghavendra, K. Sivalingam, and T. Znati, *Wireless Sensor Networks*. New York: Springer, 2004.

[2] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[4] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, 2001.

[5] J. Chang and L. Tassiulas, "Routing for maximum system lifetime in wireless ad hoc networks," in *37th Annual Allertion Conf. Communication, Control, and Computing*, vol. 37, Monticello, IL, September 1999, pp. 1191–1200.

[6] ——, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 609–619, August 2004.

[7] I. Papadimitriou and L. Georgiadis, "Maximum lifetime routing to mobile sink in wireless sensor networks," in *IEEE SoftCom*, 2005.

[8] Z. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, "Exploiting sink mobility for maximizing sensor networks lifetime," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Waikoloa, HI, January 2005, pp. 287a–287a.

[9] M. Gatzianas and L. Georgiadis, "A distributed algorithm for maximum lifetime routing in sensor networks with mobile sink," *IEEE Transactions on Wireless Communications*, vol. 7, no. 3, pp. 984–994, March 2008.

[10] Y. Shi and Y. T. Hou, "Theoretical results on base station movement problem for sensor network," in *IEEE INFOCOM '08*, 2008, pp. 1–5.

[11] D. Ciullo, G. D. Celik, and E. Modiano, "Minimizing transmission energy in sensor networks via trajectory control," in *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt) 2010*, Avignon, France, 2010, pp. 132–141.

[12] R. Shuttleworth, B. L. Golden, S. Smith, and E. Wasil, "Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. L. Golden, S. Raghavan, and E. A. Wasil, Eds. Springer, 2008, vol. 43, pp. 487–501.

[13] W. K. Mennell, “Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem,” Ph.D. dissertation, The Robert H. Smith School of Business, University of Maryland, College Park, MD, 2009.

[14] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ: Princeton University Press, 2006.

[15] J. Luo and J.-P. Hubaux, “Joint mobility and routing for lifetime elongation in wireless sensor networks,” in *IEEE INFOCOM ’05*, 2005, pp. 1735–1746.

[16] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data MULEs: Modeling a three-tier architecture for sparse sensor networks,” in *The First IEEE International Workshop on Sensor Network Protocols and Applications, SNPA 2003*, Anchorage, AK, May 2003, pp. 30–41.

[17] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z. M. Wang, “A new MILP formulation and distributed protocols for wireless sensor networks lifetime maximization,” in *IEEE International Conference on Communications 2006*, June 2006, pp. 3517–3524.

[18] W. Wang, V. Srinivasan, and K.-C. Chua, “Using mobile relays to prolong the lifetime of wireless sensor networks,” in *MobiCom ’05*, 2005, pp. 270–283.

[19] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, “Energy efficient schemes for wireless sensor networks with multiple mobile base stations,” in *IEEE GLOBECOM ’03*, December 2003, pp. 377–381.

[20] J. Li and P. Mohapatra, “An analytical model for the energy hole problem in many-to-one sensor networks,” in *Proceedings of Vehicular Technology Conference*, January 2005, pp. 2721–2725.

[21] ——, “Analytical modeling and mitigation techniques for the energy hole problem in sensor networks,” *Pervasive and Mobile Computing*, vol. 3, no. 8, pp. 233–254, 2007.

[22] X. Wu, G. Chen, and S. K. Das, “Avoiding energy holes in wireless sensor networks with nonuniform node distribution,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 710–720, 2008.

[23] L. Popa, A. Rostamizadeh, R. M. Karp, and C. Papadimitriou, “Balancing traffic load in wireless networks with curveball routing,” in *MobiHoc ’07*, September 2007, pp. 170–179.

[24] R. Madan and S. Lall, “Distributed algorithms for maximum lifetime routing in wireless sensor networks,” in *IEEE GLOBECOM ’04*, 2004, pp. 748–753.

[25] A. Sankar and Z. Liu, “Maximum lifetime routing in wireless ad-hoc networks,” in *IEEE INFOCOM ’04*, vol. 2, 2004, pp. 1089–1097.

[26] Y. Yun and Y. Xia, “Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications,” *IEEE Transactions on Mobile Computing*, vol. 9, no. 9, pp. 1308 –1318, September 2010.

[27] K. M. Chandy and J. Misra, “Distributed computation on graphs: shortest path algorithms,” *Communications of the ACM*, vol. 25, no. 11, pp. 833–837, 1982.

[28] S. Yang, H. Cheng, and F. Wang, “Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 1, pp. 52–63, January 2010.

[29] G. Wang and G. Wang, “An energy-aware geographic routing algorithm for mobile ad hoc network,” in *WiCom ’09*, September 2009, pp. 1–4.

[30] W. R. Heinzelman, A. Chadrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Wailea Maui, HI, January 2000, pp. 1–10.

[31] T. S. Rappaport, *Wireless Communications: Principles and Practice*. New Jersey: Prentice Hall, 1996.

[32] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. New Jersey: John Wiley and Sons, Inc., 2005.

[33] W. B. Heinzelman, “Application specific protocol architectures for wireless networks,” Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institure of Technology, Boston, MA, 2000.

[34] X. Wu, G. Chen, and S. K. Das, “Avoiding energy holes in wireless sensor networks with nonuniform node distribution,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, May 2008.

