Stony Brook University

Stony Brook University

Geometric Optimization Problems in Sensor Networks

A Dissertation presented

by

Gui Citovsky

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Operations Research)

Stony Brook University

May 2016

Stony Brook University

The Graduate School

Gui Citovsky

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Joseph Mitchell
Distinguished Professor. Department of Applied Mathematics and Statistics

Esther Arkin
Professor. Department of Applied Mathematics and Statistics

Zhenhua Liu
Assistant Professor. Department of Applied Mathematics and Statistics

Jie Gao
Associate Professor. Department of Computer Science

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Geometric Optimization Problems in Sensor Networks

by

Gui Citovsky

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Operations Research)

Stony Brook University

2016

In this thesis, we address a variety of algorithmic problems motivated by applications in sensor networks. In many of these problems we consider that the placement of the sensors is in a general metric space and in all of the problems we consider the case that the sensors lie in the Euclidean plane and exploit geometric properties in order to achieve better results.

We introduce the SINR_k model which is a generalization of the SINR model. Given a set of sender-receiver requests in the Euclidean plane, the goal is to minimize the number of rounds of scheduling needed to satisfy all of the requests. In order to determine whether or not receiver c successfully receives the signal sent from its paired sender s, we only consider interference from the k closest senders to c (other than s). We also consider the maximum capacity problem where the objective is to maximize the number of requests satisfied in a single round of scheduling.

We then focus on data gathering problems. In these problems we are given a set of sensors in the Euclidean plane or a general metric space, each of which generates data at a fixed rate and has a fixed capacity. Given a budget of data gathering mules, we route these mules in order to maximize their collective data gathering rate. We also look at the no data loss problem where the objective is to minimize the number of data mules needed for there not to be any data loss in the network.

Next, we consider problems in which one needs to select or cover at most one element from each tuple of a set of tuples of elements in order to optimize certain objective functions. These elements are objects in the Euclidean

plane. The applications to sensor networks are discussed later in the thesis.

Finally, we consider problems where one is given a set of pairs of points in a certain metric space with the task of partitioning the pairs into "red" and "blue" sites. Each pair must have exactly one point colored red and exactly one point colored blue. The partition should be made to optimize the cost of certain structures that will be computed on both the red points and the blue points. These types of problems are motivated by applications in sensor networks. These applications will be discussed in this thesis.

This work advances the field of sensor networks by improving on previously known results and by introducing and solving problems that have not been previously considered.

Acknowledgemets

I will forever be grateful for the constant support from my advisor, Dr. Joseph Mitchell. He has always provided me the intellectual freedom to pursue whatever research problem interests me. Joe has also allowed me and has encouraged me to experience life outside of academia in my time as a doctoral student. The freedom Joe has given me has made my experience as a doctoral student incredibly satisfying and pleasurable. Additionally, Joe taught me how to do research and more importantly how to attack and solve very difficult problems.

I am very thankful to Dr. Svetlozar (Zari) Rachev. Zari encouraged me to pursue research in Quantitative Finance and has instilled an incredible amount of confidence in my intellectual abilities. I would like to thank Dr. Sergio Focardi with whom I collaborated on nearly a year's worth of research. I am also extremely thankful to Dr. Matthew (Matya) Katz, of Ben-Gurion University (Israel), with whom I have worked very closely on many research problems. Matya has always shared interesting problems with me, including one that was the basis of my first publication.

I would like to thank my colleagues and coauthors: Dr. Esther Arkin, Dr. Rom Aschner, Dr. Aritra Banik, Dr. Paz Carmi, Dr. Jie Gao, Tyler Mayer and Jiemin Zeng. I have had countless interesting discussions with my coauthors and colleagues that have led to significant results.

Lastly, I would like to thank my parents, Dr. Nurit Ballas and Dr. Vitaly Citovsky. My parents have supported me in all aspects of life and especially in the realm of education. They strongly encouraged me to pursue a doctoral degree and I will always be grateful for this.

Table of Contents

Contents

List of Figures

List of Tables

Publications

[1] Rom Aschner, Gui Citovsky and Matthew J. Katz. Exploiting Geometry in the SINR_k Model. *ALGOSENSORS*, 125–135, 2014.

[2] Gui Citovsky, Kan Huang and Joseph S. B. Mitchell. Optimally Routing a Tracker to Maximize the Total Time a Mobile Evader is in View. *FWCG*, 2014.

[3] Gui Citovsky, Jie Gao, Joseph S. B. Mitchell and Jiemin Zeng. Exact and Approximation Algorithms for Data Mule Scheduling in a Sensor Network. *ALGOSENSORS*, 57–70, 2015. Invited to special issue of Journal Theoretical Computer Science (TCS). Selected paper from *ALGOSENSORS* 2015.

[4] Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell and Marina Simakov. Choice is Hard. *ISAAC 2015*, 318–328, 2015.

[5] Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell and Marina Simakov. Conflict-free Covering. *CCCG*, 2015.

[6] Gui Citovsky and Sergio Focardi. A Novel View of Suprathreshold Stochastic Resonance and its Applications to Financial Markets. *Frontiers in Applied Mathematics and Statistics*, 2015. Volume 1, page 10.

[7] Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Su Jia, Matthew J. Katz, Tyler Mayer and Joseph S. B. Mitchell. Network Optimization on Partitioned Pairs of Points. *Submitted for publication.*

Chapter 1

Introduction

In Chapter 2 we introduce the SINR_k model, which is a practical version of the SINR model. In the SINR_k model, in order to determine whether s's signal is received at c, where s is a sender and c is a receiver, one only considers the k most significant senders w.r.t. to c (other than s). Assuming uniform power, these are the k closest senders to c (other than s). Under this model, we consider the well-studied scheduling problem: Given a set L of sender-receiver requests, find a partition of L into a minimum number of subsets (rounds), such that in each subset all requests can be satisfied simultaneously. We present an $O(1)$-approximation algorithm for the scheduling problem (under the SINR_k model). For comparison, the best known approximation ratio under the SINR model is $O(\log n)$. We also present an $O(1)$-approximation algorithm for the maximum capacity problem (i.e., for the single round problem), obtaining a constant of approximation which is considerably better than those obtained under the SINR model. Finally, for the special case where $k = 1$, we present a PTAS for the maximum capacity problem. Our algorithms are based on geometric analysis of the SINR_k model.

Next, in Chapter 3, we consider the fundamental problem of scheduling data mules for managing a wireless sensor network. A data mule tours around a sensor network and can help with network maintenance such as data collection and battery recharging/replacement. We assume that each sensor has a fixed data generation rate and a capacity (upper bound on storage size). If the data mule arrives after the storage capacity is met, additional data generated is lost. We formulate several fundamental problems for the best schedule of single or multiple data mules and provide algorithms with provable performance. First, we consider using a single data mule to collect data from sensors, and we aim to maximize the data collection rate. We then generalize this model to consider k data mules. Additionally, we study the problem of minimizing the number of data mules such that it is possible for them to collect all data, without loss. For the above problems, when we assume that the capacities of all sensors are the same, we provide exact algorithms for special cases and constant-factor approximation algorithms for more generale cases. We also show that several of these problems are NP-

hard. When we allow sensor capacities to differ, we have a constant-factor approximation for each of the aforementioned problems when the ratio of the maximum capacity to the minimum capacity is constant.

In Chapter 4 we consider problems in which one needs to select or cover at most one element from a pair of elements in order to optimize certain objective functions. Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of color classes, where each color class C_i consists of a pair of objects. We focus on two problems in which the objects are points on the line. In the first problem (*rainbow minmax gap*), given P, one needs to select exactly one point from each color class, such that the maximum distance between a pair of consecutive selected points is minimized. This problem was studied by Consuegra and Narasimhan, who left the question of its complexity unresolved. We prove that it is NP-hard. For our proof we obtain the following auxiliary result. A 3-SAT formula is an LSAT formula if each clause (viewed as a set of literals) intersects at most one other clause, and, moreover, if two clauses intersect, then they have exactly one literal in common. We prove that the problem of deciding whether an LSAT formula is satisfiable or not is NP-complete. We present two additional applications of the LSAT result, namely, to *rainbow piercing* and *rainbow covering*. In the second problem (*covering color classes with intervals*), given P, one needs to find a minimum-cardinality set $\mathcal{I}$ of intervals, such that exactly one point from each color class is covered by an interval in $\mathcal{I}$. We consider the case that $\mathcal{I}$ must consist of only unit length intervals and the case that $\mathcal{I}$ can consist of intervals of arbitrary length. In both cases we show that this problem is NP-hard.

In Chapter 5, we follow up on the fundamental set of geometric coverage problems introduced in the second problem in Chapter 4. Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of color classes, where each color class C_i consists of a set of points. We address a family of covering problems in which each color class must be covered but no two points from the same color class are allowed to be covered by the same geometric object. We consider the case that P is restricted to be on a line and the case that P is anywhere in the Euclidean plane. We examine the following two objectives; covering at least one point from each color class and covering exactly one point from each color class. We prove that the problems in this family are NP-complete (or NP-hard) and offer several constant-factor approximation algorithms. Both of these problems have applications in sensor networks. Points of the same color class can be considered to be sensors with the same data. The objective is to retrieve data from each color class of sensors. Covering sensors with a

geometric object corresponds to having these sensors all interact with the same transceiver. In order to prevent or mitigate malicious attacks, no two sensors with the same data are allowed to interact with the same transceiver.

Finally, in Chapter 6, we study network optimization problems where one is given a set of pairs of points in some metric space with the objective of determining how to split each pair into a "red" site and a "blue site" in order to optimize two networks; one on the red sites and one on the blue sites. The partition into red and blue sets of point sites should optimize the cost of the following structures on both the red and blue network: spanning trees, traveling salesman tours or matchings. We consider several different objective functions and show that some of these problems are NP-hard. We also provide constant factor approximation algorithms for all of these problems. Pairs of points can be considered as sensors with two replications of their data. The goal is to compute two networks (a red and blue network) and to interconnect the sensors for the purpose of communication connectivity. By having two networks, one red and one blue, a malicious attack on one network will leave us protected with a network of backup sensors.

Each chapter contains a more thorough introduction than the ones provided in this introduction.

Chapter 2

Exploiting Geometry in the SINR$_k$ Model*

2.1 Introduction

The *SINR* (Signal to Interference plus Noise Ratio) model has received a lot of attention in recent years. It is considered a more realistic model for the behavior of a wireless network than the common graph-based models such as the unit disk graph, since it takes into account physical parameters such as the fading of the signal, interference caused by other transmitters and ambient noise. A fundamental problem in this context is the following: Given a set of communication sender-receiver requests, find a good scheduling for the requests. In other words, what is the minimum number of rounds needed to satisfy all the requests, such that in each round some subset of the communication links is active?

More formally, let $L = \{(c_1, s_1), (c_2, s_2), \ldots, (c_n, s_n)\}$ be a set of n pairs of points in the plane representing n (directional) links, where the points $c_1, \ldots, c_n$ represent the receivers and the points $s_1, \ldots, s_n$ represent the senders. The *length* of the link $(c_i, s_i) \in L$ is the Euclidean distance between c_i and s_i (i.e., $|c_i s_i|$) and is denoted l_i. We denote the Euclidean distance between c_i and s_j, for $j \neq i$, by l_{ij}. The set of all receivers is denoted $C = C(L)$ and the set of all senders is denoted $S = S(L)$. Finally, let p_i be the transmission power of sender s_i, for $i = 1, \ldots, n$. In the SINR model, a link (c_i, s_i) is *feasible*, if c_i receives the signal sent by s_i. That is, if the following inequality holds (assuming all senders in S are active):

$$\frac{p_i / l_i^\alpha}{\sum_{\{j : s_j \in S \setminus \{s_i\}\}} p_j / l_{ij}^\alpha + N} \geq \beta \,,$$

where $\alpha, \beta \geq 1$ and $N > 0$ are appropriate constants (α is the path-loss exponent, N is the ambient noise, and β is the threshold above which a signal is received successfully).

The *scheduling problem* is thus to partition the set of links L to a minimum number of *feasible* subsets (i.e., rounds), where a subset L_i is *feasible* if,

*This chapter is based on joint work with Rom Aschner and Matthew J. Katz. The work in this chapter appeared in *ALGOSENSORS 2014* [14].

when only the senders in $S(L_i)$ are active, each of the links in L_i is feasible. A greedy algorithm that successively finds a feasible subset of maximum cardinality of the yet unscheduled links yields an $O(\log n)$-approximation. Therefore, it is interesting to first focus on the *maximum capacity* problem, i.e., find a feasible subset of L of maximum cardinality. In other words, find a set $Q \subseteq L$, such that if only the senders in $S(Q)$ are active, then each of the links in Q is feasible, and Q is of maximum cardinality.

In the SINR model, the affectance of senders that are close to a receiver is much more significant than the affectance of those that are far from it. Moreover, in many scenarios the interference at a receiver is caused by a few nearby senders, while signals from farther senders are drastically degraded by, e.g., walls and distance. This has led us to define a restricted but more practical version of the SINR model which we name $SINR_k$.

The $SINR_k$ model. In this model, in order to determine whether a link (c, s) is feasible, one only considers the k most significant senders w.r.t. to c (other than s), which are the k closest senders to c (other than s) assuming uniform power. Formally, for a receiver c_i, let S_i^k be the set of the k most significant senders w.r.t. to c_i (other than s_i). Then, the link (c_i, s_i) is *feasible* if the following inequality holds (assuming all senders in S are active):

$$\frac{p_i/l_i^\alpha}{\sum_{\{j:s_j \in S_i^k\}} p_j/l_{ij}^\alpha + N} \geq \beta \,.$$

Assuming uniform power, we examined the validity of the SINR_k model in the specific but common setting where the senders are located on an $m \times m$ grid, for some odd integer m. Specifically, consider the sender s located at the center of the grid (i.e., at location $((m+1)/2, (m+1)/2)$. Let R denote the *reception region* of s; i.e., the region consisting of all points in the plane at which s is received according to the SINR inequality (assuming all senders are active). Avin et al. [18] showed that R is convex and fat. Let R_k denote the reception region of s according to the SINR_k inequality, i.e., when only the k closest neighbors of s are taken into account. Notice that for any two positive integers k_1, k_2, if $k_1 < k_2$, then $R_{k_1} \supset R_{k_2}$. We thus computed the region R_k for several values of k, and observed the rate at which R_k's area decreases as k increases. Consider Table 1 and Figure 1. In this example, $m = 31$ (that is, we have 961 senders), $\alpha = 4$, and $\beta = 2$. The values in the left column are those for which we computed R_k, and the values in the right column are the corresponding ratios between the area of R_k and the area of

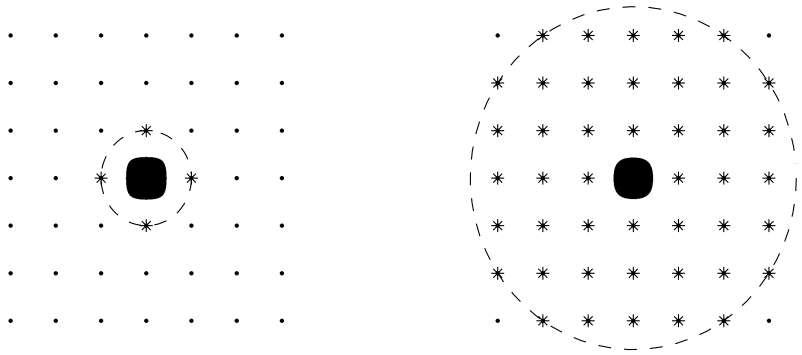
k	area(R_k)/area(R)
4	1.102
8	1.039
12	1.029
20	1.017
24	1.014
28	1.012
36	1.011
44	1.006
$31^2 - 1$	1

Table 1: The ratio area(R_k)/area(R) for several values of k, computed for a sender at the center of a 31×31 grid.

R. Notice that already for $k = 4$, R_k's area is larger than R's area by only roughly 10%, and that for $k = 44$ the difference drops to roughly 0.5%; see Figure 1.

Related work. The pioneering work of Gupta and Kumar [51] has initiated an extensive study of the maximum capacity and the scheduling problems in the SINR model. Several versions of these problem have been considered, depending on the capabilities of the underlying hardware, that is, whether and to what extent one can control the transmission power of the senders.

For the case where the transmission powers are given, Goussevskaia et al. [48] showed that the maximum capacity and the scheduling problems are NP-complete, even for uniform power. They also presented an $O(g(L))$-approximation algorithm, assuming uniform power, for the (weighted) maximum capacity problem, where $g(L)$ is the so-called diversity of the network, which can be arbitrarily large in general. Assuming uniform power, Chafekar et al. [28] presented an $O(\log \Delta)$-approximation algorithm for the maximum capacity problem, where Δ is the ratio between the longest link and the shortest link. If the ratio between the maximum power and the minimum power is bounded by Γ, then they give an $O(\log \Delta \log \Gamma)$-approximation algorithm for the problem. Goussevskaia et al. [49] and Halldórsson and Wattenhofer [56] gave constant-factor approximation algorithms for the maximum capacity problem yielding an $O(\log n)$-approximation algorithm for the scheduling problem, assuming uniform power. In [49] they note that their



(a) (b)

Figure 1: The 7×7 neighborhood of a sender s located at the center of a 31×31 grid. (a) R_4, the reception region of s for $k = 4$ (the black spot around the center). (b) R_{44}, the reception region of s for $k = 44$ (the black spot around the center).

$O(1)$-approximation algorithm also applies to the case where the ratio between the maximum power and the minimum power is bounded by a constant and for the case where the number of different power levels is constant. Later, Wan et al. [91] presented a constant-factor approximation algorithm for the maximum capacity problem, assuming uniform power; their constant is significantly better than the one in [49]. Recently, Halldórsson and Mitra [55] have considered the case of oblivious power. This is a special case of non-uniform power where the power of a link is a simple function of the link's length. They gave an $O(1)$-approximation algorithm for the maximum capacity problem, yielding an $O(\log n)$-approximation algorithm for scheduling. Finally, the case with (full) power control has also been studied, see, e.g., [54, 55, 63, 74].

Our results. We study the maximum capacity and scheduling problems in the SINR_k model, for a given constant k, under the common assumptions that (i) $p_i = p_j$, for $1 \leq i, j \leq n$, i.e., uniform power (see, e.g., [49,91]), and (ii) N $= 0$, i.e., there is no ambient noise (see, e.g., [48]). We exploit some of the geometric properties of the SINR_k model to obtain $O(1)$-approximation algorithms for both problems. For comparison, the best known approximation

ratio for the scheduling problem in the SINR model is $O(\log n)$. We also consider a variant of the maximum capacity problem in which one is free to form the links, and the goal, as in the standard problem, is to find a maximum-cardinality feasible subset of links. We obtain an $O(1)$-approximation algorithm for this variant as well. Finally, for $k = 1$, we present a PTAS for the maximum capacity problem.

This chapter is organized as follows: In Section 2.2, we prove several geometric properties of the SINR_k model and use them to obtain an $O(1)$-approximation algorithm for the maximum capacity problem, where the constant that we get is significantly better than the one in [49]. In Section 2.3, we present an $O(1)$-approximation algorithm for the scheduling problem. Finally, in Section 2.4, we show that in the special case where $k = 1$, one can obtain a PTAS for the capacity problem, by using a technique due to Chan [29] that is based on geometric separators. To the best of our knowledge our work is the first to study the SINR_k model.

2.2 Maximum capacity

Let k be a positive integer. In this section we consider the *maximum capacity* problem under the SINR_k model, assuming uniform power and no ambient noise. W.l.o.g., we shall assume that the transmission power of each of the senders is 1. Let $L' \subseteq L$ and (c_i, s_i) a link in L'. We say that (c_i, s_i) is *feasible* (in L'), if c_i receives s_i when only the senders in $S(L')$ are active. If all the links in L' are feasible, then we say that L' is *feasible*. Our goal is to find a feasible subset of links of maximum cardinality. We begin by proving a series of lemmas establishing several important geometric properties of feasible links.

Lemma 1. *Let $L' \subseteq L$ and let $(c_i, s_i) \in L'$ be a feasible link. Then, the disk centered at c_i of radius $\sqrt[\alpha]{\beta} \cdot l_i$ does not contain in its interior any sender of $S(L')$ except for s_i.*

Proof. Assume that this disk contains another sender (except s_i) in its interior. Let s_r be such a sender, i.e., $l_{ir} < \sqrt[\alpha]{\beta} l_i$. Then

$$\frac{1/l_i^\alpha}{\Sigma_{\{j:s_j \in S_i^k\}} 1/l_{ij}^\alpha} \leq \frac{1/l_i^\alpha}{1/l_{ir}^\alpha} < \beta \,,$$

where $S_i^k \subseteq S(L')$ is the set of the k closest senders to c_i, not including s_i. This is a contradiction to the assumption that (c_i, s_i) is a feasible link in

L'. □

Lemma 2. *Let $L' \subseteq L$ and let $(c_i, s_i), (c_j, s_j) \in L'$ be two feasible links. Let $D_i = D(c_i, m \cdot l_i)$, $D_j = D(c_j, m \cdot l_j)$ be two disks around the two receivers. If $m < \frac{\sqrt[\alpha]{\beta}-1}{2}$, then $D_i \cap D_j = \emptyset$.*

Proof. By Lemma 1, $l_{ij} \geq \sqrt[\alpha]{\beta} \cdot l_i$ and $l_{ji} \geq \sqrt[\alpha]{\beta} \cdot l_j$, that is, $l_i + l_j \leq \frac{1}{\sqrt[\alpha]{\beta}}(l_{ij} + l_{ji})$. By the triangle inequality, $l_{ij} \leq |c_i c_j| + l_j$ and $l_{ji} \leq |c_j c_i| + l_i$, and therefore $l_i + l_j \leq \frac{1}{\sqrt[\alpha]{\beta}}(2|c_i c_j| + l_i + l_j)$. Rearranging, we get that $|c_i c_j| \geq \frac{\sqrt[\alpha]{\beta}-1}{2}(l_i + l_j) > m(l_i + l_j) = ml_i + ml_j$. This implies that, $D_i \cap D_j = \emptyset$. □

The following lemma is actually a generalization of Lemma 1.

Lemma 3. *Let $L' \subseteq L$ and let $(c_i, s_i) \in L'$ be a feasible link. Then, the disk centered at c_i of radius $\sqrt[\alpha]{\beta k} \cdot l_i$ contains in its interior at most $k-1$ senders of $S(L') \setminus \{s_i\}$.*

Proof.

$$\beta \leq \frac{1/l_i^\alpha}{\Sigma_{\{j:s_j \in S_i^k\}} 1/l_{ij}^\alpha} \leq \frac{1/l_i^\alpha}{k \min_{\{j:s_j \in S_i^k\}} \{1/{l_{ij}}^\alpha\}} \leq \frac{\max_{\{j:s_j \in S_i^k\}} \{l_{ij}^\alpha\}}{k l_i^\alpha}.$$

Thus, $\sqrt[\alpha]{\beta k} \cdot l_i \leq \max_{\{j:s_j \in S_i^k\}} \{l_{ij}\}$. That is, the farthest among the k senders in S_i^k does not lie in the interior of the disk centered at c_i of radius $\sqrt[\alpha]{\beta k} \cdot l_i$. □

Lemma 4. *Let $L' \subseteq L$ and let $(c_i, s_i) \in L'$. If the disk centered at c_i of radius $\sqrt[\alpha]{\beta k} \cdot l_i$ does not contain in its interior any sender of $S(L')$ (except for s_i), then the link (c_i, s_i) is feasible.*

Proof. For each $s_j \in S_i^k$, we have that $l_{ij} \geq \sqrt[\alpha]{\beta k} \cdot l_i$. Therefore, $\Sigma_{\{j:s_j \in S_i^k\}} \frac{1}{l_{ij}^\alpha} \leq \frac{1}{\beta l_i^\alpha}$, and $\frac{1/l_i^\alpha}{\Sigma_{\{j:s_j \in S_i^k\}} 1/l_{ij}^\alpha} \geq \beta$. □

2.2.1 An $O(1)$-approximation for constant k

For each $(c_i, s_i) \in L$, let D_i denote the disk of radius $\sqrt[\alpha]{\beta k} \cdot l_i$ centered at c_i, and set $\mathcal{D} = \{D_i | (c_i, s_i) \in L\}$.

We apply the following simple (and well-known) algorithm that finds an independent set $\mathcal{Q}$ in the intersection graph induced by $\mathcal{D}$, such that $|\mathcal{Q}|$ is at least some constant fraction of the size of a maximum independent set

in this graph. We then prove that the set of links corresponding to $\mathcal{Q}$ is an $O(1)$-approximation of OPT, where OPT is an optimal solution for the maximum capacity problem (under SINR_k). This proof is non-trivial since the disks in $\mathcal{D}$ corresponding to the links in OPT are not necessarily disjoint.

Algorithm 1 An $O(1)$-approximation

$\mathcal{Q} \leftarrow \emptyset$
Sort $\mathcal{D}$ by the radii of the disks in increasing order.
while $\mathcal{D} \neq \emptyset$ **do**
  Let D be the smallest disk in $\mathcal{D}$
  $\mathcal{D} \leftarrow \mathcal{D} \setminus \{D\}$
  for all $D' \in \mathcal{D}$, such that $D \cap D' \neq \emptyset$ **do**
    $\mathcal{D} \leftarrow \mathcal{D} \setminus \{D'\}$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{D\}$
return $\mathcal{Q}$

Algorithm 1 returns a subset $\mathcal{Q} \subseteq \mathcal{D}$ which is an independent set, i.e., for any two disks $D_1, D_2 \in \mathcal{Q}$, $D_1 \cap D_2 = \emptyset$. Moreover, by Lemma 4, the subset of links corresponding to $\mathcal{Q}$ is feasible. From now on, we shall mostly think of OPT as a set of disks, i.e., the subset of disks in $\mathcal{D}$ corresponding to the links in OPT. Below we show that $|OPT| = O(|\mathcal{Q}|)$.

Lemma 5. *Let $L' \subseteq L$ be a feasible set of links, and let $D(L')$ denote the set of corresponding disks of radius $\sqrt[\alpha]{\beta k} \cdot l_i$ around the receivers c_i in $C(L')$. Then, every point $p \in \mathbb{R}^2$ is covered by at most $\tau = \frac{2\pi(k+1)}{\arctan(\frac{\sqrt[\alpha]{\beta k}-1}{\sqrt[\alpha]{\beta k}+1})}$ disks in $D(L')$.*

Proof. Let $p \in \mathbb{R}^2$, and consider the set $D(L_p) \subseteq D(L')$ of all disks in $D(L')$ that cover p. Let s_i be the sender (among the senders in $S(L_p)$) that is farthest from p. We draw a wedge W of angle 2θ and apex p, where $\theta = \arctan(\frac{\sqrt[\alpha]{\beta k}-1}{\sqrt[\alpha]{\beta k}+1})$, such that s_i is on its bisector (see Figure 2). We claim that the disk D_i (of radius $\sqrt[\alpha]{\beta k} \cdot l_i$ and center c_i) covers all the senders in $S(L_p) \cap W$.

Consider the line perpendicular to $\overline{ps_i}$ and passing through s_i, and let x and y be the intersection points of this line with W's rays. Then,

$$|xs_i| = |ps_i| \tan(\theta) \leq (|pc_i| + l_i) \tan(\theta) \leq (\sqrt[\alpha]{\beta k} + 1) l_i \tan(\theta) = (\sqrt[\alpha]{\beta k} - 1) l_i \ .$$

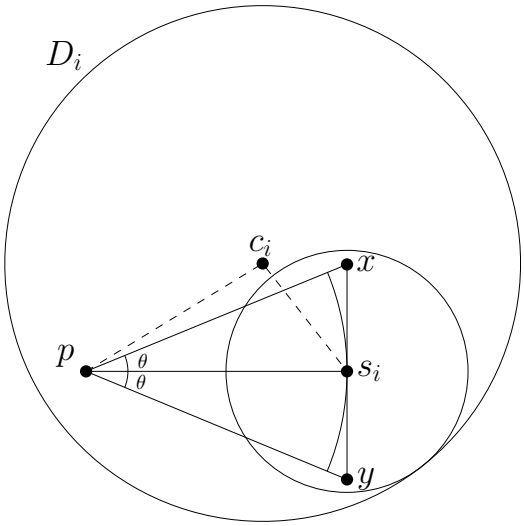


Figure 2: Proof of Lemma 5.

Similarly, $|ys_i| \leq (\sqrt[\alpha]{\beta k} - 1)l_i$. Therefore the disk of radius $(\sqrt[\alpha]{\beta k} - 1)l_i$ and center s_i contains points x and y. But this disk is contained in the disk D_i. So, D_i contains the triangle $\triangle pxy$ (since it covers its three corners), and, since all the senders in $S(L_p) \cap W$ lie in $\triangle pxy$, we conclude that D_i covers all these senders. This implies that the number of these senders is at most $k + 1$.

We now remove all the senders in $S(L_p) \cap W$ and repeat. After at most $2\pi/\theta$ iterations, we finish removing all senders in $S(L_p)$. Thus, the number of senders in $S(L_p)$ is at most τ, implying that the number of disks in $D(L')$ covering p is at most τ. □

Lemma 6. $|OPT| \leq 8\tau|\mathcal{Q}|$.

Proof. First notice that each disk in OPT intersects at least one of the disks in $\mathcal{Q}$. Since, otherwise, consider the smallest disk in OPT that does not intersect any of the disks in $\mathcal{Q}$. Then, our algorithm would have chosen this disk – contradiction. We thus associate each disk in OPT with the smallest disk in $\mathcal{Q}$ which it intersects. Let $D = D(c, r) \in \mathcal{Q}$. We show that the number of disks associated with D is at most 8τ. We first observe that each of the disks associated with D is at least as large as D. Since, if one or more of these disks were smaller than D, then our algorithm would have chosen the smallest of them instead of D – contradiction.

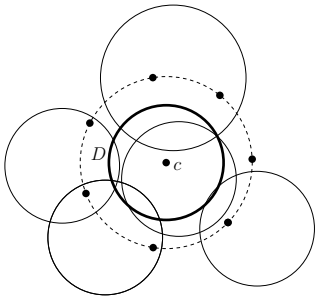


Figure 3: Any disk that is not smaller than D and intersects D covers at least one of the 8 points.

Let A be a set of 8 points including (i) the center point c, and (ii) seven points evenly spaced on a circle of radius $3r/2$ around c; see Figure 3. Notice that any disk that is not smaller than D and intersects D must cover at least one of the points in A. In particular, this is true for each of the disks in OPT associated with D. By Lemma 5, there are at most τ disks in OPT covering each of these points. Thus, at most 8τ disks in OPT intersect D. We conclude that our algorithm computes a $(1/8\tau)$-approximation of OPT. $\square$

The following theorem summarizes the main result of this section.

Theorem 1. *Given a set L of n links and a constant k, one can compute a $(1/8\tau)$-approximation for the maximum capacity problem under the $SINR_k$ model, where $\tau = \frac{2\pi(k+1)}{\arctan(\frac{\sqrt[\alpha]{\beta k}-1}{\sqrt[\alpha]{\beta k}+1})}$.*

2.2.2 All pairs maximum capacity

We now consider the maximum capacity problem where any sender and receiver can be paired. Let $L = \{(c_i, s_j) | 1 \leq i, j \leq n\}$ be a set of n^2 potential links. We seek a feasible subset of links $\mathcal{Q} \subseteq L$ of maximum cardinality, enforcing a one-to-one correspondence between $S(\mathcal{Q})$ and $C(\mathcal{Q})$.

For each $(c_i, s_j) \in L$, let D_{ij} denote the disk of radius $\sqrt[\alpha]{\beta k} \cdot l_{ij}$ centered at c_i (where $l_{ii} = l_i$), and set $\mathcal{D} = \{D_{ij} | (c_i, s_j) \in L\}$. We apply Algorithm 1 with $\mathcal{D}$ as our input set of disks. Note that any time a pair (c_i, s_j) is added to $\mathcal{Q}$, all other potential links in L that contain c_i as a receiver or s_j as a sender will be removed from consideration. This is because all other disks in $\mathcal{D}$ either using c_i as receiver or s_j as a sender clearly have a nonempty intersection

with D_{ij}. Lemma 6 shows that Algorithm 1 gives an $O(1)$-approximation for the all pairs version as well. Namely, $|OPT| \leq 8\tau|\mathcal{Q}|$.

2.3 Scheduling

In this section we consider the scheduling problem. That is, given a set L of links (i.e., requests), how many rounds are needed to satisfy all the requests? Alternatively, find a partition of L into a minimum number of feasible subsets.

We show how to obtain a constant factor approximation for the scheduling problem under the SINR_k model. As in the previous section, for each $(c_i, s_i) \in L$, let D_i denote the disk of radius $\sqrt[\alpha]{\beta k} \cdot l_i$ centered at c_i, and set $\mathcal{D} = \{D_i | (c_i, s_i) \in L\}$. The *depth* of a point $p \in \mathbb{R}^2$ with respect to $\mathcal{D}$ is the number of disks in $\mathcal{D}$ covering p. The *depth* of $\mathcal{D}$ is the depth of a point $p \in \mathbb{R}^2$ of maximum depth (i.e., it is the depth of the arrangement of the disks in $\mathcal{D}$). Notice that the depth of $\mathcal{D}$ is not necessarily bounded.

Let r be the number of rounds in an optimal solution, OPT, to the scheduling problem. We first observe that the depth of $\mathcal{D}$ is $O(r)$.

Lemma 7. *The depth of $\mathcal{D}$ is $O(r)$, where r is the number of rounds in OPT.*

Proof. Let L_i be the set of active links in round i, for $1 \leq i \leq r$. By Lemma 5 every point in the plane is covered by at most τ disks in $D(L_i)$ (i.e., the depth of $D(L_i)$ is at most τ). Therefore, the depth of $D(L)$ is at most τr. □

Miller et al. [72] showed how to color an intersection graph of a set of balls in $\mathbb{R}^d$ of bounded ply. In particular, their result implies a polynomial-time algorithm for coloring the intersection graph of the disks in $\mathcal{D}$ with $9\tau r + 1$ colors. Each color class is an independent set, and thus, by Lemma 4, is a feasible solution.

The following theorem summarizes the main result of this section.

Theorem 2. *Given a set L of n links and a constant k, one can compute a $(9\tau + 1)$-approximation for the scheduling problem under the SINR_k model.*

2.4 A PTAS for maximum capacity with $k = 1$

By plugging $k = 1$ in Lemmas 3 and 4, we obtain the following lemma.

Lemma 8. *Let $L' \subseteq L$ and let $(c_i, s_i) \in L'$. Then, (c_i, s_i) is a feasible link if and only if the disk centered at c_i of radius $\sqrt[\alpha]{\beta} \cdot l_i$ does not contain in its interior any sender of $S(L')$ (except for s_i).*

The following theorem is due to Timothy Chan [29].

Theorem 3 ([29])**.** *Given a measure μ satisfying the following five conditions, a collection $\mathcal{O}$ of n objects in $\mathbb{R}^d$ and $\varepsilon > 0$, one can find a $(1+\varepsilon)$-approximation to $\mu(\mathcal{O})$ in $O(n^{O(1/\varepsilon^d)})$ time and $O(n)$ space.*

1. *If $\mathcal{A} \subseteq \mathcal{B}$, then $\mu(\mathcal{A}) \leq \mu(\mathcal{B})$.*
2. *$\mu(\mathcal{A} \cup \mathcal{B}) \leq \mu(\mathcal{A}) + \mu(\mathcal{B})$.*
3. *If for any pair $(A, B) \in \mathcal{A} \times \mathcal{B}$, $A \cap B = \emptyset$, then $\mu(\mathcal{A} \cup \mathcal{B}) = \mu(\mathcal{A}) + \mu(\mathcal{B})$.*
4. *Given any r and size-r box R, if every object in $\mathcal{A}$ intersects R and has size at least r, then $\mu(\mathcal{A}) \leq c$ for a constant c.*
5. *A constant-factor approximation to $\mu(\mathcal{A})$ can be computed in time $|\mathcal{A}|^{O(1)}$. If $\mu(\mathcal{A}) \leq b$, then $\mu(\mathcal{A})$ can be computed exactly in time $|\mathcal{A}|^{O(b)}$ and linear space.*

Chan has applied this theorem to the measures pack$(\cdot)$ and pierce$(\cdot)$ and a collection of fat objects. We apply this theorem in a somewhat non-standard manner to obtain our PTAS.

For each $(c_i, s_i) \in L$, let D_i denote the disk of radius $\sqrt[\alpha]{\beta} \cdot l_i$ centered at c_i, and set $\mathcal{D} = \{D_i | (c_i, s_i) \in L\}$. For any $\mathcal{A} \subseteq \mathcal{D}$, let $\mu(\mathcal{A})$ denote the cardinality of a feasible subset of $\mathcal{A}$ of maximum cardinality. Below, we show that μ satisfies the five conditions above.

Notice first that two disks D_1 and D_2 in a feasible subset $\mathcal{D}'$ of $\mathcal{D}$ may intersect; in particular, one or both of the receivers c_1, c_2 may lie in the other disk. However, none of the senders s_1, s_2 may lie in the other disk. Conditions (1) and (2) are clearly satisfied. Condition (3) is also satisfied, since the assumption implies that none of the senders corresponding to the disks in $\mathcal{B}$ lies in a disk of $\mathcal{A}$ and vise versa. Concerning Condition (4), we can apply Lemma 5 in a similar way to the one described in the proof of Lemma 6, to show that (under the assumption of Condition (4)) $\mu(\mathcal{A})$ is bounded by some constant. Finally, Algorithm 1 computes a constant-factor approximation to $\mu(\mathcal{A})$ in time $|\mathcal{A}|^{O(1)}$.

The following theorem summarizes the main result of this section.

Theorem 4. *Given a set L of n links and $\varepsilon > 0$, one can compute a $(1-\varepsilon)$-approximation for the maximum capacity problem under the $SINR_1$ model.*

Remark: Notice that the only condition that is not satisfied when k is a constant greater than 1, is Condition (3). The reason for this is that for $k > 1$ the converse of Lemmas 3 is no longer true.

Chapter 3

Exact and Approximation Algorithms for Data Mule Scheduling in a Sensor Network[†]

3.1 Introduction

A number of sensor network designs integrate both static sensor nodes and more powerful mobile nodes, called *data mules*, that serve and help to manage the sensor nodes [60, 61, 82, 84]. The motivation for such designs are twofold. First, there are fundamental limitations with the flat topology of static sensors using short range wireless communication. It is known that such a topology does not scale – the network throughput will diminish if the number of sensors goes to infinity [52], while allowing node mobility will help [50]. Second, a number of fundamental network operations can benefit substantially from mobile nodes. We consider two example scenarios: sensor data collection and battery recharging. In both cases, data mules that tour around the sensors periodically can be used to maintain the normal functionality of the sensors. In addition, data collection by sensors using multi-hop routing to a fixed base station often suffers from the bottleneck issue near the base station, both in terms of communication and energy usage. Using short range wireless communication with a mobile base station can fundamentally remove such dependency and avoid the single point of failure [90].

Despite the potential benefits of introducing data mules with static sensors, a lot of new challenges emerge at the interface of coordinating the data mules with sensors. One of the most prominent challenges is the scheduling of data mule mobility to serve the sensors in a timely and energy efficient manner. This has been an active research topic for the past few years. However, as surveyed later, most prior work is evaluated by simulations or experiments [4]; algorithms with provable guarantees are scarce. We formulate data mule scheduling problems with natural objective functions and provide exact and approximation algorithms.

Our Problem. Suppose there are n sensors and a data mule traveling at a constant speed s to collect data from these sensors. A sensor i generates data

[†]This chapter is based on joint work with Jie Gao, Joseph S. B. Mitchell and Jiemin Zeng. The work in this chapter appeared in *ALGOSENSORS 2015* [33].

at a fixed rate of r_i and has a storage ("bucket") capacity of c_i where $c_i \geq r_i$. When a data mule visits a sensor, all current data stored in the sensor is collected onto the mule. We assume that the mule has unbounded storage capacity. We also assume that data collection at each sensor happens instantaneously, i.e., we ignore the time of data transmission, which is typically much smaller than the time taken by the mule to move between the sensors. If the amount of data generated at a sensor goes beyond its capacity (i.e., its bucket is full), additional data generated is lost. Thus, a natural objective is to schedule data mules to efficiently collect the continuously generated data.

We assume that the data collection and the data mule movement continues indefinitely in time. Therefore, we are mainly concerned about the long-term data gathering efficiency by periodic schedules.

The same problem arises in the case of battery recharging and energy management. In that case, each sensor i uses its battery with capacity c_i at a rate of r_i. When the battery at a sensor is depleted the sensor becomes ineffective. Thus, one would like to minimize the total amount of time of ineffectiveness, over all sensors. We formulate the following three problems.

- **Single Mule Scheduling:** Find a route for a single data mule to collect data from the sensors that maximizes the data collection rate (the average amount of data collected per time unit).
- **k-Mule Scheduling:** Given a budget of k data mules, find routes for them to maximize the rate of data collected from the sensors.
- **No Data Loss Scheduling:** Find the minimum number of data mules, and their schedules, such that all data from all sensors is collected (there is no data loss).

Our Results. We report hardness results, exact algorithms for a few special cases, and approximation algorithms for all three problems. Our algorithmic results are summarized in Table 3.1. When we assume that the capacities of all sensors are the same, we provide results for the different cases where the sensors lie in different metrics. For the case where the capacities of the sensors are different, we provide general results.

Without loss of generality, we assume that the minimum data rate is 1 and the mule velocity is 1. In fact, we can further assume that all sensors have a data rate of 1; if a sensor has data rate $r_i > 1$, we can replicate this sensor with r_i copies, each with unit data rate and capacity c_i/r_i. Thus, in

With Sensors	Single mule	k-mule	No Data Loss
on a Line	exact	$\frac{1}{3}$	exact
on a Tree	exact pseudo-polynomial	$\frac{1}{3}(1-1/e^{\frac{1}{2+\varepsilon}})$	12
General Metric Space	$1/6-\varepsilon$		
Euclidean Space	$1/3-\varepsilon$	$\frac{1}{3}(1-1/e^{1-\varepsilon})$	
with Different Capacities	$O(\frac{1}{m})$		$O(m)$

Table 2: Our approximation algorithm results for different settings. Note that $m \leq \log(\frac{c_{max}}{c_{min}})$ where c_{max} is the largest capacity and c_{min} is the smallest capacity. For the results in the first four rows, we assume that the sensor capacities are all the same. ε is any positive constant.

the following discussion we focus on the case of all sensors having unit data rates and possibly different capacities. When we consider the case where all sensors have the same capacity, we simplify notation and let the capacity of all sensors be c.

We give the first algorithms for such data mule scheduling problems with provable guarantees. In addition, we provide upper and lower bounds on the optimal solution for both problems, and we evaluate the performance using simulations, for a variety of sensor distributions and densities.

3.2 Related Work

Vehicle Routing Problems. The problems we study belong to the general family of vehicle routing problems (VRPs) and traveling salesman problems (TSPs) with constraints [17, 69, 79, 93]. But our problem is the first one considering periodically regenerated rewards/prizes and thus is the first of this type.

Related TSP variations stem from the Prize-Collecting Traveling Salesman Problem (PCTSP) [19, 24] which was originally defined by Balas [20] as the problem, given a set of cities with associated prizes and a prize quota to reach, find a path/tour on a subset of the cities such that the quota is met, while minimizing the total distance plus penalties for the cities skipped. (Some recent formulations of this problem do not include penalties for skipped cities.) Archer et. al. [5] provided a $(2-\varepsilon)$-approximation algorithm for this formulation of PCTSP where $\varepsilon \approx 0.007024$.

The Orienteering Problem [47, 88] assigns a prize to each city and, given a constraint on the length of the path, aims to maximize the total prize

collected. For the rooted version in a general metric space, Blum et. al. [26] had proven that the problem is APX-hard and provided a 4-approximation algorithm which was improved to a 3-approximation by Bansal et. al. [21] and finally to a $(2+\varepsilon)$-approximation by Chekuri et. al. [30]. For the rooted version in $\Re^2$, Arkin et. al. [7] give a 2-approximation, which was improved to a $(1+\varepsilon)$-approximation (PTAS) [31] for fixed dimension Euclidean space. For additional information we refer to the review papers [44, 88].

Similar to our problems, the Profitable Tour Problem [16] balances the two competing objectives of maximizing total prize collected and minimizing tour length. In some problems, the profit collected is dependent on the latency [36].

Our problems are also very similar to many multi-vehicle routing problems [6, 42, 43, 66]. Arkin et. al. [13] give constant-factor approximation algorithms for some types of multiple vehicle routing problems including a 3-approximation for the problem of finding a minimum number of tours shorter than a given bound that cover a given graph. Nagarajan and Ravi [76] provide a 2-approximation for tree metrics and a bicriteria approximation algorithm for general metrics. Khani and Salavatipour [64] present a 2.5-approximation algorithm for the problem of finding, for a given graph and bound λ, the minimum number of trees, each of weight at most λ, to cover the graph (improving on a bound given in [13]).

Data Mule Scheduling. Increasingly, there has been interest in using mobile data mules to collect data in sensor networks. A common question that has arisen is how to schedule multiple mules effectively and efficiently. Many heuristics have been proposed to schedule multiple mules with various constraints and objective functions (e.g., evenly distributing loads [60], scheduling short path lengths [71, 92], and minimizing energy [3]). Somasundara et. al. [85] address a very similar problem to ours, but with different methods; we obtain provable polynomial-time algorithms, while they employ (worst case exponential-time) integer linear programming and explore heuristics.

3.3 Single Mule Scheduling

Given a single mobile data mule with unit velocity, n sensors with uniform capacity and unit data rate, the goal is to route the mule in effort to maximize its data gathering rate. We explore this problem with sensors on a line, on a

tree, and in other metric spaces.

3.3.1 Exact Algorithms on a Line or a Tree

3.3.1.1 Line Case

We first look at the case when the sensors are on a line. We assume that the input data is integral; specifically, the sensors p_i are located at integer coordinates and the capacities c_i for all i are integers. With this assumption, the optimal schedule can be shown to be periodic.

Lemma 9. *The optimal schedule that minimizes data loss is periodic, assuming integral input data.*

Proof. If the sensors are located at integral positions, the distances between any two of them are integers as well. Thus, all states of the problem can be encoded by the position of the mule and the current amount of data at each sensor i. All of these values are integers. Thus, the total number of possible states is finite; after a state reappears we realize that the robot must follow the same schedule, making the schedule periodic. □

Theorem 5. *Let there be n sensors, $p_1, p_2, \ldots, p_n$ on a line. Assume that the capacities and rates of all sensors are the same: $c_i = c$ and $r_i = 1$, for $1 \leq i \leq n$. Then there exists an optimal path that minimizes data loss with the following properties: (1) its leftmost and rightmost points are at sensors, (2) it is a path making U-turns only at the leftmost and rightmost sensors.*

Despite the simple and clean statement, the proof is in fact fairly technical. To provide intuition for Theorem 5, note that paths that have U-turns not at the outermost points are making a tradeoff of collecting more data from middle sensors at the cost of having more overflow at the outer sensors. If this tradeoff is worth it, then we can show it is also worth it to forgo collecting data from some of the outer sensors. The main technical challenge is to figure out and compare the data rate between the two choices. Below is the proof of Theorem 5.

Proof. The first claim is obvious since one can remove the portion of the path beyond the leftmost sensor and shorten the time for one period of the trip, which results in all covered sensors being visited more frequently than

before. Now, let there be an optimal path P with leftmost sensor at x_l and rightmost sensor at x_r.