[35] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Transactions on Automatic Control*, vol. 12, no. 12, pp. 1936–1948, December 1992.

[36] M. J. Neely, E. Modiano, and C.-P. Li, “Fairness and optimal stochastic control for heterogeneous networks,” *IEEE Transactions on Networking*, vol. 16, no. 2, April 2008.

[37] E. Horowitz, S. Sahni, and S. Rajasekaran, *Computer Algorihtms/C++*. New York: W. H. Freeman & Co., 1996.

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[39] J. R. Current and D. A. Schilling, “The covering salesman problem,” *Transportation Science*, vol. 23, no. 3, pp. 208–213, 1989.

[40] E. Balas, “The prize collecting traveling salesman problem,” *Networks*, vol. 19, no. 6, pp. 621–636, 1989.

[41] ——, “The prize collecting traveling salesman problem and its applications,” in *The Traveling Salesman Problem and its Variations*, D. Du, P. M. Pardalos, G. Gutin, and A. Punnen, Eds. Springer US, 2004, vol. 12, pp. 663–695.

[42] M. Gendreau, G. Laporte, and F. Semet, “The covering tour problem,” *Operations Research*, vol. 45, pp. 568–576, 1997.

[43] B. Yuan, M. Orlowska, and S. Sadiq, “On the optimal robot routing problem in wireless sensor networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 1252–1261, 2007.

[44] D. J. Gulczynski, J. W. Heath, and C. C. Price, “The close enough traveling salesman problem: A discussion of several heuristics,” *Perspectives in Operations Research*, vol. 36, pp. 271–283, 2006.

[45] E. M. Arkin and R. Hassin, “Approximation algorithms for the geometric covering salesman problem,” *Discrete Applied Mathematics*, vol. 55, pp. 197–218, 1994.

[46] C. S. Mata and J. S. B. Mitchell, “Approximation algorithms for geometric tour and network design problems,” in *SCG ’95: Proceedings of the Eleventh Annual Symposium on Computational Geometry*, New York, 1995, pp. 360–369.

[47] A. Dumitrescu and J. S. B. Mitchell, “Approximation algorithms for TSP with neighborhoods in the plane,” *Journal of Algorithms*, vol. 48, no. 1, pp. 135–159, 2003.

[48] M. Padberg and G. Rinaldi, “A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems,” *SIAM Review*, vol. 33, no. 1, pp. 60–100, 1991.

[49] Concorde, http://www.tsp.gatech.edu/concorde.html, 2011.

[50] M. Gendreau, G. Laporte, and F. Semet, “A branch-and-cut algorithm for the undirected selective traveling salesman problem,” *Networks*, vol. 32, no. 4, pp. 263–273, 1998.

[51] N. Jozefowiez, F. Semet, and E. Talbi, “The bi-objective covering tour problem,” *Computers & Operations Research*, vol. 34, no. 7, pp. 1929–1942, 2007.

[52] M. Fischetti, J. S. Gonzalez, and P. Toth, “A branch-and-cut algorithm for the symmetric generalized traveling salesman problem,” *Operations Research*, vol. 45, no. 3, pp. 378–394, 1997.

[53] C. A. Valle, A. S. da Cunha, G. R. Mateus, and L. C. Martinez, “Exact algorithms for a selective vehicle routing problem where the longest route is minimized,” *Electronic Notes in Discrete Mathematics*, vol. 35, pp. 133–138, 2009.

[54] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York: W. H. Freeman & Co., 1979.

[55] B. Behdani, Y. S. Yun, J. C. Smith, and Y. Xia, “Decomposition algorithms for maximizing the lifetime of wireless sensor networks with mobile sinks,” *Computers & Operations Research*, 2012 (to appear).

[56] Y. Yun, Y. Xia, B. Behdani, and J. C. Smith, “Distributed algorithm for lifetime maximization in delay-tolerant wireless sensor network with mobile sink,” in *49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010, pp. 370–375.

[57] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, “A discrete-time battery model for high-level power estimation,” in *Proceedings of the Design, Automation and Test in Europe (DATE 2000)*, Paris, France, March 2000, pp. 35–39.

[58] M. E. Keskin, I. K. Altinel, N. Aras, and C. Ersoy, “Lifetime maximization in wireless sensor networks using a mobile sink with nonzero traveling time,” *The Computer Journal*, 2012 (to appear).

[59] J. F. Benders, “Partitioning procedures for solving mixed-variables programming problems,” *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[60] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, “Approximation algorithms for lawn mowing and milling,” *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[61] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel, “The polygon exploration problem,” *SIAM Journal on Computing*, vol. 31, no. 2, pp. 577–600, 2002.

[62] E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. B. Mitchell, and S. Sethia, “Optimal covering tours with turn costs,” *SIAM Journal on Computing*, vol. 35, no. 3, pp. 531–566, 2006.

BIOGRAPHICAL SKETCH

Behnam Behdani was born in 1983 in Birjand, Iran. He received his bachelor's and master's degrees in industrial engineering from Sharif University of Technology, Tehran, Iran. He then joined the Department of Industrial and Systems Engineering at the University of Florida in August 2007. He received his Doctor of Philosophy degree in industrial and systems engineering from the University of Florida in the summer of 2012. His research interests lie in the operations research area, including integer programming, network optimization, decomposition approaches to large-scale optimization problems, and robust optimization.