Let us assume that P contains U-turns that do not occur at the extreme points x_l, x_r. A U-turn is called a left U-turn if the mule was traveling from right to left before the turn, and a right U-turn otherwise. Note that one cycle of P can be split into two sub-paths, a path P_l from x_l to x_r and a path P_r from x_r to x_l. These sub-paths may have U-turns at the extreme points x_l, x_r. (In that case, P_l and P_r are not unique.) Without loss of generality, assume that there are U-turns in the sub-path from x_l to x_r. Consider the closest left U-turn to x_l and name the point it occurs at z_l. Since P is periodic, we can assume that in one full period, P starts and ends at x_l.

Now, we will modify the trip P. Notice that for the mule to make a left U-turn at z_l, it must make a right U-turn in the path from x_l to z_l. Let the rightmost such U-turn be z_r. We remove the path that begins at the first time P_l reaches z_l to the last time P_l reaches z_l. See Figure 4 for an example.

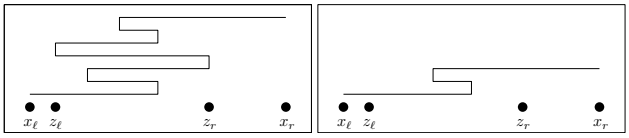


Figure 4: Remove a segment of the path to eliminate a U-turn at z_l.

After the surgery, we get a schedule without the round trip beginning at z_l. Suppose the trip eliminated has duration δ and the trip after this surgery, denoted by P', has round-trip time length T. That means, the trip P has duration $T + \delta$.

Since the rate of data accumulation and the bucket size are the same for all sensors, all sensors take the same amount of time to overflow. As a result, when a mule travels from right to left and encounters an overflowing sensor b, then all sensors to the left of b are also overflowing – since the last time they were visited was definitely before the last time that b was visited.

In the shortened path P', the sensors to the left of z_l are reached δ time sooner than in P. In the paths P and P', all sensors to the left of z_l overflow in both paths or there exists a sensor to the left of z_l that does not overflow in either P or P'. Let the leftmost such sensor be y_l. We define b_l to be the leftmost of either z_l or y_l. Let n_l denote the number of sensors to the left of b_l. Since all sensors to the left of b_l overflow in both P and P', the amount of data lost in those sensors is δn_l. Also, note that there exists a path in either

P or P' from y_l to x_r to y_l where none of the sensors from x_r to y_l overflow. The terms b_r, y_r, and n_r are defined similarly in the right direction.

Let m denote the amount of overflow for sensors covered by P' in one full period and let n'' be the number of sensors not covered by the path P' (i.e., those that are either to the left of x_l or the right of x_r). The overflow rate is $m/T + n''$ in path P' and for P it is $\frac{m+\delta(n_l+n_r)+e}{T+\delta} + n''$ where e is any extra overflow in P that is not accounted for.

Since P is an optimal path, then $\frac{m+\delta(n_l+n_r)+e}{T+\delta} + n'' \leq m/T + n''$. We simplify this relationship to

$$n_l + n_r + \frac{e}{\delta} \leq \frac{m}{T}. \tag{1}$$

Consider a simple periodic path P'' only making U-turns at b_l and b_r. If b_l or b_r is defined by a non-overflowing sensor, then there is no overflow between b_l and b_r since there exists a longer path between b_l and x_r (or b_r and x_l) in P or P' that does not cause any sensor to overflow. Thus the overflow rate of the path is just $n_l + n_r + n''$. If b_l is z_l and b_r is z_r then the overflow rate of the path is $n_l + n_r + \frac{e'}{\delta} + n''$ where e' is the extra overflow. Note that by definition, $e' \leq e$ since a cycle of P'' is shorter than the path cut out in P. For either case, the overflow rate of the direct periodic path between b_l and b_r is bound by $n_l + n_r + \frac{e'}{\delta} + n''$. By Equation 1,

$$n_l + n_r + \frac{e'}{\delta} + n'' \leq \frac{m}{T} + n''$$

$$\frac{(T+\delta)(n_l + n_r + \frac{e'}{\delta})}{T+\delta} \leq \frac{m}{T+\delta} + \frac{\delta(n_l + n_r + (\frac{e}{\delta}))}{T+\delta}$$

$$n_l + n_r + \frac{e'}{\delta} + n'' \leq \frac{m + \delta(n_l + n_r) + e}{T+\delta} + n''$$

Since P is optimal, the direct path must be optimal as well. Therefore, there exists an optimal path that is direct with only U-turns at the extreme points. □

The immediate consequence from Theorem 5 is that one can find the optimal schedule in $O(n^2)$ time, enumerating all possible pairs of extreme points.

It is important to note that it is sometimes necessary for the mule in the optimal solution to gather data more than once from a given sensor in

a period. In Figure 5, sensors are split into six groups, where each group has either k or $2k$ sensors. Within each group, each sensor has the same x-coordinate. In the optimal solution, the data mule traverses the entire interval back and forth, picking up data whenever it reaches a sensor. This solution has data gathering rate $\frac{10.5k}{2} = 5.25k$. In comparison, the best solution that gathers data from a sensor at most once per period has rate $4k$.

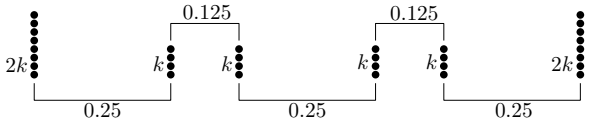


Figure 5: The optimal solution repeats sensors

3.3.1.2 Tree Case

We extend our results to a tree topology, with the sensors placed on a tree network embedded in the plane. Then, we show that the structure of an optimal schedule for the mule is to follow (repeatedly) a simple cycle (a doubling of a subtree). Again we assume that all sensors have the same capacity c and the same rate, 1, of data accumulation. We also assume that the input is integral, i.e., c is an integer and the distance between any two sensors on the tree network is an integer.

Theorem 6. *Let there be n sensors, $p_1, p_2, \ldots, p_n$ on a tree G. For all p_i, $1 \leq i \leq n$, let $c_i = c$ and $r_i = 1$, i.e. let the capacity and rates of all sensors be the same. There exists an optimal path that minimizes data loss with the following properties: (1) it only changes direction at sensors, (2) it is a cycle obtained by doubling a subtree.*

Proof. For the first claim, we argue that any path that contains a U-turn that does not coincide with a sensor can be shortened to the preceding sensor. The shortened path will visit all nodes more frequently and will not lose any more data than the original path.

Our argument for the second claim is similar to the proof for the line case. Let there be an optimal path P that is not a cyclic, depth-first traversal on a subtree G' of G. We denote the set of nodes $X = \{x_1, x_2, \ldots, x_m\}$ to be leaves of G'. Therefore, the path P must contain subcycles that do not visit any node in X.

Let the path P_i denote a section of P that begins at x_i and ends at earliest visit to any other node in X. Without loss of generality, let P_i contain a subcycle. We denote the node z_b to be the closest node in P_i to x_i that is also part of a subcycle.

Since P is periodic, we can assume that in one full period, P always starts at x_i and ends at x_i. Beginning at x_i, the mule travels past z_b along P_i and returns back to z_b without visiting any nodes in X. Let this path be denoted as S.

Now, we will modify the trip P by removing S. Suppose S has duration δ and the path after this subcycle is removed, denoted by P', has duration T. That means, the trip P has duration $T + \delta$. We now calculate the overflow rate of one entire cycle with and without the subcycle S.

Let Y denote the set of sensors that overflow in both P and P' but are reached δ time sooner in P'. Also, let $|Y| = n'$ and $R = P \setminus Y$. Note that $S \subseteq R$.

Let m denote the amount of overflow in P' in one full period not including the overflow of the n'' nodes that P and P' miss altogether. The overflow rate is $m/T + n''$ in path P' and $\frac{m+\delta \cdot n' + \epsilon}{T+\delta} + n''$ in path P where ϵ is any extra overflow in P that is not accounted for. Since P is an optimal path, then $\frac{m+\delta n' + \epsilon}{T+\delta} + n'' \leq m/T + n''$. We simplify this relationship to

$$n' + \frac{\epsilon}{\delta} \leq \frac{m}{T}. \tag{2}$$

Consider a cyclic, depth-first traversal P'' on the subtree R. If $R \setminus S \neq \emptyset$, then there is no overflow within the subtree since there exists a path beginning at x_i that reaches all nodes in R without overflowing. Thus the overflow rate of the path on R is $n' + n''$. If $R = S$ then the overflow rate of the path is $n' + \frac{\epsilon'}{\delta} + n''$ where ϵ' is the extra overflow. Note that by definition, $\epsilon' \leq \epsilon$ since P'' is shorter than or has the same duration as S. For either case, by the overflow rate is bound by $n' + \frac{\epsilon'}{\delta} + n''$. By Equation 2,

$$n' + \frac{\epsilon'}{\delta} + n'' \leq \frac{m}{T} + n''$$

$$\frac{T(n' + \frac{\epsilon'}{\delta})}{T+\delta} + \frac{\delta(n' + \frac{\epsilon'}{\delta})}{T+\delta} \leq \frac{m}{T+\delta} + \frac{\delta(n' + \frac{\epsilon}{\delta})}{T+\delta}$$

$$n' + \frac{\epsilon'}{\delta} + n'' \leq \frac{m + \delta n' + \epsilon}{T+\delta} + n''$$

Since P is optimal, the direct path on R must be optimal as well.

Therefore, there exists an optimal path that is a cyclic, depth-first traversal on a subtree of G. □

A consequence of Thereom 6 is that we can compute an optimal mule route (we can identify an optimal subtree of G) in time that is pseudo-polynomial, using a dynamic programming algorithm.

It is unlikely that there is a strongly polynomial time algorithm for an exact solution, since we show that the problem is weakly NP-hard.

3.3.2 Hardness

We show that single mule scheduling on a tree is weakly NP-hard. Further, we show that the data gathering problem for a single mule and sensors in Euclidean (or any metric) space is NP-hard.

Theorem 7. *The data gathering problem scheduling a single mule among uniform capacity sensors on a tree is (weakly) NP-hard.*

Proof. Our reduction is from PARTITION (or SUBSET-SUM): given a set $S = \{x_1, \ldots, x_n\}$ of n integers, does there exist a subset, $S' \subset S$, such that $\sum_{x_i \in S'} x_i = M/2$, where $M = \sum_i x_i$? Given an instance of PARTITION, we construct a tree as follows: There is a node v connected to a node u by an edge of length $M/2$. Incident on v are n additional edges, of lengths x_i; the edge of length x_i leads to a node where there are exactly x_i sensors placed. Also, at node u there are M^2 sensors placed. (If one disallows multiple ($x > 1$) sensors to be at a single node w of the tree, we can add x very short (length $\Theta(1/n)$) edges incident to w, each leading to a leaf with a single sensor.) Consider the problem of computing a maximum data-rate tour in this tree, assuming each sensor has capacity $2M$. Then, in order to decide if it is possible to achieve data collection rate of $M^2 + M/2$ we need to decide if it is possible to find a subtree that includes node u (where the large number, M^2, of sensors lie) and a subset of nodes having x_i sensors each, with the sum of these x_i's totalling exactly $M/2$. (If the sum is any less than $M/2$, we fail to collect enough data during the cycle of length $2M$ that is allowed before data overflow; if the sum is any more than $M/2$, we lose data to overflow at u, which cannot be compensated for by additional data collected at the x_i nodes, since M^2 is so large compared to x_i.) □

Theorem 8. *The data gathering problem scheduling a single mule among uniform capacity sensors in the Euclidean (or any metric) space is NP-hard.*

Proof. We reduce from the Hamiltonian cycle problem in a grid graph where n points are on an integer grid and an edge exists between two points if and only if they are unit distance apart. If we place a sensor at each point with capacity n, it follows that there exists a Hamiltonian cycle in this graph if and only if there exists a data gathering solution with no data loss. □

3.3.3 Approximation Algorithm

Theorem 9. *For uniform capacity sensors in fixed dimension Euclidean space, there exists a* $(1/3-\varepsilon)$*-approximation for maximizing the data gathering rate of a single mule. For general metric spaces, a* $(1/6-\varepsilon)$*-approximation exists.*

Proof. In order to achieve this, we approximate the maximum number of distinct sensors a mule can cover in time $c/2$, the amount of time for sensors to fill from empty to half capacity (it can be shown that one half capacity is the optimal choice). The result will be a path, to which we assign one mule to traverse back and forth. The data gathering rate of this solution is equal to the number of distinct sensors covered as a mule on a schedule with period t will collect exactly t units of data from each sensor. We denote R to be the maximum number of distinct sensors that can be covered by a path of length $c/2$. Note that R can be approximated to within a factor of $1+\varepsilon$ in fixed dimension Euclidean space using the PTAS for orienteering [31]. In general metric spaces, R can be approximated to within a factor of $2+\varepsilon$ [30]. Let R^* be the data gathering rate of the optimal solution. We now show that $R^* \leq 3R$.

Consider the interval of time $c/2$ in the optimal solution that has the highest data gathering rate. This is an upper bound on R^*. In this time period, we know that the number of distinct sensors visited is at most R. We also know that during this time period at most $\frac{3}{2}c$ units of data can be downloaded from any visited sensor (at most c units of data immediately downloaded and at most $c/2$ units of data downloaded after $c/2$ units of time have passed). Therefore, the total amount of data collected in the optimal solution during this period of time is at most $\frac{3}{2}cR$. Averaging the data collected over the time interval $c/2$, the data gathering rate of the optimal solution is at most $3R$. □

3.4 k-Mule scheduling

Given a budget of k data mules, we now consider the problem of maximizing the total data gathering rate of these mules. We assume the n sensors have uniform capacity, unit data rate, and unit velocity. It is important to note that even with sensors on a line, the optimal solution may not assign mules to private tours; sensors may need to be visited by multiple mules. Consider an input with two mules and sensors uniformly spaced $c/4$ apart from one another. Any time a mule makes a U-turn, it will gather only $c/2$ data from the next sensor it visits. In order to maximize the frequency of full buckets collected, we want to minimize the frequency of U-turns made. This can be done by maintaining separation of length c between the mules and having the mules zig-zag across (nearly) the entire line (see Figure 6). Interestingly, this example also shows that mules can travel arbitrarily far distances.

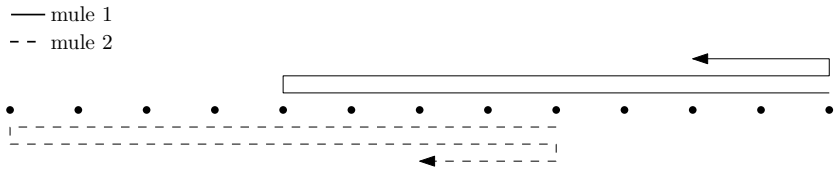


Figure 6: With two data mules and sensors uniformly spaced $c/4$ apart, many sensors will be visited by both mules in the optimal solution.

3.4.1 Sensors on a line

Theorem 10. *Given a budget of k data mules, for uniform capacity sensors on a line, there exists a 1/3-approximation for maximizing the data gathering rate.*

Proof. Similar to the case when $k = 1$, we find the maximum amount of distinct sensors that k mules can cover in time $c/2$ (it can be shown that half capacity is the optimal choice). The result will correspond to a set of disjoint intervals; we assign one mule to each interval. The duration of a cycle for each mule is the length of time a sensor fills up to capacity so no sensor is allowed to overflow. Therefore, the data gathering rate of this solution, call it R, is equal to the number of sensors covered. Note that R, the maximum amount of distinct sensors that can be covered by k disjoint intervals of length at most $c/2$, can be computed exactly in polynomial time using dynamic programming. Let R^* be the data gathering rate of the optimal solution. It

follows from the same argument given for the $k = 1$ case (Theorem 9) that $R^* \leq 3R$. □

3.4.2 Sensors in a general metric space

Theorem 11. *Given a budget of k data mules, for uniform capacity sensors in a general metric space, there exists a $\frac{1}{3}(1-\frac{1}{e^\beta})$-approximation with $\beta = \frac{1}{2+\varepsilon}$ for maximizing the data gathering rate. In fixed dimension Euclidean space there exists a $\frac{1}{3}(1-\frac{1}{e^\beta})$-approximation with $\beta = 1-\varepsilon$.*

Proof. The proof is similar to the proof of Theorem 9. In order to approximate the maximum amount of distinct sensors that k mules can cover in $c/2$ time, we compute an orienteering path with a travel distance budget of $c/2$ on the uncovered sensors. We repeat this operation for a total of k times. In the Maximum Coverage problem, one is given a universe of elements, a collection of subsets, and an integer k. The objective is to maximize the number of elements covered using k subsets. It has been shown by Hochbaum et al. [57] that greedily choosing the set with the largest number of uncovered elements k times yields a $(1-\frac{1}{e})$-approximation. Interestingly, Hochbaum et al. also show that using a β-approximation for covering the maximum amount of uncovered elements in each of the k rounds yields a $(1-\frac{1}{e^\beta})$-approximation. Computing orienteering k times on only the remaining uncovered sensors, we achieve a $\frac{1}{(2+\varepsilon)}$-approximation each round and therefore a $(1-\frac{1}{e^\beta})$-approximation for $\beta = \frac{1}{2+\varepsilon}$ for covering the maximum amount of sensors with k mules. Using similar arguments as the case where $k = 1$ (Theorem 9), it is now easy to see that having mules traverse the k orienteering paths back and forth yields a $\frac{1}{3}(1-\frac{1}{e^\beta})$-approximation. □

3.5 No Data Loss Scheduling

In situations in which it is not possible for a fixed number of data mules to collect all data in the network, it is natural to increase the number of data mules and let them collectively finish the data collection task. In the no data loss scheduling problem, we seek to minimize the number of mules in order to avoid data loss. Throughout this section we assume that all sensors have unit data rate, unit velocity, and uniform capacity.

3.5.1 Exact Algorithm on a Line

When sensors all lie along a line, we show that the problem can be solved in polynomial time. As before, we can assume that the sensors lie at integer coordinates so that, by the same argument as in Lemma 9, the mules in an optimal solution follow periodic schedules.

Lemma 10. *For the minimum cardinality data mule problem with no data loss, if the sensors have uniform capacity and lie on a line, there is an optimal schedule in which all mules follow periodic cycles, zigzaging within disjoint intervals, each with length at most $c/2$.*

Proof. The proof is by induction on the number of sensors. If there is only one sensor, then one mule is enough and it remains stationary at the sensor; this is a trivial zigzag schedule.

Suppose we have n sensors and consider the schedule of one optimal solution using k^* mules. The mule i follows along a periodic schedule C_i visiting a set of sensors S_i. Without loss of generality we can assume the mules in this optimal schedule do not visit any point to the left of the leftmost sensor. Consider the leftmost sensor, p_1, and the mules that collect data from it. If p_1 is visited by more than one mule, we will find another optimal schedule in which p_1 is visited by only one mule. To see this, we first fix a time t_1 when p_1 is visited by some mule, denoted as the first mule m_1. Suppose the next visit to p_1 is by a different mule, say m_2, at time t_2. Clearly, at the time when m_1 visits p_1, m_2 is to the right of p_1. Later, at the time when m_2 is visiting p_2, m_1 is to the right of p_1. By continuity of motion, there must be a time in $[t_1, t_2]$ such that m_1 and m_2 meet. At the intersection we can swap the motion plan of m_1 and m_2. So m_1 turns back to visit p_1 and m_2 turns back and follow the original motion plan of m_1. This modification does not change the data rate, so the modified schedule collects all data and remains optimal. Similarly, if p_1 is later visited by another mule m_3, we perform the same swap. This way p_1 is exclusively visited by one mule, m_1.

Since p_1 is only visited by a single mule, m_1, the schedule C_1 does not visit any node that is more than distance $1/2$ away from p_1 to the right (otherwise m_1 does not have enough time to go back to p_1 before its bucket becomes full). Thus we can assume without loss of generality that C_1 is a simple zig-zag tour on an interval I_1 of length $c/2$ with p_1 as the left endpoint. All sensors in this interval can be collected by this mule without data loss. Thus, we can modify the schedule for all other mules such that they do not

need to visit any nodes in I_1. Clearly, $k^* - 1$ mules are sufficient to collect data from the remaining sensors with no data loss. Since there are at most $n - 1$ sensors to be covered, by the induction hypothesis there is a schedule of $k^* - 1$ mules covering disjoint intervals. Together with the first mule, this is the disjoint schedule whose existence is claimed. □

By the above structural lemma, we can use a simple greedy algorithm to minimize the number of data mules necessary to collect data, without loss, for sensors on a line: Starting at the leftmost sensor, schedule a mule to zigzag within an interval of length $c/2$ whose left endpoint is the leftmost sensor, and then continue to the right, adding further intervals of length $c/2$ until all sensors are covered. This is an $O(n)$ algorithm for n (sorted) sensors.

3.5.2 Hardness

Theorem 12. *For uniform capacity sensors in the Euclidean plane (or a general metric space), the problem of minimum cardinality data mule scheduling with no data loss is (strongly) NP-hard.*

Proof. Similar to the reduction for Theorem 8, the reduction is from Hamiltonian cycle in a grid graph. We place a sensor with capacity n at each of the n nodes in the graph and observe that one mule suffices to cover all sensors with no data loss if and only if a Hamiltonian cycle exists in the graph. □

3.5.3 Approximation Algorithm

In the following we describe an algorithm achieving a constant-factor approximation.

It is tempting to think that an optimal solution will allocate each mule to cover an exclusive set of sensors S', that are not covered by other mules. We denote such a set of tours as a *private tour set* on S'. However, the following example shows that this is no longer the case when sensors lie in the plane. Consider $n > 2$ sensors placed on a circle, uniformly spaced with adjacent sensors at (Euclidean) distance exactly $c - 1/n$. The convex hull of these sensors is a regular n-gon of perimeter $n(c - 1/n) = nc - 1$. The optimal solution would use $n - 1$ mules, with each mule touring periodically at constant speed (1) along the boundary of this n-gon, with time/distance separation of exactly c between consecutive mules. This ensures that each sensor is visited exactly when its storage (bucket) becomes full. However,

any solution using private tours will have to use n mules, since no mule can use a private tour to cover two or more sensors (since it would have length at least $2c - 2/n > c$).

While it may be that no private tour set is optimal, we now argue that the optimal schedule using only private tours is provably close to optimal (in terms of minimizing the number of data mules). Denote by k^* the minimum number of cycles, each of length at most c, to cover all nodes, which is denoted as a *light cycle cover*. And denote by m^* the minimum number of data mules required to collect all data.

Lemma 11. $m^* \leq k^* \leq 2m^*$.

Proof. First, note that using k^* mules, each traversing a (private) light cycle, results in all data being collected; thus, $m^* \leq k^*$.

Now consider an optimal schedule of m^* data mules. Mule i moves along a schedule C_i. Consider any particular time t. Each sensor j is visited by at least one mule. We assign it to the mule that visits it first, i.e., at the earliest time after t. We know this time is at most c, since no data is lost at sensor j. Thus, consider mule i, at current position p_i (at time t) and all the sensors along C_i that are assigned to it. They all lie on a path (along C_i) of length at most c. Let s_i be the sensor furthest away from p_i, measuring distance along C_i. Let γ_i be the corresponding path along C_i, from p_i to s_i. Let b_i be the midpoint of this path. Place a clone of mule i at point s_i, and create two private cycles for mule i and his clone: one cycle goes from p_i to b_i along γ_i, then returns to p_i directly (along a shortest path or a straight segment), the other goes from b_i to s_i along γ_i, then returns to b_i directly. Mule i traverses the first cycle; his clone traverses the second cycle. Do this for all mules.

We have doubled (via cloning) the number of mules, but now each mule/clone has a private cycle, of sensors assigned only to it, and these cycles are each of total length at most c. Thus, this is a valid solution to the light cycle cover problem. Thus, the minimum number of light cycles, k^* is no greater than $2m^*$. □

By the above lemma, an α-approximation for the minimum light cycle cover gives a 2α-approximation for the minimum number of data mules. Arkin et al. [13] gave a 6-approximation algorithm for the minimum light cycle cover problem; thus, we have a 12-approximation for minimum data mule scheduling. This is summarized in the following theorem.

Theorem 13. *For uniform capacity sensors within a general metric space or in the Euclidean plane, computing the minimum number of data mules to collect all data is NP-hard. There is a polynomial-time* 12*-approximation algorithm for sensors in a general metric space.*

3.6 Different Capacities

We now consider both the k-mule scheduling problem and the no data loss scheduling problem on n sensors with potentially different sensor storage capacities. Each sensor has unit data rate. The result for the k-mule scheduling problem obviously holds for the single mule problem (i.e. when $k = 1$).

3.6.1 k-Mule Scheduling

Lemma 12. *With m groups of sensors, each group having the same storage capacity, optimally solving each group independently and taking the solution with the highest data gathering rate yields a $O(1/m)$-approximation to the k-mule scheduling problem.*

Proof. Let $r(\cdot)$ be the data gathering rate of a solution. Let OPT_i be the optimal solution to group i and let OPT be the schedule with highest data rate. $r(OPT) \leq \sum_{i=1}^{m} r(OPT_i) \leq m \cdot \max_i\{r(OPT_i)\}$. The first inequality is from the following observation. Consider the optimal schedule OPT and modify it such that we only visit the nodes in group i. This is obviously a solution for collecting data from group i and thus has data rate no greater than $r(OPT_i)$. □

Let c_{max} and c_{min} be the storage capacities of the largest and smallest sensors respectively. We round the storage capacity of each sensor down to its nearest power of two. Doing so, we create m groups of sensors where m is at most $\log(\frac{c_{max}}{c_{min}})$. Note that m may be significantly smaller than $\log(\frac{c_{max}}{c_{min}})$. In the rounding down process, the storage capacity of each sensor is at most halved, thus the optimal solution on the new sensors has data gathering rate of at least 1/2 of the same solution before rounding. We approximate the optimal solution to each of the groups within a constant factor and choose the one with highest data gathering rate. By Lemma 12, we have the following.

Theorem 14. *By rounding down the sensor capacities into $m \leq \log(\frac{c_{max}}{c_{min}})$ groups, the group with highest data gathering rate has rate at least $O(1/m) \cdot r(OPT)$ where OPT is the optimal solution to the k-mule scheduling problem.*

3.6.2 No Data Loss Scheduling

Theorem 15. *By rounding down the sensor capacities into $m \leq \log(\frac{c_{max}}{c_{min}})$ groups and solving each group independently, at most $O(m) \cdot |OPT|$ mules are used in total, where $|OPT|$ is the minimum number of mules needed to avoid data loss.*

Proof. Using the same rounding technique as the previous section, we again obtain m groups of sensors with $m \leq \log(\frac{c_{max}}{c_{min}})$. In the rounding down process, the capacity of any sensor is at most halved. Thus, the optimal solution on the rounded down sensors requires at most two times the number of mules as the optimal solution to the original set of sensors. Let $|OPT_i|$ be the minimum number of mules needed for no data loss to occur in group i and let $|OPT|$ be the number of mules in the optimal solution. Since $|OPT| \geq |OPT_i|$ for $1 \leq i \leq m$, we have that $m \cdot |OPT| \geq \sum_{i=1}^{m} |OPT_i|$. Approximating $|OPT_i|$ within a constant factor for all i, we use $O(m) \cdot |OPT|$ mules. □

3.7 Practical Algorithms

Our discussion on mule scheduling algorithms have focused on getting the best theoretical approximations. Although some of the algorithms (especially the one for the case of a line) are practical and implementable, some of the other algorithms mentioned depend on solving the orienteering problem or k-TSP problem. These algorithms are too complicated for practical implementation. In this section we try to find heuristic algorithms that are easily implementable. We also discuss upper and lower bounds that can be used to estimate the optimal solution and provide comparisons for the heuristic algorithms. These are tested in our simulations.

3.7.1 Single Mule Scheduling

3.7.1.1 Lower Bounds

For maximizing data rate by a single mule, any algorithm giving a feasible solution will be a lower bound. We run approximation algorithms of TSP on the entire point set, then traverse the TSP in one direction, searching all windows of variable length w in the tour. Each window starts at a distinct point p in the TSP and is pinned to the point that is farthest away from p

along the TSP and still within length w. Thus, these windows will always be pinned to two points in the TSP. We can then traverse these interval back and forth and determine the data gathering rate. For example, suppose $w = 1$ and suppose this interval covers r nodes, then cycling through the r nodes gives data rate of at least $r/2$. We will refer to this algorithm as 'shifted window along TSP'.

3.7.1.2 Upper Bounds

To get an upper bound on the highest data rate by a single mule, we start from the optimal schedule to discuss how to estimate it from above.

Let S be a set of nodes in the plane and let P be the optimal schedule with data rate of R. If the length of P is greater than one, then there must be a unit length interval along P such that the data gathering rate is at least R, by pigeon hole principle. Now we take such a unit length interval and denote it P'. Suppose the number of nodes covered (with repeats) by P' is h and the set of nodes is H. $h \geq |H|$. Note that in a unit length interval, a node can be considered to be visited at most twice because if a node is visited three times, the second visit can always be skipped and no data will be lost. Thus each node in H is visited at most twice in P'. $h \leq 2|H|$. In addition, $R \leq h$ since we collect at most a full bucket of data at each node. Thus we have an upper bound $2|H|$ on the optimal data rate. Since we do not know the optimal schedule, we do not know H either and this upper bound is not immediately useful. In the following we do further relaxation and find an upper bound on $|H|$.

Recall that the set of nodes in H can be connected by a path of length at most 1. Therefore if we can find the maximum number of nodes visited by a path of length at most 1, then we have an upper bound on $|H|$. This is an instance of an orienteering problem and is still hard to solve. But we know that this set H is fully contained in a square of unit side length. The distance between the leftmost and rightmost point in H is at most 1. The same can be said in the vertical direction. Thus, if we lay out a grid over our point set with cell side length of $1/2$, we know that there exists a 3×3 square in the grid that contains any unit square, and therefore we can find such a square that contains H.

We enumerate all 3×3 squares in the grid and on each square we run Kruskal's algorithm until the sum of the edge weights exceeds 1. We choose the 3×3 square that maximizes the number of points picked up. Call this

point set K. By the aforementioned argument, we know that $|K| \geq |H|$. Since points in H can be repeated at most twice in the optimal schedule, we conclude that the optimal data gathering rate is at most $2|K|$.

3.7.2 No Data Loss Scheduling

By Lemma 11, a solution to the minimum light cycle cover is a valid solution to our minimum cardinality mule scheduling problem with no data loss and is a natural upper bound. Further, if we have a lower bound L on the minimum light cycle cover problem, $L \leq k \leq 2k^*$ then $L/2$ is a lower bound on the number of mules needed. Thus in the following we discuss upper and lower bounds on the minimum light cycle cover problem.

Besides the algorithm by Arkin et al. [13], The following algorithm provides a solution (upper bound) on the optimal number of data mules. Take the minimum spanning tree of the set of sensors and remove all edges of length 1 or more. This leaves a number of connected components $S_1, S_2, \cdots, S_k$. For each connected component S_i, we can duplicate each edge to form a Euler tour. Remove duplicate occurrences of any node on this tour and we get a cycle C_i to visit the nodes in S_i. If C_i has length ℓ_i, we use $\lceil \ell_i \rceil$ mules along C_i with uniform timewise separation. This algorithm is called 'chopped MST'.

3.8 Simulations

In this section we show the results of implementations of the upper and lower bounds that were discussed in the previous section.

3.8.1 Single Mule Simulations

We generated one hundred different point sets. For each instance we generated 500 uniformly distributed points in a 5×5 square and computed shifted window along TSP with window lengths between 0.5 and 1.0 in increments of 0.05. We compare the results to the grid based upper bound. Figure 7 shows the results. With window length 0.7, the worst ratio over all point sets was roughly 0.18353. i.e. the data gathering rate of a mule following our heuristic was 0.18353 times as fast as the rate produced by the grid based upper bound. In order to compare to a sparser point set, we ran the same experiment in a 10×10 square (see Figure 8) and noticed improved results.

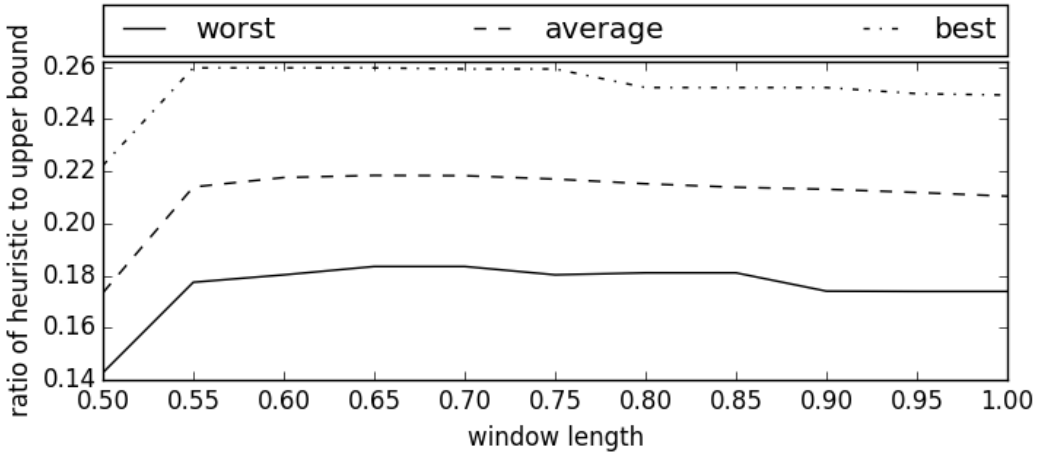


Figure 7: Shifted window along TSP of uniformly distributed point sets in a 5×5 square.

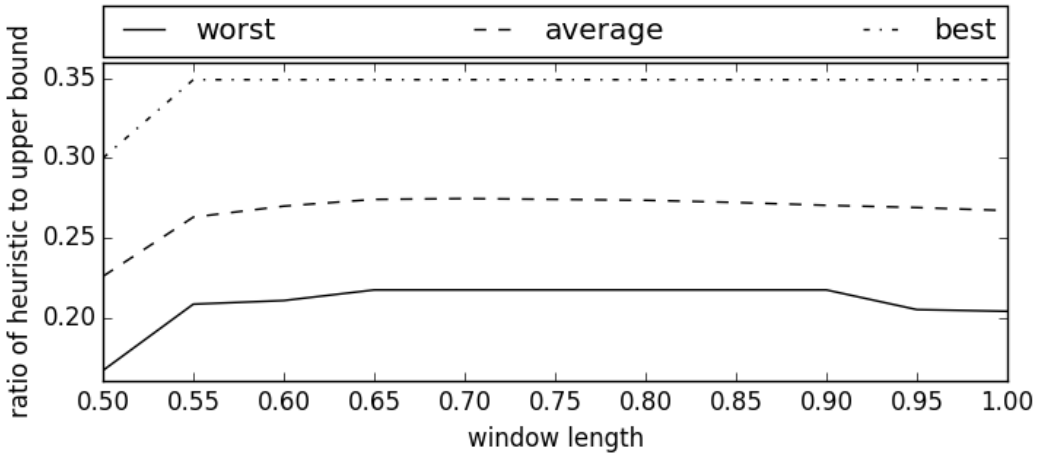


Figure 8: Shifted window along TSP of uniformly distributed point sets in a 10×10 square.

We also computed shifted window along TSP on the 4,663 cities of Canada [1] (see Figure 9) with varying window lengths. The point set was scaled down so that the average distance between any two points is 1.

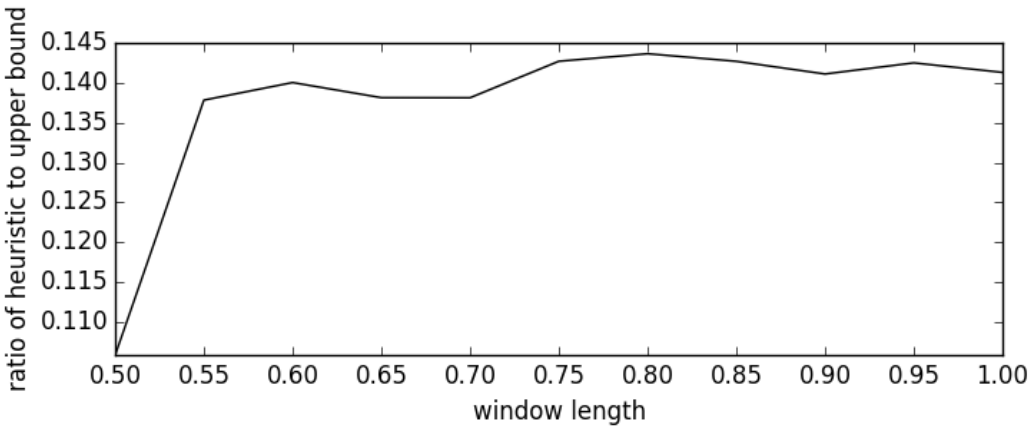


Figure 9: Shifted window along TSP of 4,663 cities of Canada.

3.8.2 No Data Loss Simulations

First, we generated one hundred point sets, each set containing 200 points. For each set, we generated 10 uniformly distributed points in a 5×5 square. Centered at each of these points we generated 20 uniformly distributed points in a 1×1 square and discarded the center point. Over all point sets, we compared the number of mules used in a chopped MST solution to the number of mules used in a light cycles solution. The results can be seen in table 3. In addition to the results in this table, we found that the point set with highest ratio of number of mules used in a light cycles solution to number of mules in a chopped MST solution was 1.46154. The lowest ratio was 1.18868. We repeated this experiment for a 10×10 square (see table 4). The highest and lowest ratios were 1.45652 and 1.16949 respectively.

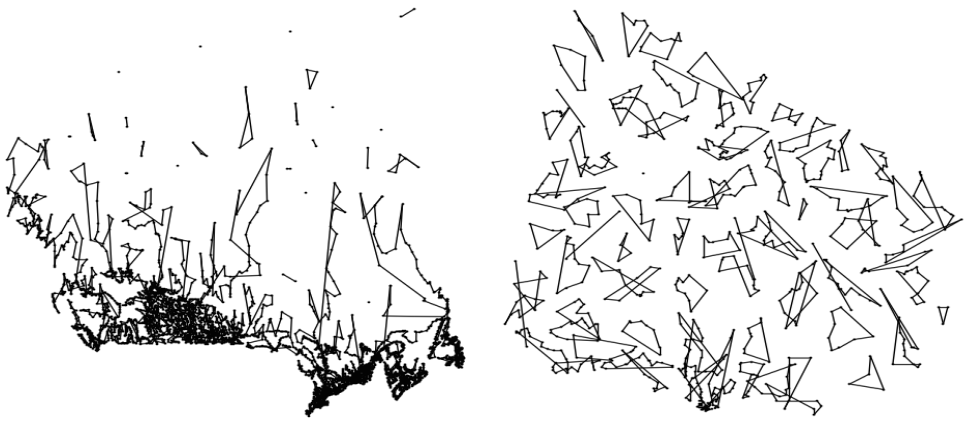
We computed chopped MST on the cities in Canada (fig 10). The points were scaled down so that the average distance between any two points is 10. 602 mules are enough to cover these points. Note that there are edge crossings in this solution because we used a two approximation to compute TSP.

	Chopped MST	Light Cycles
Best	39	53
Worst	55	69
Average	47.45	62.75

Table 3: Number of mules used in chopped MST vs. light cycles in 5×5 square.

	Chopped MST	Light Cycles
Best	45	62
Worst	59	75
Average	52.04	67.58

Table 4: Number of mules used in chopped MST vs. light cycles in 10×10 square.



(i) (ii)

Figure 10: (i) Chopped MST on 4,663 cities of Canada. 602 mules used; (ii) Approximation of minimum light cycles on 734 cities of Uruguay. 85 mules used.

We also tested the approximation algorithm to minimum light cycles. This algorithm runs significantly slower than chopped MST. Therefore, we used the 734 cities of Uruguay [1] (fig 10). The point set was scaled down so that the average distance between any two points is 1. 85 mules are enough to cover these points. For comparison, chopped MST produces a TSP on the entire point set and uses 50 mules. We also scaled the Uruguay point set so that the average distance between any two points is 10. This time the approximation to minimum light cycles outperformed chopped MST using 436 and 470 mules respectively.

Chapter 4

Choice is Hard[‡]

4.1 Introduction

A multiple choice problem consists of a set of color classes $P = \{C_1, C_2, \ldots, C_n\}$, where each color class C_i consists of a pair of objects. When the underlying objects are points (resp., intervals) on the x-axis, we say that P is a set of point (resp., interval) color classes. Consider a set P of point color classes. We call an interval on the x-axis that contains at most one point from each color class a *conflict-free* interval (or CF-interval for short). Given a set of color classes P and a set $Q \subseteq \cup_{i=1}^{n} C_i$, we say that Q is a *rainbow* if it contains at most one object from each color class. The first problem that we study (rainbow minmax gap) is mentioned in a recent paper by Consuegra and Narasimhan [37].

Rainbow minmax gap (decision version): Given a set P of n point color classes and a value $d > 0$, determine whether there exists a rainbow Q of size n with *max gap* at most d, where the max gap of Q is the maximum distance between a pair of consecutive points in Q.
This problem is the 1-dimensional version of a more general 2-dimensional problem. Consider a set of agents (represented by points in the plane) where each agent provides a certain service, and for each of these services, there are several agents in the set providing this service. The goal is to compute a minimum bottleneck spanning tree consisting of exactly one agent for each of the available services. In [37], the authors present a 2-approximation algorithm for rainbow minmax gap, but leave the question whether the problem is NP-hard or not open. In Section 4.3 we prove that the problem is NP-hard.

In order to obtain this result we define a new and especially simple satisfiability problem, which we call *linear SAT* (or LSAT for short), and prove that it is still NP-complete. A 3-SAT formula is an LSAT formula if each clause (viewed as a set of literals) intersects at most one other clause, and, moreover, if two clauses intersect, then they have exactly one literal in common.

[‡]This chapter is based on joint work with Esther M. Arkin, Aritra Banik, Paz Carmi, Matthew J. Katz, Joseph S. B. Mitchell and Marina Simakov. The work in this chapter appeared in *ISAAC 2015* [8].

An LSAT formula can be depicted as a set of disjoint semi-closed intervals on a line, see Figure 11. We prove that the problem of deciding whether an LSAT formula is satisfiable or not is NP-complete. This is quite surprising, since the satisfiability problem for the class of formulas that can be depicted as disjoint closed intervals on a line is already polynomially solvable. We believe that the NP-completeness of LSAT may be useful in deriving other hardness results. In particular, we use LSAT to prove NP-hardness of the following two multiple choice problems, see Section 4.3.

Rainbow piercing: Given a set P of point color classes and a set of intervals $\mathcal{I}$ on the x-axis, determine whether there exists a rainbow Q that is a *piercing set* for $\mathcal{I}$ (i.e., each interval in $\mathcal{I}$ is pierced by at least one point in Q.)

Rainbow covering: Given a set P of interval color classes, i.e., where each color class C_i is a pair of intervals on the x-axis, and a set of points S on the x-axis, determine whether there exists a rainbow Q that *covers* S (i.e., each point in S is covered by at least one interval in Q).

A fascinating related problem is: cover exactly one point from each color class using a minimum number of (arbitrary) intervals. This problem is motivated by the following problem. Consider a sensor network where two sensors (points) have the same color if they generate the same data. The goal is to collect data from each color class of sensors. Covering a set of points with an interval is equivalent to using one transceiver to retrieve data from a set of sensors. In order to prevent all data from a color class from being corrupted by a malicious attack, we only allow one sensor from each color class to interact with a transceiver; the other sensors act as backups.

Covering color classes with intervals of arbitrary length: Given a set P of point color classes, find a minimum-cardinality set $\mathcal{I}$ of intervals of arbitrary length, such that exactly one point from each color class is covered by an interval in $\mathcal{I}$.
In Section 4.4 we show that this problem is NP-hard, by first showing that the following simpler problem is NP-hard.

Covering color classes with unit length intervals: Given a set P of point color classes, decide whether or not there exists a set of unit length intervals, $\mathcal{I}$, such that exactly one point from each color class is covered. Assuming a feasible solution exists, minimize the cardinality of $\mathcal{I}$.

Related work. As far as we know, the first to consider a "multiple-choice" problem of this kind were Gabow et al. [46], who studied the following problem. Given a directed acyclic graph with two distinguished vertices s and t and a set of k pairs of vertices, determine whether there exists a path from s to t that uses at most one vertex from each of the given pairs. They showed that the problem is NP-complete. A sample of additional graph problems of this kind can be found in [11, 58, 86]. The first to consider a problem of this kind in a geometric setting were Arkin and Hassin [12], who studied the following problem. Given a set V and a collection of subsets of V, find a cover of minimum diameter, where a cover is a subset of V containing at least one representative from each subset. They also considered the multiple-choice dispersion problem, which asks one to maximize the minimum distance between any pair of elements in the cover. They proved that both problems are NP-hard. Recently, Arkin et al. [10] considered the following problem. Given a set S of n pairs of points in the plane, color the points in each pair by red and blue, so as to optimize the radii of the minimum enclosing disk of the red points and the minimum enclosing disk of the blue points. In particular, they consider the problems of minimizing the maximum and minimizing the sum of the two radii. In another recent paper, Consuegra and Narasimhan [37] consider several problems of this kind, including the rainbow minmax gap problem, for which they present a 2-approximation algorithm (and we prove NP-hardness).

4.2 A New Satisfiability Result

In the boolean satisfiability problem (SAT), one is given a formula in conjunctive normal form and the goal is to determine whether it is satisfiable or not. SAT is one of the first problems that was shown to be NP-complete (by Cook [38]). Subsequently, many variants of SAT were shown to be NP-complete, including the variant known as 3-SAT, in which each clause consists of at most three literals [25, 67, 87]. Some restricted variants of SAT can be solved in polynomial time [15, 41, 68]. In this subsection we define an especially simple variant of 3-SAT, which we call *linear SAT* (LSAT for short), and prove that it is NP-complete. A 3-SAT formula is an LSAT formula if each clause (viewed as a set of literals) intersects at most one other clause. Moreover, if two clauses intersect, then they have exactly one literal in common. Let F be an LSAT formula and let T be its corresponding set of literals,

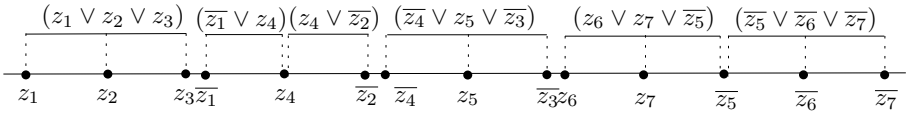


Figure 11: An example of an LSAT formula.

then F can be depicted as in Figure 11. That is, one can sort the literals in T, such that (i) each clause of F corresponds to at most three consecutive literals in the sorted list, and (ii) each clause shares at most one of its literals with another clause, in which case this literal is extreme in both clauses.

Observe that if the clauses of a 3-SAT formula F are pairwise disjoint, then one can determine in polynomial time whether F is satisfiable or not, by determining whether the corresponding bipartite graph in which there is an edge between clause C and variable x if and only if either x or $\overline{x}$ appear in C contains a perfect matching. It is therefore somewhat surprising that LSAT is NP-complete, since the clauses of an LSAT formula are almost pairwise disjoint. We now prove that LSAT is NP-complete by a reduction from 3,4-SAT. A 3-SAT formula is a 3,4-SAT formula if each variable appears in at most 4 clauses, either negated or unnegated. 3,4-SAT was shown to be NP-complete by Tovey [87].

Let F be a 3,4-SAT formula, let X be the underlying set of variables, and let $\mathcal{C}$ be the set of clauses of F. Without loss of generality, we assume that each variable $x_i \in X$ appears unnegated (i.e., as x_i) in at most three clauses and negated (i.e., as $\overline{x_i}$) in at most two clauses. We construct an LSAT formula $F_L = (X_L, \mathcal{C}_L)$ from F, and show that there is a truth assignment for X such that each clause in $\mathcal{C}$ is satisfied if and only if there is a truth assignment for X_L such that each clause in $\mathcal{C}_L$ is satisfied. We construct X_L from X as follows. For each variable $x_i \in X$ we add to X_L the variables $x_i, a_i, y_{i1}, y_{i2}, y_{i3}, z_{i1}, z_{i2}$. Also, for each variable x_i we add the following clauses to $\mathcal{C}_L$:

1. $(y_{i1} \vee x_i)$ 2. $(x_i \vee a_i)$ 3. $(y_{i2} \vee \overline{a_i})$ 4. $(\overline{a_i} \vee y_{i3})$ 5. $(z_{i1} \vee \overline{x_i})$ 6. $(\overline{x_i} \vee z_{i2})$.

Observe that clause 1 and clause 2 share x_i, clause 3 and 4 share $\overline{a_i}$, and clause 5 and 6 share $\overline{x_i}$. Now, for each clause $C_i \in \mathcal{C}$, we add a clause to $\mathcal{C}_L$ as follows. For each variable x_i that appears in C_i, if x_i appears unnegated, then we replace it by $\overline{y_{i1}}$, $\overline{y_{i2}}$, or $\overline{y_{i3}}$, depending on whether this is the first, second, or third occurrence of x_i, and if x_i appears negated, we replace it by

$\overline{z_{i1}}$ or $\overline{z_{i2}}$, depending on whether this is the first or second occurrence of $\overline{x_i}$. For example, given the formula

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_4 \vee \overline{x_2}) \wedge (x_1 \vee \overline{x_4} \vee \overline{x_2}) \ ,$$

we create the following three clauses (in addition to the six clauses that are created for each of the variables x_1, x_2, x_3, x_4).

$$(\overline{y_{11}} \vee \overline{y_{21}} \vee \overline{z_{31}}), (\overline{y_{12}} \vee \overline{y_{41}} \vee \overline{z_{21}}), (\overline{y_{13}} \vee \overline{z_{41}} \vee \overline{z_{22}}) \ .$$

It is easy to see that the obtained formula, F_L, is indeed an LSAT formula, since each clause in $\mathcal{C}_L$ that was obtained from a clause in $\mathcal{C}$ by replacement does not share any of its literals with another clause in $\mathcal{C}_L$.

Theorem 16. *F is satisfiable if and only if F_L is satisfiable.*

Proof. Assume F is satisfiable, that is, there exists a truth assignment for X such that each clause in $\mathcal{C}$ is satisfied. We show that F_L is satisfiable. If $x_i \in X$ was assigned FALSE (i.e., 0), then we assign TRUE (i.e., 1) to the variables $a_i, y_{i1}, y_{i2}, y_{i3}$ and 0 to the variables x_i, z_{i1}, z_{i2}. On the other hand, if $x_i \in X$ was assigned 1, then we assign 0 to $a_i, y_{i1}, y_{i2}, y_{i3}$ and 1 to x_i, z_{i1}, z_{i2}. We claim that this truth assignment to the variables of X_L satisfies F_L. Observe first that all the $6|X|$ clauses created for the variables in X are satisfied, since each of them consists of two literals which assume opposite values. It remains to show that the assignment satisfies the clauses consisting of three literals. But this is obvious, since for any clause $C \in \mathcal{C}$ and any literal t of C, the value of t and the value of t' are equal, where t' is the literal replacing t in F_L. (That is, if $t = x_i$, then $t' = \overline{y_{ik}}$, for some $k \in \{1, 2, 3\}$, and $x_i = 1$ if and only if $\overline{y_{ik}} = 1$, and if $t = \overline{x_i}$, then $t' = \overline{z_{ik}}$, for some $k \in \{1, 2\}$, and $\overline{x_i} = 1$ if and only if $\overline{z_{ik}} = 1$.)

We now prove that if F_L is satisfiable, then so is F. Consider any truth assignment for X_L that satisfies F_L. This truth assignment (restricted to X) also satisfies F. This is true, since, as can easily be verified, $\overline{y_{ik}} = 1 \implies x_i = 1$ and $\overline{z_{ik}} = 1 \implies \overline{x_i} = 1$. (For example, if $\overline{y_{i2}} = 1$, then since $(y_{i2} \vee \overline{a_i})$ is satisfied, we deduce that $\overline{a_i} = 1$ and therefore, $a_i = 0$. But now, since $(x_i \vee a_i)$ is satisfied, we deduce that $x_i = 1$.) □

We conclude that

Theorem 17. *LSAT is NP-complete.*

4.3 Applications of LSAT to Rainbow Problems

In this subsection we prove that the rainbow problems mentioned in the introduction are NP complete, by devising reductions from LSAT. Specifically, we first prove that (the decision version) of minmax gap is NP-complete, and then we show that rainbow piercing and rainbow covering are NP-complete.

4.3.1 Rainbow minmax gap (decision version) is NP-complete

Let P be a set of n color classes, where each color class C_i is a pair of points $\{p_i, \overline{p_i}\}$ on the x-axis, and let $d > 0$. We prove that the decision version of rainbow minmax gap is NP-complete, that is, it is NP-complete to determine whether there exists a rainbow $Q \subset \cup_{i=1}^{n} C_i$ of size n, such that the maximum gap between a pair of consecutive points in Q is at most d.

We present a reduction from LSAT. Let F be an LSAT formula, and let X be the underlying set of variables and $\mathcal{B}$ be the set of clauses of F. Let k be the number of clauses in $\mathcal{B}$ that do not intersect any other clause in $\mathcal{B}$. Place the points $q_1, q_2, \ldots, q_{k+1}$ on the x-axis, from left to right, such that the distance between any two consecutive points is $d + \frac{d}{4}$. Now, for each clause B_i of these k clauses, place three additional points between q_i and q_{i+1}, one for each of its literals. For example, if $B_i = (x_a \vee x_b \vee \overline{x_c})$, then we place the points p_a, p_b, and $\overline{p_c}$ such that p_b is at the middle of the interval $\overline{q_i q_{i+1}}$ and p_a and $\overline{p_c}$ are to its left and right, respectively, at distance $\frac{d}{8}$ from p_b (see Figure 12(a)).

Next, consider the pairs of clauses that have a single literal in common, and let l be their number. Place the points $q_{k+2}, \ldots, q_{k+l+1}$, from left to right, such that the distance between q_{k+i} and q_{k+i+1} is $2d$, for $i = 1, \ldots, l$. Now, for the i'th pair B, B' of these l pairs of clauses, place five additional points between q_{k+i} and q_{k+i+1}. For example, if $B = (x_a \vee \overline{x_b} \vee x_c)$ and $B' = (x_c \vee x_d \vee \overline{x_e})$, then we place the points $p_a, \overline{p_b}, p_c, p_d, \overline{p_e}$ such that the distance between q_{k+i} and the first point (p_a), as well as the distance between q_{k+i+1} and the last point $(\overline{p_e})$, is $\frac{d}{2}$ and the distance between any two consecutive points is $\frac{d}{4}$ (see Figure 12(b)). Finally, place the points $\overline{q_1}, \ldots, \overline{q_{k+l+1}}$ such that distance between any two consecutive points, including the distance between q_{k+l+1} and $\overline{q_1}$, is $d + \epsilon$, for some $\epsilon > 0$. See Figure 13 for a complete example.

Notice that in our reduction, we have assumed that each clause in F consists of three literals. However, we can adapt the reduction to fit formulas

containing two literal clauses.

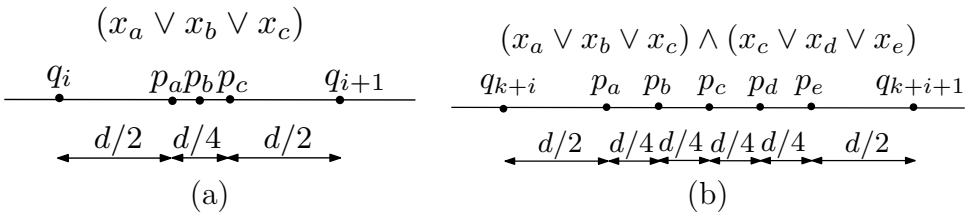


Figure 12: The reduction from LSAT to the decision version of minmax gap.

Lemma 13. *Let P be the resulting set of color classes (i.e., $P = \{\{q_1, \overline{q_1}\}, \ldots, \{q_{k+l+1}, \overline{q_{k+l+1}}\}, \{p_a, \overline{p_a}\}, \{p_b, \overline{p_b}\}, \ldots\}$). F is satisfiable if and only if there exists a rainbow Q consisting of one point from each color class, such that the maximum gap between a pair of consecutive points in Q is at most d.*

Proof. Assume F is satisfiable, and consider the rainbow Q that is obtained as follows. First, add the points $q_1, \ldots, q_{k+l+1}$ to Q. Next, for each variable x_i appearing in F, if x_i was assigned TRUE, then add the point p_i to Q; otherwise, add the point $\overline{p_i}$ to Q. Obviously, Q consists of exactly one point from each color class. We claim that the distance between any two consecutive points in Q is at most d. To see this, it is enough to examine the situation between q_i and q_{i+1}, for $i = 1, \ldots, k+l$. If $i \leq k$, then since the clause, B_i, corresponding to this interval is satisfied, one of its literals is true and the point corresponding to it was added to Q. Clearly, the distance between this point and q_i (alternatively, q_{i+1}) is at most $\frac{3d}{4}$ (see Figure 12(a)). If $k+1 \leq i \leq k+l$, then consider the pair of clauses B, B' corresponding to this interval. Since both are satisfied, then either the literal that is common to both is true, or each of them has a unique literal that is true. In the former case, Q contains the midpoint between q_i and q_{i+1}, whose distance from q_i (alternatively, q_{i+1}) is exactly d, and in the latter case, Q contains two points p_1, p_2, such that the distance between them is at most d and the distance between p_1 and q_i, as well as the distance between p_2 and q_{i+1} is at most $\frac{3d}{4}$ (see Figure 12(b)). We have shown that Q is as required.

Assume now that there exists a rainbow Q consisting of one point from each color class, such that the maximum gap between a pair of consecutive points in Q is at most d. Observe that Q cannot contain any of the points $\overline{q_1}, \ldots, \overline{q_{k+l+1}}$, so it must contain all the points $q_1, \ldots, q_{k+l+1}$. We assign

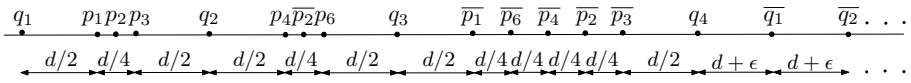


Figure 13: A complete example: $F = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee \overline{x_2} \vee x_6) \wedge (\overline{x_1} \vee \overline{x_6} \vee \overline{x_4}) \wedge (\overline{x_4} \vee \overline{x_2} \vee \overline{x_3})$.

values to the variables appearing in F is follows. If $p_i \in Q$, then set $x_i = 1$, and if $\overline{p_i} \in Q$, then set $x_i = 0$. We claim that this assignment satisfies each of the clauses of F. Consider any clause B_i that does not intersect any other clause in $\mathcal{B}$. Since the distance between q_i and q_{i+1} is greater than d, at least one of the three points corresponding to B_i's literal belongs to Q, implying that B_i is satisfied. Consider now any two clauses B, B' that have a single literal in common. In this case, the distance between q_i and q_{i+1} is $2d$. So, either Q contains the midpoint between q_i and q_{i+1} which corresponds to the common literal, implying that both clauses are satisfied, or Q contains two points, one corresponding to a literal of B and one to a literal of B', again implying that both clauses are satisfied. Thus, we have shown that F is satisfiable. □

Hence, we have proved the following theorem.

Theorem 18. *The decision version of rainbow minmax gap is NP-complete.*

Corollary 1. *Rainbow minmax gap is NP-hard.*

4.3.2 Rainbow piercing and rainbow covering are NP-complete

Let P be a set of n color classes, where each color class C_i is a pair of points on the x-axis, and let $\mathcal{I}$ be a set of intervals on the x-axis. Recall that a set of points $Q \subseteq \cup_{i=1}^{n} C_i$ is a rainbow, if Q contains at most one point from each color class. We prove that rainbow piercing is NP-complete, that is, it is NP-complete to determine whether there exists a rainbow Q such that each interval in $\mathcal{I}$ is pierced by at least one point in Q.

Theorem 19. *Rainbow piercing is NP-complete.*

Proof. Let F be an LSAT formula and let T be the set of literals in F. Recall that one can sort the literals in T, such that each clause of F corresponds to at most three consecutive literals in the sorted list. Given such a sorted list, we map each literal to a point on the x-axis, such that if literal t_1 precedes

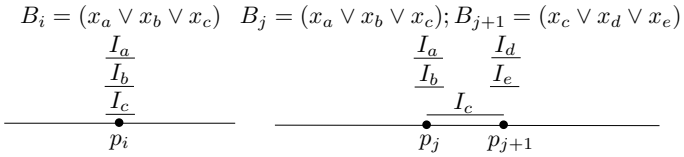


Figure 14: The reduction from LSAT to rainbow covering.

literal t_2 in the list, then the point corresponding to t_1 is to the left of the one corresponding to t_2. We also define, for each variable x_i, a color class C_i that consists of the points corresponding to x_i and to $\overline{x_i}$. Next, we map each clause B of F to the interval spanning the points corresponding to the literals in B; see Figure 11. It is easy to see that F is satisfiable if and only if there exists a rainbow Q that is a piercing set for the set of intervals. (If x_i is assigned true, then the point corresponding to x_i is added to Q else the point corresponding to $\overline{x_i}$ is added to Q; and vice versa.) Hence, the result holds. □

Let P be a set of n color classes, where each color class C_i is a pair of intervals $\{I_i, \overline{I_i}\}$ on the x-axis, and let S be a set of points on the x-axis. We prove that rainbow covering is NP-complete, that is, it is NP-complete to determine whether there exists a rainbow Q such that each point in S is covered by at least one interval in Q.

Theorem 20. *Rainbow covering is NP-complete.*

Proof. Let F be an LSAT formula, and let X be the underlying set of variables and $\mathcal{B}$ be the set of clauses of F. We create an instance of rainbow covering, and show that F is satisfiable if and only if there exists a rainbow cover. For each clause $B_i \in \mathcal{B}$, we create a point s_i. Moreover, for each literal t of B_i, we create an interval covering only s_i; if $t = x_j$, then we name the interval I_j, and if $t = \overline{x_j}$, then we name it $\overline{I_j}$. Finally, if clauses B_i and B_{i+1} share a literal, say x_j, then we replace the two intervals named I_j by a single interval I_j covering both s_i and s_{i+1} (see Figure 14). Set $P = \{C_1, C_2, \ldots\}$, where $C_i = \{I_i, \overline{I_i}\}$, and $S = \{s_1, s_2, \ldots\}$. It is easy to verify that F is satisfiable if and only if there exists a rainbow $Q \subset \cup_i C_i$ that covers S. (If x_i is assigned true, then the interval I_i is added to Q else the interval $\overline{I_i}$ is added to Q; and vice versa.) Hence the result holds. □

4.4 Exact Coverage of Color Classes

Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of n color classes, where each color class C_i is a pair of points $\{p_i, \overline{p_i}\}$ on the x-axis. We consider coverage problems where the goal is to use intervals on the x-axis to cover *exactly one* point from each color class. We now prove that the following three problems are NP-hard; the decision versions are easily seen to be in NP. Note that in each of these problems, it is implied that the intervals are conflict-free (no interval can contain two points from the same color class). In this subsection, we represent point pairs in Figures 15-19 as the tips of a $\sqcap$ shape or the tips of a $\sqcup$ shape. Certain pairs in these figures are drawn in color in order to help explain the constructions.

Problem 1 (Covering color classes with unit length intervals)**.** *Decide whether or not there exists a set of unit length intervals, $\mathcal{I}$, such that exactly one point from each color class is covered by an interval in $\mathcal{I}$.*

Problem 2 (Covering color classes with the fewest unit length intervals)**.** *Find a minimum-cardinality set $\mathcal{I}$ of unit length intervals (assuming a feasible solution exists), such that exactly one point from each color class is covered by an interval in $\mathcal{I}$.*

Problem 3 (Covering color classes with intervals of arbitrary length)**.** *Find a minimum-cardinality set $\mathcal{I}$ of intervals of arbitrary length, such that exactly one point from each color class is covered by an interval in $\mathcal{I}$.*

4.4.1 Unit intervals

Theorem 21. *Problem 1 is NP-complete.*

Proof. Problem 1 is clearly in NP because we can check whether or not exactly one point from each color class is covered in polynomial time. The reduction is from 3-SAT. Given n variables $\{x_1, x_2, x_3, \ldots, x_n\}$, and m clauses $\{c_1, c_2, c_3, \ldots, c_m\}$, we design the following gadgets.

Each clause gadget (Figure 15) consists of five points. It contains a pair of points d_i (represented by a $\sqcap$ shape in Figure 15), interleaved with three blue points; each of the three paired to a point in a variable gadget (these blue pairs are represented by a $\sqcup$ shape in Figure 15). In a clause gadget, the Euclidean distance between any two consecutive (blue) points that are paired to variables is $1 - \varepsilon$ for $\varepsilon > 0$ (ε should be bounded above; $\varepsilon < 1/3$ suffices).

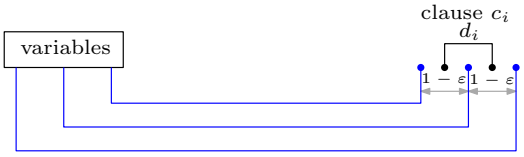


Figure 15: Clause gadget for problem 1.

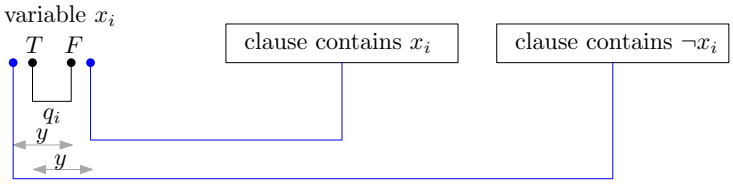


Figure 16: Variable gadget for problem 1.

Each variable gadget (Figure 16) consists of a consecutive pair of points, q_i, surrounded by blue points on each side. If variable x_i (resp. $\overline{x_i}$) appears in clause c_i, then one blue point will be placed to the right (resp. left) of q_i and this point will be paired to a blue point in c_i. The blue points that surround q_i are placed a distance of y from their respective farthest points in q_i. We set $y < 1$, ensuring that any unit interval that covers a point in q_i must also cover either the surrounding blue points to the left or right of q_i. Setting x_i to FALSE is equivalent to covering the right point of q_i. Setting x_i to TRUE is equivalent to covering the left point of q_i. We line up all of the variables, followed by all of the clauses, so that each consecutive gadget is spaced farther than unit distance apart.

If a clause evaluates to FALSE, each of the three blue points in a clause cannot be covered. Pair d_i will now be left uncovered because we cannot cover a point in d_i with a unit interval without covering one of the blue points in the clause. If a clause evaluates to TRUE, then a point from d_i can always be covered. Therefore, there exists a satisfying truth assignment in 3-SAT if and only if there exists a covering with unit length intervals such that exactly one point from each color class is covered. □

Theorem 22. *Problem 2 is NP-hard.*

Proof. We now suppose that there indeed exists a set of unit length intervals $\mathcal{I}$ such that exactly one point from each color class is covered by an interval

in $\mathcal{I}$. We show that finding such a set of minimum-cardinality is NP-hard. The reduction is from 3-SAT. We use the same variable gadgets and modify the clause gadgets. A clause gadget, c_i, contains 13 points. It contains four consecutive pairs of points, d_{ij}, $1 \leq j \leq 4$ (see Figure 17) and another pair of points, d_i, one of which lies between d_{i1} and d_{i2} and the other lies between d_{i3} and d_{i4}. The remaining three points (blue in Figure 17) lie between d_{i1} and d_i, between d_{i2} and d_{i3} and between d_{i3} and d_i. The Euclidean distance between the right point in d_{ij} and the left point in d_{ij+1}, $1 \leq j \leq 3$, is less than one, ensuring that one unit interval can cover both d_{ij} and d_{ij+1}. The two points that define d_{ij} are spaced unit distance apart.

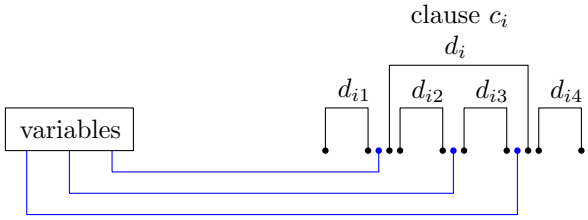


Figure 17: Clause gadget for problem 2.

Again, we line up all of the variables, followed by all of the clauses, so that each consecutive gadget is spaced farther than unit distance apart. Note again that if a clause evaluates to FALSE, each of the three blue points in a clause cannot be covered. Therefore, four unit intervals are required to cover this clause. If a clause evaluates to TRUE, then three intervals (and no less) can cover the clause. Pair d_i is vital to this being true.

The claim is that there exists a satisfying truth assignment in 3-SAT if and only if a minimum covering uses $n + 3m$ unit intervals. Suppose there exists a satisfying truth assignment. No clause can be covered by fewer than three intervals and we need one interval per variable. Therefore, any feasible solution requires at least $n + 3m$ intervals. A satisfying truth assignment achieves the lower bound. Now suppose that a minimum cover uses $n + 3m$ unit intervals. If even one clause evaluated to FALSE we would have required an extra interval. □

Remark: Problem 1 being NP-hard already implies that Problem 2 is NP-hard. However, we rely on the proof of Theorem 22 to prove Theorem 23.

4.4.2 Arbitrary length intervals

With intervals of arbitrary length, there always exists a solution that gives complete coverage. We show that finding such a solution of mininum-cardinality is NP-hard.

Theorem 23. *Problem 3 is NP-hard.*[*]

Proof. The reduction is again from 3-SAT. In this case, spacing of points is irrelevant. Variable gadgets are set up very similarly to the unit interval version. This time, in order to ensure that in a minimum-cardinality cover, the blue points (either to the left or to the right of q_i) in a variable gadget are covered with the same interval that covers q_i, we enclose pair q_i and its surrounding blue points with a 'safety' pair s_i (see Figure 18). We will see that covering a point in q_i and not using the same interval to cover a point in s_i would be too costly. Clause gadgets are set up in the same way as the unit interval, optimization problem (Figure 17).

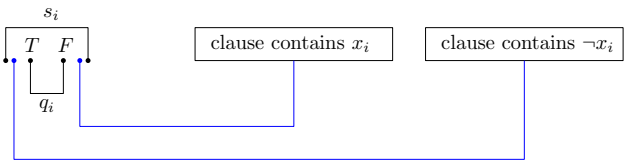


Figure 18: Variable gadget for arbitrary length intervals

We break the set of points in the construction into two halves, H_1 which contains the variable and clause gadgets, and H_2 which contains another gadget which will be described soon (see Figure 19). Surrounding each variable and each clause we place a cluster of $M >> n + 3m$ points. Note that the points in a cluster are laid out side-by-side (rather than on the same x-coordinate). In H_2, we create M groups of points, where each group is made up of $n + m + 1$ points, one paired to each cluster in H_1. Surrounding these groups are pairs of consecutive points, g_1 and g_2. Pair g_1 lies to the left of the first group and pair g_2 lies to the right of the last group. The gadget in H_2 will help us isolate all of the variable and clause gadgets in H_1.

First, we show that any feasible solution uses at least $n+3m+1$ intervals.

[*]A restatement of this problem was previously shown to be NP-hard by another group using a different approach [27].

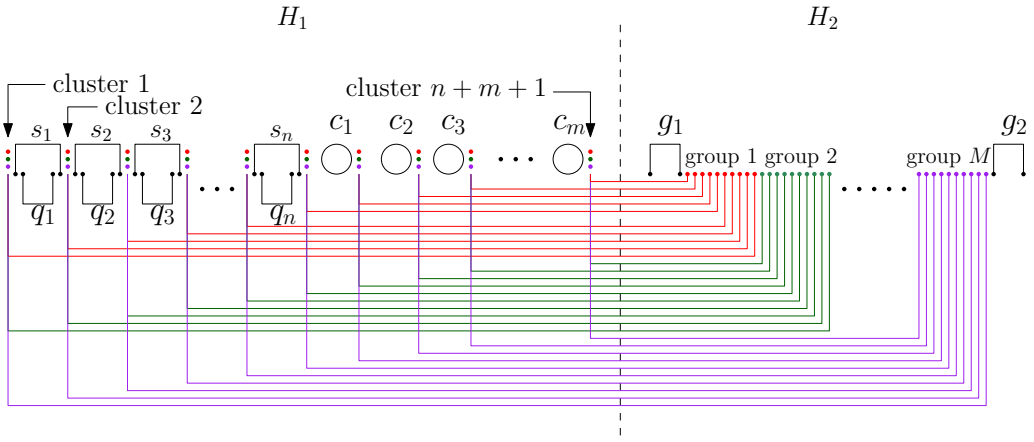


Figure 19: Arbitrary length intervals – the big picture.

Case 1: No cluster in H_1 is completely covered. The variable and clause gadgets are now isolated. We need at least n intervals to cover the variable gadgets and at least $3m$ intervals to cover the clause gadgets. At least one more interval is needed to cover the remaining points in H_2.

Case 2: At least one cluster in H_1 is completely covered. If any cluster is completely uncovered then at least M intervals will be needed in H_2. If all clusters are "touched" by an interval then at least $n + 3m + 1$ intervals will be used in H_1 (at least $n + m + 1$ intervals touch a cluster and at least $2m$ intervals are needed to finish covering the clauses). At least one more interval is needed to cover points in H_2. At least $n + 3m + 2$ intervals are used in total.

Now we claim that there exists a satisfying truth assignment in 3-SAT if and only if a minimum cover uses $n + 3m + 1$ intervals. Suppose there exists a satisfying truth assignment. Any feasible solution must use at least $n + 3m + 1$ intervals. This lower bound can be achieved by covering pairs in H_1 the same way as in the unit interval optimization problem construction and using one more interval in H_2 to cover g_1, all groups, and g_2.

Now suppose that a minimum cover uses $n + 3m + 1$ intervals. By Case 2, we know that no cluster in H_1 can be completely covered. Therefore, all variable and clause gadgets are isolated the same way they were in the unit interval version. Recall that in the variable gadgets, a 'safety' pair s_i encloses the set of blue points that extend to clause gadgets. If the interval used to cover q_i does not also cover pair s_i, then an extra interval will be needed

in the covering; this would be one interval too many. Therefore, we now see that variable gadgets work the same way as in the unit interval version. This means that if any clause would have evaluated to FALSE then at least $n + 3m + 2$ intervals would have been needed. □

Chapter 5

Conflict-free Covering[§]

5.1 Introduction

Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of color classes, where each color class C_i consists of a set of points. We address several closely related covering problems, in which one is allowed to cover at most one point from each color class. Before defining the problems, let us introduce some terminology. Let the set P of point color classes be on a line. We call an interval on the x-axis that contains at most one point from each color class a *conflict-free* interval (or CF-interval for short). We consider the following problems.

Covering color classes with CF-intervals: Given a set P of point color classes on a line where each color class consists of a pair, find a minimum-cardinality set $\mathcal{I}$ of CF-intervals, such that at least one point from each color class is covered by an interval in $\mathcal{I}$.

Covering color classes with arbitrary unit squares: Given a set P of point color classes in the Euclidean plane where each color class consists of a vertically or horizontally unit separated pair of points, find a minimum-cardinality set $\mathcal{S}$ of unit squares (assuming a feasible solution exists), such that exactly one point from each color class is covered by a square in $\mathcal{S}$.

Covering color classes with a convex polygon: Given a set P of point color classes in the Euclidean plane where each color class consists of either a pair or a triple of points, decide whether or not there exists a convex polygon Q such that Q contains exactly one point from each color class. We also consider the related problem in which each color class consists of a pair of points and the goal is to maximize the number of color classes covered by a convex polygon Q, with Q containing exactly one point from each color class.

[§]This chapter is based on joint work with Esther M. Arkin, Aritra Banik, Paz Carmi, Matthew J. Katz, Joseph S. B. Mitchell and Marina Simakov. The work in this chapter appeared in *CCCG 2015* [9].

These problems are motivated by applications in sensor networks. Each point can be considered a sensor. Two points with the same color can be thought of as two sensors that have the exact same data. Covering a set of points with a geometric object corresponds to having a set of sensors interact with the same transceiver. We desire to cover at most one point from each color class in order to prevent or mitigate malicious attacks.

5.1.1 Our results

In Section 5.2 we consider the problem dealing with covering color classes, each consisting of a pair of points, with a minimum-cardinality set of CF-intervals. We prove that it is NP-hard by first proving that the following problem (covering color classes with a *given* set of CF-intervals) is NP-hard. Given a set P of point color classes and a set $\mathcal{I}$ of CF-intervals, find a minimum-cardinality set $\mathcal{I}' \subseteq \mathcal{I}$ (if it exists), such that, at least one point from each color class is covered by an interval in $\mathcal{I}'$. The latter proof is by a reduction from minimum vertex cover. The former proof also requires the following auxiliary result, which we state as an independent theorem. More precisely, we prove that minimum vertex cover remains NP-hard even when we restrict the underlying set of graphs to graphs in which each vertex is of degree at least $|V|/2$, where V is the set of vertices of the graph. We present a 4-approximation algorithm for this problem. We also present a 2-approximation algorithm for covering with arbitrary CF-intervals.

In Section 5.3 we consider the case where P is a set of point color classes in the Euclidean plane.

Suppose each color class consists of a pair and each pair of points from the same color class is unit distance apart, either vertically or horizontally separated. We show that finding a minimum-cardinality set $\mathcal{S}$ of axis parallel unit squares (assuming a feasible solution exists), such that exactly one point from each color class is covered by a square in $\mathcal{S}$ is NP-hard. We then present a 6-approximation algorithm.

We then consider the case that each color class consists of either a pair or triple of points. We show that deciding if there exists a convex polygon Q such that Q contains exactly one point from each color class is NP-complete. If each color class consists of a pair of points, we show that maximizing the number of color classes covered by Q is NP-hard. Finally, we consider the case that each color class consists of an arbitrary amount of points and all points from the same color class are vertically collinear. We (optimally)

maximize the number of color classes covered (exactly one point from each color class) by Q in polynomial time.

5.1.2 Related work

The related work discussed in Section 4.1 is also very much related to the work discussed in this chapter. Following is additional related work. Abellanas et al. [2], Das et al. [39] and Khanteimouri et al. [65] considered problems of the following nature. Given a set of colored point sites in the plane, find the smallest object that encloses at least one point site from each color. Barba et al. [22] consider the following problem. Given a set of n colored points in the Euclidean plane and a vector $c = (c_1, c_2, \ldots, c_m)$ with m being the amount of colors present, find a region (axis-aligned rectangle, square, disk) that encloses exactly c_i points of color i for each i. Claverol et al. [34, 35] considered as input a set S of n line segments and examined the following problem. Find a stabber (line segment, wedge, double-wedge, zigzag) for S so that each segment of S is stabbed exactly once by the stabber.

5.2 Covering Color Classes

Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of n color classes, where each color class C_i is a pair of points $\{p_i, \overline{p_i}\}$ on the x-axis. We call an interval on the x-axis that contains at most one point from each color class a conflict free interval (CF-interval). A main goal in this subsection is to prove that the following problem is NP-hard; additionally, we give a 2-approximation.

Problem 4. Covering color classes with CF-intervals. *Find a minimum-cardinality set $\mathcal{I}$ of* arbitrary *CF-intervals, such that at least one point from each color class is covered by an interval in $\mathcal{I}$.*

Before presenting the proof, we prove that the problem in which one has to pick the covering CF-intervals from a given set of CF-intervals is NP-hard. We then use this result in our proof for Problem 4, together with an auxiliary result stated as Theorem 25 below.

5.2.1 Covering with a given set of CF-intervals

We prove that the following problem is NP-hard.

Problem 5. Covering color classes with a given set of CF-intervals. *Given a set $\mathcal{I}$ of CF-intervals, find a minimum-cardinality set $\mathcal{I}' \subseteq \mathcal{I}$ (if it exists), such that at least one point from each color class is covered by an interval in $\mathcal{I}'$.*

We describe a reduction from vertex cover. A *vertex cover* of a graph G is a subset of the vertices of G, such that each edge of G is incident to at least one vertex of the subset. Given a positive integer k, determining whether there exists a vertex cover of size k is an NP-complete problem [62]. Let $G = (V, E)$ be a graph, where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct a set P of point color classes and a set $\mathcal{I}$ of CF-intervals, such that G has a vertex cover of size k if and only if there exists a subset $\mathcal{I}' \subseteq \mathcal{I}$ of size k that covers at least one point from each color class.

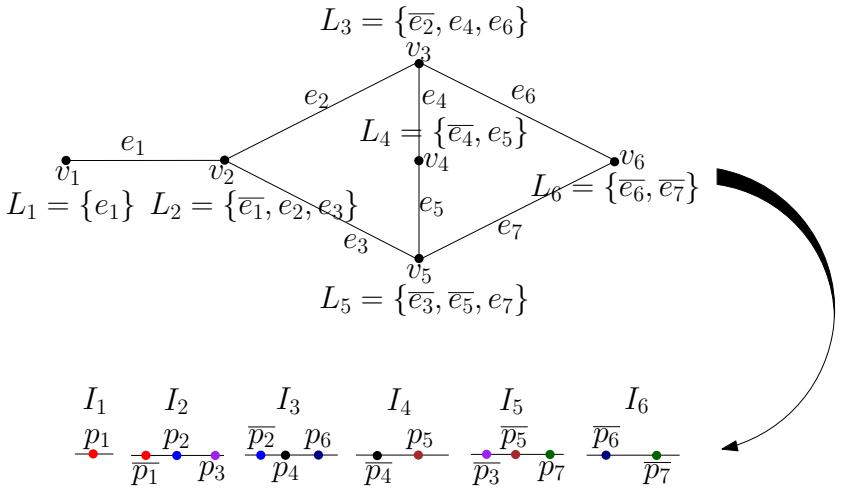


Figure 20: Reduction from vertex cover.

For each vertex v_i create an initially empty set L_i. For each edge $e_k = \{v_i, v_j\}$, where $i < j$, add e_k to L_i and $\overline{e_k}$ to L_j. Now, draw n disjoint intervals on the x-axis, one per set, such that interval I_{i+1} is to the right of interval I_i, $i = 1, \ldots, n-1$. Moreover, for each set L_i, draw $|L_i|$ arbitrary points on the interval I_i as follows. For each element in L_i, if it is of the form e_j, then add the point p_j to I_i, and if it is of the form $\overline{e_j}$, then add the point $\overline{p_j}$ to I_i. Finally, set $P = \{\{p_1, \overline{p_1}\}, \ldots, \{p_m, \overline{p_m}\}\}$ and $\mathcal{I} = \{I_1, \ldots, I_n\}$. See Figure 20 for an illustration.

It is easy to see that G has a vertex cover of size k if and only if there exist k intervals in $\mathcal{I}$ which together cover at least one point from each color

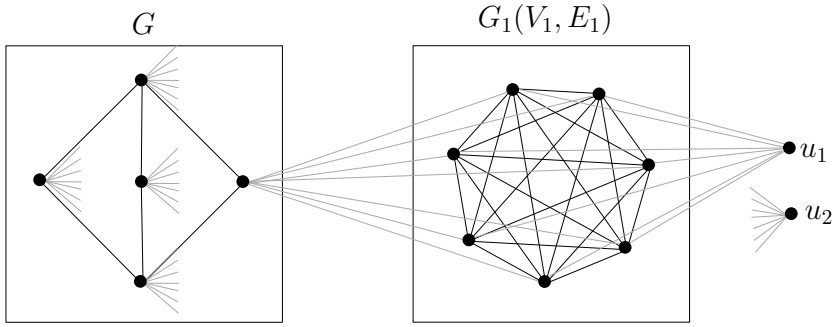


Figure 21: The graph G'.

class in P. Hence we have the following theorem.

Theorem 24. *Problem 5 is NP-hard.*

5.2.2 Covering with arbitrary CF-intervals

In order to show that Problem 4 is NP-hard, we first need to prove the following theorem, which says that minimum vertex cover remains NP-hard even when we restrict our attention to highly dense graphs.

Theorem 25 (Min vertex cover in dense graphs)**.** *Finding a minimum vertex cover of a graph in which the degree of each vertex is at least $\frac{n}{2}$ is NP-hard, where n is the number of vertices in the graph.*

Proof. Let $G = (V, E)$ be any graph. We construct a new graph $G' = (V', E')$ in which the degree of each vertex is at least $\frac{|V'|}{2}$, and show that one can immediately obtain a minimum vertex cover of G from a minimum vertex cover of G' (and vice versa).

Let $G_1 = (V_1, E_1)$ be the complete graph of $|V|+2$ vertices. We construct G' as follows. Set $V' = V \cup V_1 \cup \{u_1, u_2\}$, where u_1, u_2 are two new vertices. Set $E' = E \cup E_1 \cup E_2 \cup E_3$, where $E_2 = V \times V_1$ and $E_3 = V_1 \times \{u_1, u_2\}$ (see Figure 21). Notice that G' has the desired property, i.e., for each $v \in V'$, the degree of v (in G') is at least $\frac{|V'|}{2} = \frac{2|V|+4}{2} = |V| + 2$. (If v comes from V, then $deg_{G'}(v) = deg_G(v) + |V| + 2 \geq |V| + 2$, if v comes from V_1, then $deg_{G'}(v) = deg_{G_1}(v) + |V| + 2 \geq |V| + 2$, and if $v \in \{u_1, u_2\}$, then $deg_{G'}(v) = |V_1| = |V| + 2$.)

We now claim that given a minimum vertex cover of G', one can immediately obtain a minimum vertex cover of G, and vice versa. Let V^* be a

minimum vertex cover of G'. We first show that $V_1 \subseteq V^*$. Since G' contains the complete graph G_1 of size $|V|+2$, any minimum vertex cover of G' must include at least $|V|+1$ vertices of V_1. If one of V_1's vertices, v, is not in V^*, then both u_1 and u_2 are necessarily in V^* (to cover the edges $\{v, u_1\}, \{v, u_2\}$). But, if so, V^* is not a minimum vertex cover, since $V^* \setminus \{u_1, u_2\} \cup \{v\}$ is also a vertex cover of G'. We conclude that $V_1 \subseteq V^*$. Notice that V_1 covers all the edges in E' except for the edges in E. Thus, the rest of the vertices in V^* consist of a minimum vertex cover of G. In other words, $V^* \cap V$ is a minimum vertex cover of G.

On the other hand, let $\tilde{V}$ be a minimum vertex cover of G, then $V_1 \cup \tilde{V}$ is a minimum vertex cover of G'. (Since, as shown above, V_1 is contained in any minimum vertex cover of G', and in order to cover the remaining uncovered edges, we need a minimum vertex cover of G.) □

Corollary 2. *Finding a minimum vertex cover of a graph $G = (V, E)$ in which the degree of each vertex is at least $\varepsilon|V|$, where $0 < \epsilon < 1$, is NP-hard.*

Proof. Similar to the proof of Theorem 25. □

We are now ready to prove that Problem 4 is NP-hard. We describe a reduction from minimum vertex cover in dense graphs (see Theorem 25 above). Let $G = (V, E)$ be any graph in which the degree of each vertex is at least $\frac{n}{2}$, where $n = |V|$. By Dirac's theorem [40] (or Ore's theorem [77]), G contains a Hamiltonian cycle; moreover, Palmer [78] presented a simple and efficient algorithm for computing such a cycle, under the conditions of Ore's theorem.

Let $v_1, v_2, \ldots, v_n, v_1$ be a Hamiltonian cycle in G. As for Problem 5, we construct a set P of point color classes. For each vertex $v_i \in V$, we construct a set L_i as follows. For each edge $e_k = \{v_i, v_j\}$ adjacent to v_i, we add e_k (resp., $\overline{e_k}$) to L_i, if $i < j$ (resp., $j < i$). We now draw n disjoint intervals on the x-axis, such that interval I_i corresponds to set L_i and precedes interval I_{i+1} (for $i < n$). We draw $|L_i|$ points in I_i as follows. Let $e_j = \{v_{i-1}, v_i\}$ and $e_k = \{v_i, v_{i+1}\}$. Then $\overline{e_j}, e_k \in L_i$. Place a point $\overline{p_j}$ corresponding to $\overline{e_j}$ at the left endpoint of I_i and place a point p_k corresponding to e_k at the right endpoint of I_i. In addition, place a point anywhere in the interior of I_i, for each of the other elements in L_i. For example, in Figure 22 e_j and e_k are the edges connecting v_i to v_{i-1} and to v_{i+1}, respectively, and e_1, e_2, e_3, e_4 are the other edges incident to v_i. The corresponding interval representation is shown in Figure 22.

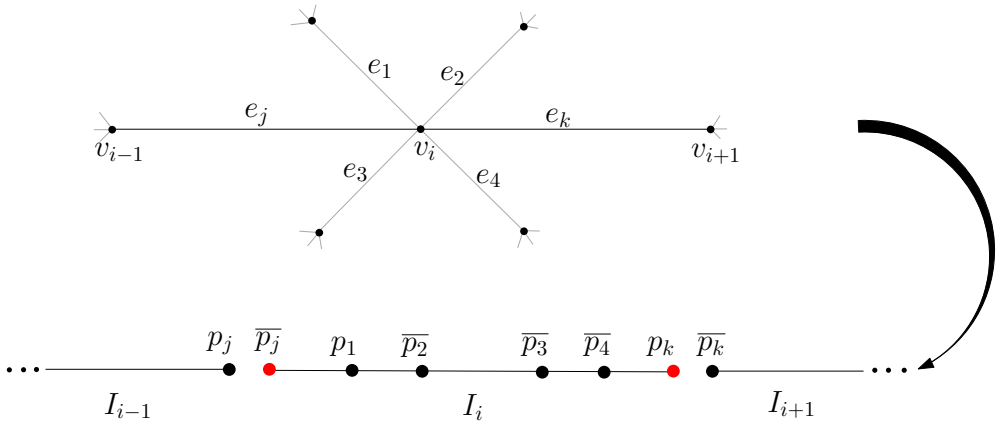


Figure 22: Illustration of Theorem 26.

Now, set $P = \{\{p_1, \overline{p_1}\}, \{p_2, \overline{p_2}\}, \ldots\}$ and $\mathcal{I} = \{I_1, \ldots, I_n\}$. Observe that I_i is conflict free (by construction), for $i = 1, \ldots, n$. Moreover, any other CF-interval is necessarily contained in one of the intervals already in $\mathcal{I}$ (since any interval that covers the right endpoint of I_i and the left endpoint of I_{i+1} is not conflict free). Thus, one might as well pick intervals from $\mathcal{I}$ when covering the color classes of P with a minimum number of arbitrary CF-intervals. But, by Theorem 24 this is NP-hard. Hence we have the following theorem.

Theorem 26. *Problem 4 is NP-hard.*

A 4-approximation algorithm for Problem 5.
Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of point color classes (pairs, $C_i = \{p_i, \overline{p_i}\}$) on the set of points $\mathcal{P} = \bigcup_i C_i$. We assume there exists $\mathcal{I}' \subseteq \mathcal{I}$ such that $\mathcal{I}'$ covers at least one point from each color class and we provide a 4-approximation algorithm for covering P with the fewest number of CF-intervals. For a given $p \in \mathcal{P}$, let $I_p \in \mathcal{I}$ be a CF-interval (if it exists) that covers p and extends farthest to the right among all intervals that cover p. Let $I_p^{(r)} \subseteq I_p$ be the subinterval of I_p that contains p and all points to the right of p.

Lemma 14. $|\mathcal{I}'| \leq 4|OPT|$.

Proof. Consider the set $\mathcal{T}$ of intervals at the end of the while loop. Let $OPT_{\mathcal{T}} \subseteq \mathcal{T}$ be an optimal set cover of the C_i's. First we claim that $|OPT_{\mathcal{T}}| \leq 2|OPT|$. Consider the leftmost point p in an arbitrary interval A of OPT. By the construction of Algorithm 2, we know that there must exist an interval $T \in \mathcal{T}$ that contains p. If there exists a point that is covered by A and not

Algorithm 2 An algorithm for Problem 5.

Input: $P = \{C_1, C_2, \ldots, C_n\}$, a set of point color classes and $\mathcal{I}$, a set of CF-intervals.

Output: A subset $\mathcal{I}' \subseteq \mathcal{I}$ covering at least one point from each color class.

$\mathcal{T} = \emptyset$

while there exists $p \in \mathcal{P}$ such that p is uncovered in $\mathcal{T}$ and there exists $I \in \mathcal{I}$ such that $p \in I$ **do**

Let p be the leftmost uncovered point in $\mathcal{P}$ that is contained in some interval in $\mathcal{I}$.

$\mathcal{T} \leftarrow \mathcal{T} \cup I_p^{(r)}$

Compute a subset of intervals $\mathcal{T}$ to cover at least one point of each of the C_i's, using a low-frequency set cover approximation algorithm.

Let $\mathcal{I}'$ be the set of intervals $I_p \in \mathcal{I}$ corresponding to each $I_p^{(r)}$ of $\mathcal{T}$ in the resulting cover.

covered by T, then let q be the leftmost such point. We know there exists an interval $I_q^{(r)} \in \mathcal{T}$ that starts at q and extends at least as far to the right as does A. Thus, for any $A \in OPT$, there exist at most two intervals in $\mathcal{T}$, the union of which entirely contains A.

Observe that since each newly added interval to $\mathcal{T}$ cannot contain a previously covered point, then, at the end of the while loop, each $p \in \mathcal{P}$ is contained in at most one interval of $\mathcal{T}$; thus, each pair C_i is covered by at most two intervals of $\mathcal{T}$ (one covering p_i, one covering $\overline{p_i}$). Therefore, we are approximating a low-frequency (at most 2) set cover instance, for which LP relaxation gives a 2-approximation [89] (pp. 119-120). Hence, we have $|\mathcal{I}'| \leq 2|OPT_{\mathcal{T}}| \leq 4|OPT|$. (For color classes of size at most c, we obtain a $2c$-approximation.) □

A 2-approximation algorithm for Problem 4.

Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of point color classes (pairs, $C_i = \{p_i, \overline{p_i}\}$) on the set of points $\mathcal{P} = \bigcup_i C_i$. We provide a simple 2-approximation algorithm for covering P with arbitrary CF-intervals. For any point $p \in \mathcal{P}$, denote the maximal CF-interval starting at p and ending at a point of $\mathcal{P}$ to the right of p (or at p) by $I_{\max}(p)$.

Consider the set $\mathcal{I}$ computed by Algorithm 3. Clearly, $\mathcal{I}$ is a set of (disjoint) CF-intervals, such that at least one point from each color class is covered by the intervals of $\mathcal{I}$. It remains to prove that $\mathcal{I}$ is a 2-approximation

Algorithm 3 A greedy algorithm for Problem 4.

Input: $P = \{C_1, C_2, \ldots, C_n\}$, a set of point color classes.
Output: A set $\mathcal{I}$ of CF-intervals.
$\mathcal{I} = \emptyset$
while $\mathcal{P} \neq \emptyset$ **do**
  Let p be the leftmost point in $\mathcal{P}$
  $\mathcal{I} \leftarrow \mathcal{I} \cup I_{\max}(p)$
  For each point of $\mathcal{P}$ that lies in $I_{\max}(p)$, remove it and its twin point from $\mathcal{P}$

of OPT, where OPT denotes any optimal solution.

Lemma 15. $|\mathcal{I}| \leq 2|OPT|$.

Proof. For any two points x and y, let $[x, y]$ (resp., (x, y)) denote the closed (resp., open) interval with endpoints x and y. Let $[p_a, p_b]$ and $[p_c, p_d]$ be two consecutive intervals in $\mathcal{I}$. Observe that since $[p_a, p_b]$ is a maximal CF-interval, there exists a point p_i (resp., $\overline{p_i}$) in $[p_a, p_b]$, such that $\overline{p_i}$ (resp., p_i) is in (p_b, p_c). Therefore any interval in OPT can intersect at most two intervals in $\mathcal{I}$. Moreover, since OPT must cover the color class $C_i = \{p_i, \overline{p_i}\}$, there exists an interval $I \in OPT$, such that $I \cap \{p_i, \overline{p_i}\} \neq \emptyset$. We thus conclude that $|OPT| \geq |\mathcal{I}|/2$. □

5.3 Two Dimensions

Let $P = \{C_1, C_2, \ldots, C_n\}$ be a set of n color classes in the Euclidean plane. We explore covering problems where exactly one point from each color class must be covered.

5.3.1 Unit Squares

Problem 6. Covering color classes with arbitrary unit squares. *Let $P = \{C_1, C_2, \ldots, C_n\}$ be in the Euclidean plane and let each color class C_i consist of a vertically or horizontally unit separated pair of points. Find a minimum-cardinality set $\mathcal{S}$ of axis-aligned unit squares (assuming a feasible solution exists), such that exactly one point from each color class is covered by a square in $\mathcal{S}$.*

Theorem 27. *Problem 6 is NP-hard.*

Proof. The reduction is from PLANAR 3-SAT [70], where one is given a formula in conjunctive normal form with at most three literals per clause, with the objective of deciding whether or not the formula is satisfiable. Given variables $\{x_1, x_2, \ldots, x_n\}$ and clauses $\{c_1, c_2, \ldots, c_m\}$, we consider the graph whose nodes are the clauses and variables and whose edges join variable x_i with clause c_j if and only if $x_i \in c_j$ or $\neg x_i \in c_j$. The resulting bipartite graph, G, is planar.

In a manner similar to Fowler et al. [45], in a planar embedding of G we replace all of the edges incident to a variable node with a variable chain that visits the corresponding clauses and returns to the variable node to form a loop. The variable chains consist of a sequence of unit separated pairs (see Figure 23) and are designed in such a way that any minimum cardinality solution will either cover $\{\overline{a_{i+k}}, a_{i+k+1} : \text{k is even}\}$ or $\{\overline{a_{i+k}}, a_{i+k+1} : \text{k is odd}\}$. That is, for a given variable chain, either all blue unit squares or all red unit squares will be used. Using red (resp. blue) squares for variable x_i is equivalent to setting this variable to TRUE (resp. FALSE). Using planarity of the graph embedding, no two variable chains intersect, and any two points from different chains are spaced at least unit distance apart.

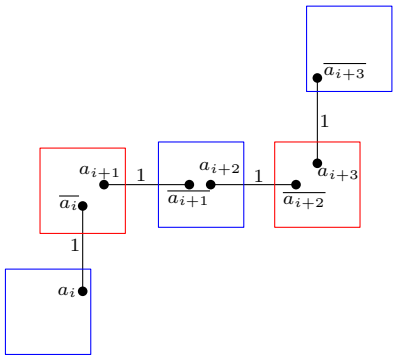


Figure 23: Variable chain.

Clause c_i consists of a single (green) pair (see Figure 24). If c_i evaluates to FALSE, then a square that is not associated with any variable loop will be needed to cover c_i. If c_i evaluates to TRUE, then a point from c_i can be covered by a square from an incoming loop whose literal evaluates to TRUE.

Let r_i be the number of pairs used in variable chain $i, 1 \leq i \leq 3m$. We design the variable chains so that r_i is even for all i. Let $r = \sum_i r_i$. It is

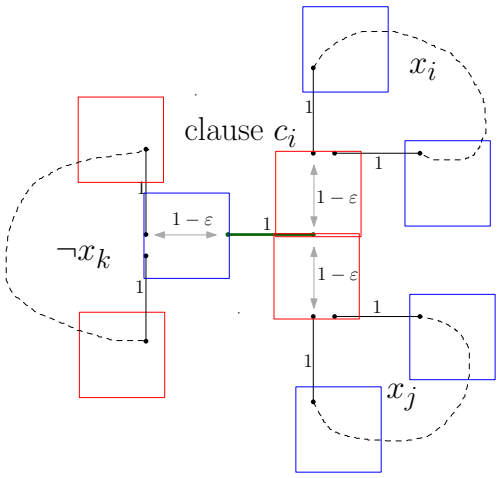


Figure 24: Clause gadget.

now apparent that there exists a satisfying truth assignment in PLANAR 3-SAT if and only if a minimum cardinality covering with unit squares uses $\frac{r}{2}$ squares. □

Remark: If P is on a line and pairs are unit separated, we can minimize the number of unit intervals used in a complete cover (assuming a solution exists) in polynomial time using dynamic programming.

5.3.1.1 A 6-approximation algorithm.

We lay out a grid with unit dimensions on top of our point set P and two-color the cells of the grid red and black in the style of a checkerboard. We say that a cell is occupied if it contains a point in P. Let R be the set of occupied red cells and B the set of occupied black cells. As a solution, we use the set of smaller cardinality.

Lemma 16. $\min\{|R|, |B|\} \leq 6|OPT|$.

Proof. Suppose w.l.o.g that $\min\{|R|, |B|\} = |R|$. Note that R is a feasible solution because any two points of a color class are unit separated either vertically or horizontally, thus one of the two points must occupy a red cell and the other must occupy a black cell. Therefore, R covers all color classes of points and no two points from the same color class are covered by R.

Now we claim that in the optimal solution, OPT, at least $\frac{1}{12}(|R| + |B|)$ unit squares are used. An arbitrary unit square, s, used in OPT stabs at most four cells of the checkerboard. These four cells are adjacent to at most eight other cells in total, each of which can be occupied by the pair of one of the points covered by s. Thus, at most 12 occupied cells of the checkerboard can be accounted for by any unit square used in OPT.

Combining the fact that $\min\{|R|, |B|\} \leq \frac{1}{2}(|R| + |B|)$ and $|OPT| \geq \frac{1}{12}(|R| + |B|)$, we have that $\min\{|R|, |B|\} \leq 6|OPT|$. □

5.3.2 Covering with a Convex Polygon

Problem 7. *Let $P = \{C_1, C_2, \ldots, C_n\}$ be in the Euclidean plane and let each color class C_i consist of either a pair or a triple of points. Decide whether or not there exists a convex polygon Q such that Q contains exactly one point from each color class.*

Problem 8. *Let $P = \{C_1, C_2, \ldots, C_n\}$ be in the Euclidean plane and let each color class C_i consist of a pair of points. Maximize the number of color classes covered by a convex polygon Q such that Q contains exactly one point from each covered color class.*

Theorem 28. *Problem 7 is NP-complete.*

Proof. Problem 7 is clearly in NP because we can check whether or not polygon Q is convex and whether or not Q contains exactly one point from each color class in polynomial time. We present a reduction from EXACTLY 1-IN-3-SAT, where one is given a formula in conjunctive normal form with at most three literals per clause, with the objective of deciding whether or not the formula is satisfiable. In a satisfying assignment, every clause must contain exactly one TRUE literal.

Given variables $\{x_1, x_2, \ldots, x_n\}$ and clauses $\{c_1, c_2, \ldots, c_m\}$, we start by considering $2n$ points, $S = \{s_1, s_2, \ldots, s_{2n}\}$, in the position of a regular $2n$-gon. These $2n$ points are not part of any color class; we use them to help explain the construction. We place two pairs of points around each point of S in such a way that convex polygon Q must have vertices at each point of S (see Figure 25). We create a variable gadget x_i in between points s_{2i-1} and s_{2i} for $1 \leq i \leq n$. Each variable gadget consists of color class that is a pair of points $\{q_i, \overline{q_i}\}$, $1 \leq i \leq n$ (see Figure 26). We place $\{q_i, \overline{q_i}\}$ so that Q can be expanded to cover either q_i (green lines in Figure 26) or $\overline{q_i}$ (red lines

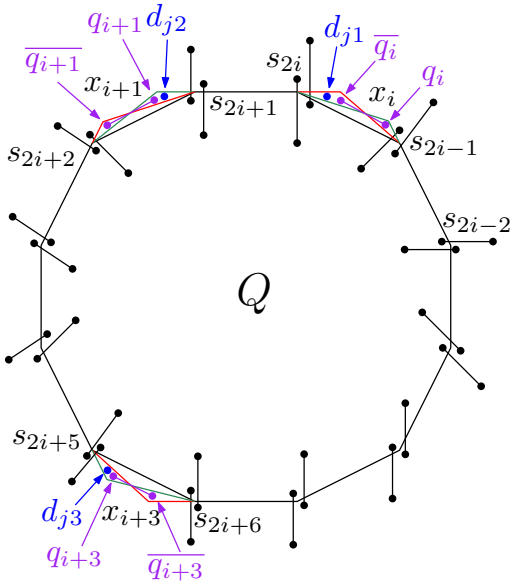


Figure 25: Construction of hardness for Problem 7.

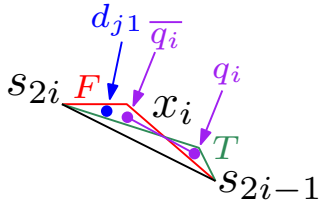


Figure 26: Close-up of variable gadget for Problem 7.

in Figure 26), while remaining convex. Setting x_i to TRUE (resp. FALSE) corresponds to expanding Q to cover q_i (resp. $\overline{q_i}$). If x_i (resp. $\neg x_i$) appears in clause c_j, a point from a color class that contains triple $\{d_{j1}, d_{j2}, d_{j3}\}$ will appear in the expansion of Q that covers q_i (resp. $\overline{q_i}$), and not in the expansion of Q that covers $\overline{q_i}$ (resp. q_i). It is now apparent that there exists a satisfying truth assignment in EXACTLY 1-IN-3-SAT if and only if convex polygon Q covers exactly one point from each color class. □

Theorem 29. *Problem 8 is NP-hard.*

Proof. The reduction is from MAX EXACTLY 1-IN-2-SAT where each clause has at most two literals and the objective is to maximize the number of clauses that evaluate to TRUE. A clause evaluates to TRUE if and only if it contains exactly one TRUE literal. Using the same construction as in Problem 7, it is easy to see that maximizing the number of TRUE clauses is equivalent to maximizing the number of color classes covered. □

Chapter 6

Network Optimization on Partitioned Pairs of Points ¶

6.1 Introduction

We study a class of network optimization problems on *pairs* of sites in a metric space. Our goal is to determine how to split each pair, into a "red" site and a "blue" site, in order to optimize *both* a network on the red sites and a network on the blue sites. In more detail, given n pairs of points, $\mathcal{S} = \{\{p_1, q_1\}, \{p_2, q_2\}, \ldots, \{p_n, q_n\}\}$, in the Euclidean plane or in a general metric space, we partition the points in $S = \bigcup_{i=1}^{n} \{p_i, q_i\}$ into two sets, S_1 and S_2, so that $p_i \in S_1$ if and only if $q_i \in S_2$. The partition should optimize the cost of certain structures built on both S_1 and S_2: spanning trees, traveling salesman tours (TSP tours) or matchings. Let $f(X)$ be a certain structure computed on point set X and let $\lambda(X)$ be the bottleneck edge of a structure computed point set X. For each of the aforementioned structures we consider the objective of minimizing $|f(S_1)| + |f(S_2)|$, minimizing $max\{|f(S_1)|, |f(S_2)|\}$ and minimizing $max\{|\lambda(S_1)|, |\lambda(S_2)|\}$. Here, $|\cdot|$ denotes the cost (e.g., sum of edge lengths) of a certain structure.

The problems we study are natural variants of well-studied network optimization problems. Our motivation comes also from a model of secure connectivity in networks involving sensors with replicated data. Consider a set of sensors each having two (or more) replications of their data; the sensors are associated with pairs of points (or k-tuples of points in the case of higher levels of replication). Our goal may be to compute two networks (a "red" network and a "blue" network) to interconnect the sensors, each network visiting exactly one data site from each sensor type; for communication connectivity, we would require each network to be a tree, while for servicing sensors with a mobile agent, we would require each network to be a Hamiltonian path/cycle. By keeping the red and blue networks distinct, a malicious attack within one network is isolated from the other.

¶This chapter is based on joint work with Esther M. Arkin, Aritra Banik, Paz Carmi, Su Jia, Matthew J. Katz, Tyler Mayer and Joseph S. B. Mitchell. The work in this chapter has been submitted for publication.

6.1.1 Our results

We show that several of these problems are NP-hard and we give O(1)-approximation algorithms for each of them. Table 5 summarizes our O(1)-approximation results.

	min $\|f(S_1)\| + \|f(S_2)\|$	min-max$\{\|\lambda(S_1)\|, \|\lambda(S_2)\|\}$	min-max$\{\|f(S_1)\|, \|f(S_2)\|\}$
Matching	2	3	3
Spanning tree	3α	9 3 $\mathbb{R}^1$	4α
TSP tour	3β	18	6β

Table 5: Table of results: α is the Steiner ratio for a particular metric space. β is the approximation factor for the traveling salesperson problem in a certain metric space.

6.1.2 Related work

Several optimization problems have been studied of the following sort: Given sets of tuples of points (in a Euclidean space or a general metric space), select exactly one point or at least one point from each tuple in order to optimize a specified objective function on the selected set. Many problems of this sort have been discussed in Section 4.1 and in Section 5.1.2. In addition, Myung et al. [75] introduced the Generalized Minimum Spanning Tree Problem: Given an undirected graph with the nodes partitioned into subsets, compute a minimum spanning tree that uses exactly one point from each subset. They show that this problem is NP-hard and that no constant-factor approximation algorithm exists for this problem unless $P = NP$. Related work addresses the generalized traveling salesperson problem [23, 80, 81, 83], in which a tour must visit one point from each of the given subsets.

While optimization problems of the "one of a set" flavor have been studied extensively, the problems we study here are fundamentally different: we care not just about a single structure (e.g., network) that makes the best "one of a set" choices on, say, pairs of points; we must consider also the cost of a second network on the "leftover" points (one from each pair) *not* chosen. As far as we know, the problem of partitioning points from pairs into two sets in order to optimize objective functions on *both* sets has not been extensively studied. One recent work of Arkin et al. [10] does address optimizing objectives on both sets: Given a set of pairs of points in the Euclidean plane, color the

points red and blue so that if one point of a pair is colored red (resp. blue), the other must be colored blue (resp. red). The objective is to optimize the radii of the minimum enclosing disk of the red points and the minimum enclosing disk of the blue points. They studied the objectives of minimizing the sum of the two radii and minimizing the maximum radius.

6.2 Spanning Trees

Let $MST(X)$ be a minimum spanning tree over the point set X, and $|MST(X)|$ be the cost of the tree. Let $\lambda(X)$ be the bottleneck edge in a spanning tree on point set X and $|\lambda(X)|$ be the cost of the bottleneck edge. Given n pairs of points in a metric space, partition the point set S into two sets, S_1 and S_2, so that for pair $\{p_i, q_i\}$, $p_i \in S_1$ if and only if $q_i \in S_2$ and the cost of a spanning tree built over each set is minimized.

6.2.1 Minimum Sum

In this section we consider minimizing $|MST(S_1)| + |MST(S_2)|$.

Theorem 30. *The Min-Sum 2-MST problem is NP-hard in general metric spaces.*

Proof. The reduction is from Max 2SAT where one is given n variables $\{x_1, x_2, \ldots, x_n\}$ and m clauses $\{c_1, c_2, \ldots, c_m\}$. Each clause contains at most two literals joined by a logical *or*. The objective is to maximize the number of clauses that evaluate to *true*.

For each variable x_i we create a variable gadget that consists of two pairs of points: $\{p_{2i}, q_{2i}\}$ and $\{p_{2i+1}, q_{2i+1}\}$ (see Figure 27). Setting x_i to *true* is equivalent to using edges (p_{2i+1}, q_{2i}) and (p_{2i}, q_{2i+1}). Setting x_i to *false* is equivalent to using edges (p_{2i}, p_{2i+1}) and (q_{2i+1}, q_{2i}). Variable gadgets will be arranged on a line with distance $O(L)$ between consecutive variable gadgets for $L = \Omega(n + m)$ (see Figure 28).

For every pair of variable gadgets corresponding to variables $x_i, x_j, i \neq j$ we place a cluster $A_{i,j}$ of $M = \Omega(m^2)$ points near point p_{2i+1}. Each of these points is paired to a point in a cluster $B_{i,j}$ of M points near point q_{2j+1}. Any two points in the same cluster, $A_{i,j}$ or $B_{i,j}$, are separated by distance two from each other and by distance one from point p_{2i+1}, q_{2j+1} respectively. Note that this enforces points p_{2i+1} and q_{2j+1} to be in different trees for all $1 \leq i, j \leq n$. Otherwise, if p_{2i+1} and q_{2j+1} were placed in the same tree, then

connecting the points in clusters $A_{i,j}, B_{i,j}$ to the trees would cost at least M more than it would to have p_{2i+1} and q_{2j+1} in different trees.

Now we argue that the optimal solution uses edges (p_{2i+1}, p_{2i+3}) and (q_{2i+1}, q_{2i+3}), $1 \leq i \leq n-1$, as "backbones" of the two MSTs. To see this, observe that if any other edge was used to connect two consecutive variable gadgets, then we would need to use at least one edge of length $L+2$. Since p_{2i+1} and q_{2j+1} will be in different trees for all $1 \leq i, j \leq n$ and since points p_{2i} and q_{2i} will be connected to points p_{2i+1} and q_{2i+1} $(1 \leq i \leq n)$, we have a set of "lower" components that must be connected and a set of "upper" components that need to be connected. No upper component can be connected to a lower component. Any edge of length at least $L+2$ connecting any of these components can thus be replaced by an edge of length L.

The remaining variable gadget points, $\{p_{2i}, q_{2i}\}$ $(1 \leq i \leq n)$, must be connected to the backbones. That is, for variable x_i, points p_{2i} and q_{2i} will be picked up either by using edges (p_{2i+1}, q_{2i}) and (p_{2i}, q_{2i+1}) (green in Figure 29) or edges (p_{2i}, p_{2i+1}) and (q_{2i+1}, q_{2i}) (red in Figure 29). As mentioned, the green edges correspond to setting x_i to *true* and the red edges correspond to setting x_i to *false*.

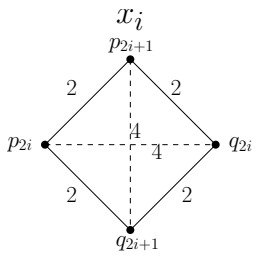


Figure 27: Variable gadget.

A clause gadget consists of a configuration of 3 point pairs surrounding variable gadgets corresponding to the variables in that clause (see Figure 30). The placement of the 3 point pairs depends on whether the literals appear positively or negatively.

Consider clause c_i which consists of variables x_j and x_k. We create a pair of points $\{a_i, b_i\}$, each of which will be placed next to variable x_j. If x_j appears negatively in c_i, then we place a_i at distance 1 away from an endpoint of a green edge of x_j and place b_i at distance 1 away from the other endpoint of the green edge (see Figure 30c). If x_j appears positively in c_i,

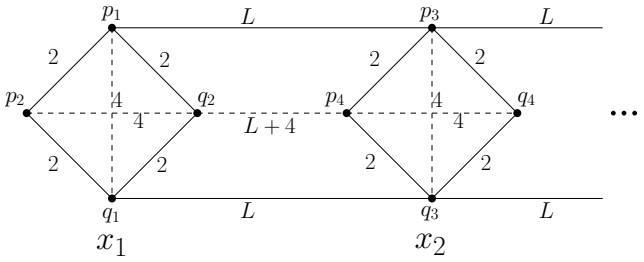


Figure 28: Metric distances between variable gadgets.

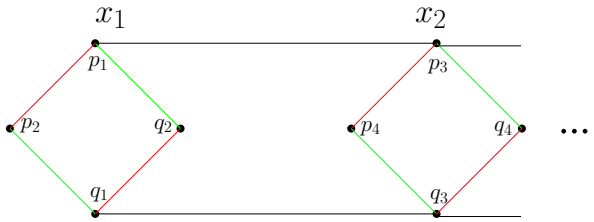
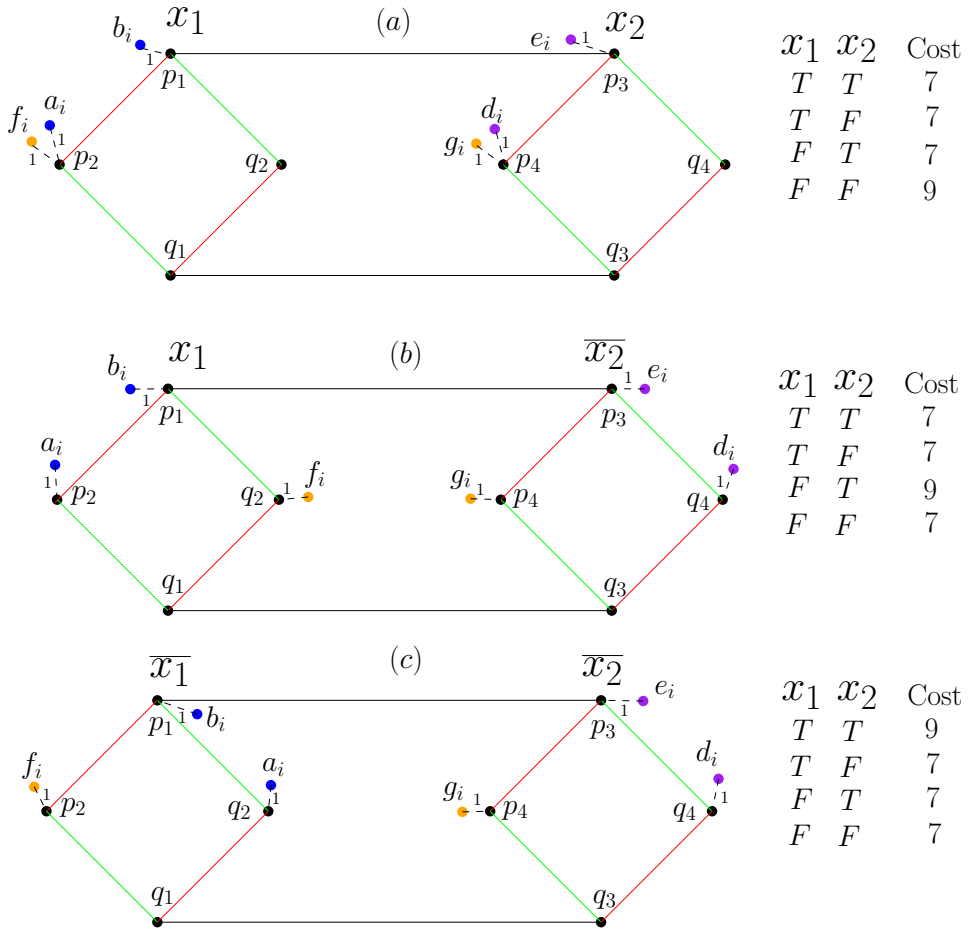


Figure 29: Truth assignment.

then we do the same thing at the endpoints of a red edge (see Figure 30a). Then, we create a pair of points $\{d_i, e_i\}$ and follow the same procedure for variable x_k. Finally, for each clause gadget we create a pair of points $\{f_i, g_i\}$ and place them at a distance of 1 from certain variable gadget points chosen based on how many literals appear positively in clause c_i; zero, one, or two. The placement of $\{f_i, g_i\}$ for all three cases can be found in Figure 30.

As a technical note, to complete the construction, all clause gadget points placed around the same variable gadget vertex are separated from each other by distance 2, and these points are at distance only 1 from the nearest variable gadget vertex. This will ensure that the optimal solution will not link any two clause gadget points to each other. Also, note that we use the metric completion to define the weights of the rest of the edges in the graph.

Connecting all points except those associated with clause gadgets into two MSTs has a base cost of $(2n-2)L+4n+2\binom{n}{2}M$. Now note that a clause evaluates to *true* if and only if it costs 7 units to attach the clause gadget points to the backbones. A clause evaluates to *false* if and only if it costs 9 units to attach the clause gadget points to the backbones (see Figure 30). Thus, it is now apparent that there exists a truth assignment in 2SAT with k clauses satisfied if and only if there exists a solution to the Min-Sum 2-MST problem with cost $(2n-2)L + 4n + 2\binom{n}{2}M + 7k + 9(m-k)$. □



x_1	x_2	Cost
T	T	7
T	F	7
F	T	7
F	F	9

x_1	x_2	Cost
T	T	7
T	F	7
F	T	9
F	F	7

x_1	x_2	Cost
T	T	9
T	F	7
F	T	7
F	F	7

Figure 30: Placement of clause gadget points and extra cost incurred to incorporate clause gadget points into two MSTs once a truth assignment over the variables is fixed.

An $O(1)$-approximation algorithm for Min-Sum 2-MST problem. Compute $MST(S)$, a minimum spanning tree on all $2n$ points. Imagine removing the heaviest edge, h, from $MST(S)$. This leaves us with two trees; T_1 and T_2. Perform a preorder traversal on T_1, assigning nodes to S_1 as long as there is no conflict. If there is a conflict (q_i is reached in the traversal and p_i was already assigned to S_1) then assign the node to S_2. Repeat this for T_2. We then return S_1 and S_2 as our approximate partition. In order to help analyze this algorithm, we will think of the nodes assigned to S_1 as colored red (set R) and the nodes assigned to S_2 as colored blue (set B).

- Case 1: All nodes in T_1 are of the same color and all nodes in T_2 are of the same color.

  This partition is optimal. To see this, note that the weight of $MST(S) \setminus h$ is a lower bound on the cost of the optimal solution as it is the cheapest way to create two trees, the union of which span all of the input nodes. Since each tree is single colored, we know that each tree must have n points, exactly one from each pair, and thus is also feasible to our problem.

- Case 2: One tree is multicolored and the other is not.

  Let OPT be the optimal solution. Suppose WLOG that T_1 contains only red nodes and T_2 contains both blue and red nodes. Then, $\exists\, i$: $p_i, q_i \in T_2$. Imagine also constructing an MST on each side of the optimal partition. By definition, in the MSTs built over the optimal partition, at least one point in T_2 must be connected to a point in T_1. This implies that the weight of the optimal solution is at least as large as $|h|$, as h is the cheapest edge which spans the cut (T_1, T_2). Therefore, $|h| \leq |OPT|$.

  Consider $MST(R)$. By the Steiner property, we have that an MST over a subset $U \subseteq S$ has weight at most $\alpha|MST(S)|$ where α is the Steiner ratio of the metric space. Recall that $|MST(S) \setminus h| \leq |OPT|$. In this case, since $|h| \leq |OPT|$, we have that $|MST(R)| \leq \alpha|MST(S)| \leq 2\alpha|OPT|$.

  Next, consider building $MST(B)$. Since no blue node exists in T_1, there does not exist an edge that crosses the cut (T_1, T_2) in $MST(B)$, and

thus we have that $|MST(B)| \leq \alpha|MST(S) \setminus h| \leq \alpha|OPT|$. Therefore, $|MST(R) \cup MST(B)| \leq 3\alpha|OPT|$.

- Case 3: Both trees are multicolored.

  In this case, $\exists\, i \;:\; p_i, q_i \in T_1$ and $\exists\, j \;:\; p_j, q_j \in T_2$. Imagine also constructing an MST on each side of the optimal partition. In this case, there must be at least two edges crossing the cut (T_1, T_2), one edge belonging to each tree built on a partition of the optimal solution. Note that each of these edges has weight at least $|h|$ as h is the cheapest edge spanning the cut (T_1, T_2), implying that $|h| \leq |OPT|/2$. Thus, $|MST(S)| \leq 1.5|OPT|$ as $|MST(S) \setminus h| \leq |OPT|$ and $|h| \leq |OPT|/2$.

  Using our partition, one can compute $MST(B)$ and $MST(R)$, each with weight at most $\alpha|MST(S)|$. Therefore $|MST(R) \cup MST(B)| \leq 2\alpha|MST(S)| \leq 3\alpha|OPT|$, where α is again the Steiner ratio of the metric space.

Using the above case analysis, we have the following theorem.

Theorem 31. *There exists a 3α-approximation for the Min-Sum 2-MST problem.*

Remark: In a general metric space $\alpha = 2$ and in the Euclidean plane $\alpha \leq 1.3546$ [59].

6.2.2 Min-max

In this section the objective is to $min - max\{|MST(S_1)|, |MST(S_2)|\}$.

Theorem 32. *The Min-Max 2-MST problem is weakly NP-hard.*

Proof. The reduction is from Partition. In the partition problem we are given a set of n integers, $P = \{x_1, x_2, \ldots, x_n\}$, with the objective of deciding if there exists a partition $P = P_1 \cup P_2 \;:\; P_1 \cap P_2 = \emptyset$, with $\sum_{i \in P_1} x_i = \sum_{j \in P_2} x_j$. Let $M = \sum_i x_i$ be the sum of all integers in a given instance of partition. Then, we can construct an instance of Min-Max 2-MST in the plane with $2n$ pairs of points as follows. Place n point pairs $\{p_i, q_i\}_{i=1}^n$ along two ϵ-separated horizontal lines, such that p_i, q_i are vertically adjacent, with horizontal separation of M between consecutive pairs. Then, for each x_i in

the instance of PARTITION we place a point p_{n+i} at distance x_i from p_i, and its corresponding pair q_{n+i} at distance $\epsilon/2$ from q_i and p_i (See Figure 31).

Imagine building an MST on each side of the optimal partition for this instance. Notice that for pairs $\{p_i, q_i\}_{i=1}^{n}$, edges (p_i, p_{i+1}) and (q_i, q_{i+1}) will exist in the optimal solution for $1 \leq i \leq n-1$ and serve as "backbones" of the two MSTs. For each remaining point pair i, $n+1 \leq i \leq 2n$, as $\{p_i, q_i\}$ can't be assigned to the same tree, by definition, one tree will incur an "extra cost" of $\epsilon/2$ and the other will incur an "extra cost" of x_i. Therefore, any algorithm which minimizes the maximum weight of either tree also minimizes $max\{\sum_{i \in P_1} x_i, \sum_{j \in P_2} x_j\}$ across all partitions of $P = P_1 \cup P_2$.

Thus, by choosing ϵ small enough, we can guarantee that any proposed exact algorithm to Min-Max 2-MST will produce two trees, each of weight $(n-1)M + M/2 + o(1)$, if and only if there exists a feasible solution to the partition instance. □

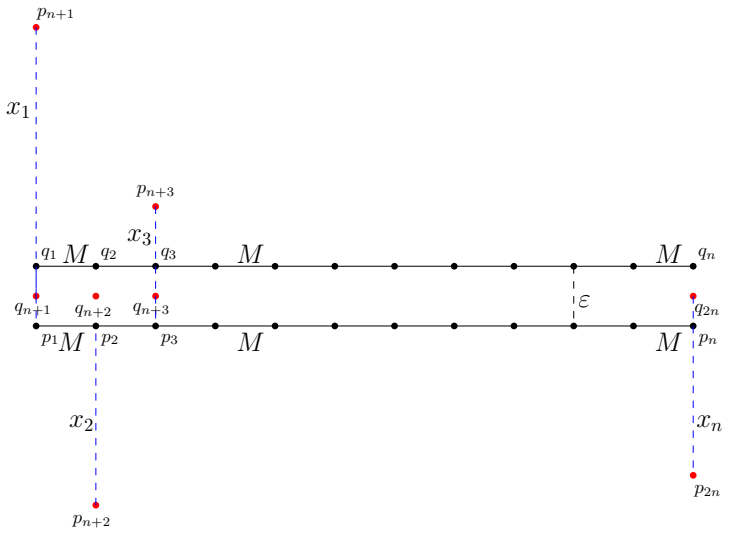


Figure 31: Set up of Min-Max 2-MST instance given an instance of PARTITION: $\{x_1, x_2, ..., x_n\}$.

Theorem 33. *There exists a 4α-approximation for the Min-Max 2-MST problem.*

Proof. We use the same algorithm as we did for the Min-Sum 2-MST problem (Section 6.2.1). The approximation factor is dominated by case 2 in the Min-Sum 2-MST analysis. For the Min-Max objective function, we have that

$max\{|MST(B)|, |MST(R)|\} \leq \alpha|MST(S)|$ and that $|MST(S)| \leq 4|OPT|$. Thus, $max\{|MST(B)|, |MST(R)|\} \leq 4\alpha|OPT|$. □

6.2.3 Bottleneck

In this section the objective is to $min - max\{|\lambda(S_1)|, |\lambda(S_2)|\}$.

Lemma 17. *Given n pairs of points on a line where consecutive points on the line are unit separated, there exists a partition of these points into two sets, S_1 and S_2, such that $max\{|\lambda(S_1)|, |\lambda(S_2)|\} \leq 3$.*

Proof. The proof will be constructive, using Algorithm 4. We partition the points on the line into n disjoint buckets, each bucket containing two consecutive points.

Algorithm 4 Coloring points on a line.

```
Color the leftmost point, p, red
Let p' be the point that is in p's bucket
Let R be a set of red points and B be a set of blue points
R ← {p}; B ← ∅
while There exists an uncolored point do
  while p' is uncolored do
    if p is red then
      Color p's pair, q, blue
      B ← B ∪ {q}
      p ← q
    else
      Let p'' be the point in p's bucket
      Color p'' red
      R ← R ∪ {p''}
      p ← p''
  Find the leftmost uncolored point x and color it red. Let x' be the point
  in x's bucket
  p ← x; p' ← x'
S1 ← R; S2 ← B
return {S1, S2}
```

Observe that at the end of Algorithm 4, each bucket has exactly one red point and one blue point. Thus, the maximum distance between any two points of the same color is 3. □

Theorem 34. *There exists a 3-approximation algorithm for the Bottleneck 2-MST problem on a line.*

Proof. Note that if the leftmost n points do not contain two points from the same pair, then it is optimal to let S_1 be the leftmost n points and S_2 be the rightmost n points. Suppose now that the leftmost n points contain two points from the same pair. We run algorithm 4 on the input. Imagine building two bottleneck spanning trees over the approximate partition (coloring) as well as over the optimal partition. Let λ be the longest edge (between two points of the same color) in our solution and λ^* be the longest edge in the optimal solution.

Consider any two consecutive input points s_i and s_{i+1} on the line. We first show that $|\lambda^*| \geq |s_i s_{i+1}|$ by arguing that the optimal solution must have an edge that covers the interval $[s_i, s_{i+1}]$. Suppose to the contrary that no such edge exists. This means that s_i is connected to $n-1$ points only to its left and s_{i+1} is connected to $n-1$ points only to its right. This contradicts the assumption that the leftmost n points contain two points from the same pair.

Let the longest edge in our solution be defined by two points, p_i and p_j. Consider the number of input points in interval $[p_i, p_j]$. Input points in this interval other than p_i and p_j will have a different color than p_i and p_j. It is easy to see that if $[p_i, p_j]$ consists of two input points, that $|\lambda^*| = |\lambda|$, and if $[p_i, p_j]$ consists of three input points, that $|\lambda^*| \geq |\lambda/2|$. We know by lemma 17 that $[p_i, p_j]$ can consist of no more than four input points. In this last case, $|\lambda^*|$ must be at least the length of the longest edge of the three edges in $[p_i, p_j]$. Thus, we see that $|\lambda^*| \geq |\lambda|/3$. □

Theorem 35. *There exists a 9-approximation algorithm for the Bottleneck 2-MST problem in a metric space.*

Proof. First, we compute $MST(S)$ and consider the heaviest edge, h. The removal of this edge induces a cut that separates the nodes into two sets, H_1 and H_2. If $\nexists\, i \ : p_i, q_i \in H_j$ for $1 \leq i \leq n$ and $1 \leq j \leq 2$, then we let $S_1 = H_1$ and $S_2 = H_2$ and return S_1 and S_2. Let λ^* be the heaviest edge in the bottleneck spanning trees built on the optimal partition. Note that $MST(S)$ lexicographically minimizes the weight of the kth heaviest edge, $1 \leq k \leq 2n-1$, among all spanning trees over S, and thus the weight of the heaviest edge in $MST(S) \setminus h$ is a lower bound on $|\lambda^*|$. Thus, in this case, our

solution is clearly feasible and is also optimal as $MST(S_1)$ and $MST(S_2)$ are subsets of $MST(S) \setminus h$.

Now suppose $\exists\, j \in \{1,2\} : p_i, q_i \in H_j, 1 \leq i \leq n$. This means that $|\lambda^*| \geq |h|$. In this case, we compute a bottleneck TSP tour on the entire point set. It is known that that a bottleneck TSP tour with bottleneck edge λ can be computed from $MST(S)$ so that $|\lambda| \leq 3|h| \leq 3|\lambda^*|$.

Next we run Algorithm 4 on the TSP tour and return two paths, each having the property that the largest edge has weight no larger than $9|\lambda^*|$. $\square$

Remark: Consider the problem of partitioning the point set into two sets, S_1 and S_2, such that $p_i \in S_1$ if and only if $q_i \in S_2$, and computing a bottleneck TSP tour on S_1 and on S_2. The objective is to minimize the heaviest edge (across both paths). Let the heaviest edge in the bottleneck TSP tours built on the optimal partition be λ^{**}. The above algorithm gives a 9-approximation to this problem as well because the algorithm returns two Hamilton paths and we know that (using the notation in the above proof) $|\lambda^*| \leq |\lambda^{**}|$. Thus, $|\lambda| \leq 9|\lambda^*| \leq 9|\lambda^{**}|$.

The following is a generalization of Lemma 17. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be a set of n k-tuples of points on a line and let S be the point set. Each set S_i, $1 \leq i \leq n$, must be colored with k colors. That is, no two points in set S_i can be of the same color. Consider two consecutive points of the same color, p and q. We show that there exists a polynomial time algorithm that colors the points in S so that the number of input points in interval (p, q) is $O(k)$.

Lemma 18. *There exists a polynomial time algorithm to color S so that for any two consecutive input points of the same color, p and q, the interval (p, q) contains at most $2k - 2$ input points.*

Proof. The algorithm consists of k steps, where in the jth step, we color n of the yet uncolored points with color j. We describe the first step.

Divide the kn points into n disjoint buckets, each of size k, where the first bucket B_1 consists of the k leftmost points, the second bucket B_2 consists of the points in places $k+1, k+2, \ldots, 2k$, etc. Let $G = (V, E)$ be the bipartite graph, with node set $V = \{\mathcal{S} \cup B = \{B_1, \ldots, B_n\}\}$, in which there is an edge between B_i and S_j if and only if at least one of S_j's points lies in bucket B_i. According to Hall's theorem [53], there exists a perfect matching in G. Let M be such a matching and for each edge $e = (B_i, S_j)$ in M, color one of the points in $B_i \cap S_j$ with color 1. Now, remove from each tuple the point that

was colored 1, and remove from each bucket the point that was colored 1. In the second step we color a single point in each bucket with the color 2, by again computing a perfect matching between the buckets (now of size $k-1$) and the $(k-1)$-tuples. It is now easy to see that for any two consecutive points of the same color, p and q, at most $2k-2$ points exist in interval (p, q). □

6.3 Matchings

Given a set $\mathcal{S}$ of n pairs of points in a metric space, partition the point set S into two sets, S_1 and S_2, such that for each pair $\{p_i, q_i\}$, $p_i \in S_1$ if and only if $q_i \in S_2$. Let $M(X)$ be the minimum weight matching on point set X and $|M(X)|$ be the cost of the matching. Let $\lambda(X)$ be the bottleneck edge in a matching on point set X and $|\lambda(X)|$ be the cost of the bottleneck edge.

6.3.1 Minimum Sum

The objective is to minimize $|M(S_1)| + |M(S_2)|$.

Theorem 36. *There exists a 2-approximation for the Min-Sum 2-Matching problem in general metric spaces.*

Proof. First, note that the weight of the minimum weight perfect matching on S, M^*, which excludes edges $(p_i, q_i)\ \forall\ i$ is a lower bound on $|OPT|$. Let $\hat{M}$ denote the minimum weight (exactly) one of a pair matching between point pairs in $\mathcal{S}$. That is, a minimum weight matching on the pairs of points where the weight of an edge between two pairs is equal to the weight of the cheapest edge (of four possible edges) that connects points between the two pairs. $|\hat{M}|$ is a lower bound on the weight of the smaller of the two matchings of OPT and therefore has weight at most $|OPT|/2$.

Our algorithm computes $\hat{M}$, and sets the points involved in this matching to be in S_1 and sets $S_2 = S \setminus S_1$. We return the partition S_1, S_2 as our approximate solution.

We have that $|M(S_1)| = |\hat{M}| \leq |OPT|/2$. To bound $|M(S_2)|$, consider the multigraph $G = (V = S, E = M^* \cup \hat{M})$. All $v \in S_2$ have degree 1 (from M^*), and all $u \in S_1$ have degree 2 (from M^* and $\hat{M}$). For each $v_i \in S_2$, either v_i is matched to $v_j \in S_2$ by M^*, or v_i is matched to $u_i \in S_1$ by M^*. In the former case we can consider v_i and v_j matched in S_2 and charge the weight of this edge to $|M^*|$. In the latter case, note that each $u \in S_1$ is part

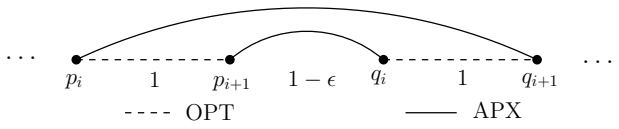


Figure 32: $\frac{|APX|}{|OPT|} \approx 2$

of a unique cycle, or a unique path. If $u \in S_1$ is part of a cycle then no vertex in that cycle belongs to S_2 due to the degree constraint. Thus, if $v_i \in S_2$ is matched to $u_i \in S_1$, u_i is part of a unique path whose other terminal vertex x belongs to S_2, due to the degree constraint. We can consider v_i, and x matched and charge the weight of this edge to the unique path connecting v_i and x in G. Thus, $|M(S_2)|$ can be charged to $|M^* \cup \hat{M}|$ and has weight at most $1.5|OPT|$.

Therefore, our partition guarantees $|M(S_1)| + |M(S_2)| \leq 2|OPT|$. Figure 32 shows the approximation factor using our algorithm is tight. □

6.3.2 Min-max

In this section the objective is to $min - max\{|M(S_1)|, |M(S_2)|\}$.

Theorem 37. *The Min-Max 2-Matching problem is weakly NP-hard.*

Proof. The reduction is from PARTITION: given a set $P = \{x_1, x_2, ..., x_n\}$ of n integers, decide if there exists a partition $P = P_1 \cup P_2 : P_1 \cap P_2 = \emptyset$, with $\sum_{i \in P_1} x_i = \sum_{j \in P_2} x_j$. Let $M = \sum_i x_i$. Given any instance P of PARTITION, we create a geometric instance of the Min-Max 2-Matching problem, as shown in Figure 33.

We place n point pairs $\{p_i, q_i\}_{i=1}^n$ along two ϵ-separated horizontal lines, such that p_i, q_i are vertically adjacent, with horizontal separation of M between consecutive pairs. Then, for each x_i in the instance of PARTITION we place a point p_{n+i} at distance x_i from p_i, and its corresponding pair q_{n+i} at distance $\epsilon/2$ from both q_i and p_i.

Notice that any solution which minimizes the weight of the larger of the two matchings created only uses edges between points of the same "cluster" $\{p_i, q_i, p_{n+i}, q_{n+i}\} \forall\, i$. Any edge between two clusters $\{p_i, q_i, p_{n+i}, q_{n+i}\}$, $\{p_j, q_j, p_{n+j}, q_{n+j}\}, i \neq j$ costs at least M and if matching edges are chosen

within clusters the entire matching can be constructed with cost at most $M + o(1)$ for $\epsilon > 0$ chosen small enough.

Within each cluster an assignment will have to be made, that is, WLOG, $\{p_i, p_{n+i}\} \in S_1, \{q_i, q_{n+i}\} \in S_2$ or $\{p_i, q_{n+i}\} \in S_1, \{q_i, p_{n+i}\} \in S_2$. Therefore, any algorithm that minimizes the maximum weight of either matching also minimizes $max\{\sum_{i \in P_1} x_i, \sum_{j \in P_2} x_j\}$ across all partitions of $P = P_1 \cup P_2$. Thus, for $\epsilon > 0$ chosen small enough the instance of partition is solvable if and only if the weight of the larger matching created is at most $\frac{M}{2} + o(1)$.

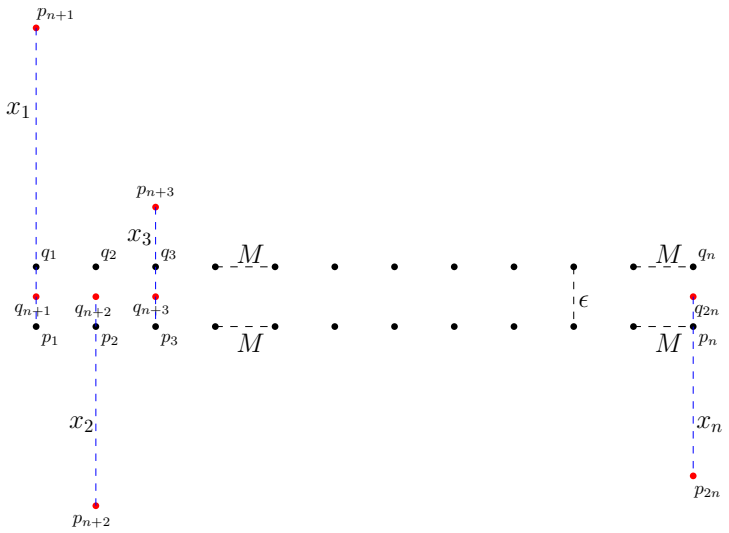


Figure 33: Set up of the Min-Max 2-Matching instance given an instance of PARTITION: $\{x_1, x_2, ..., x_n\}$.

□

Theorem 38. *The approximation algorithm for the Min-Sum 2-Matching problem serves as a 3-approximation for the Min-Max 2-Matching problem in general metric spaces.*

Proof. In this case we are concerned only with the larger of the two matchings returned by our approximation, $M(S_2)$, which, as described above, has weight bounded above by $|\hat{M}| + |M^*|$. However, in this case, under the new cost function $|\hat{M}| \leq |OPT|$, and $|M^*| \leq 2|OPT|$. Therefore, $|M(S_2)| \leq |\hat{M}| + |M^*| \leq 3|OPT|$. The example illustrated in Figure 32 shows the approximation factor achieved by this algorithm is tight. □

6.3.3 Bottleneck

In this section the objective is to $min-max\{|\lambda(S_1)|, |\lambda(S_2)|\}$. We begin with a lemma concerning the structure of a feasible solution. Let $S = S_1 \cup S_2$ be any feasible partition to the Bottleneck 2-Matching instance. Let $M_B(X)$ be a minimum bottleneck matching on point set X. Construct a graph $G = (V, E)$ where $V = S$ and $E = (M_B(S_1) \cup M_B(S_2) \cup (p_i, q_i)_{i=1}^n)$ is the union of any pair of optimal bottleneck matchings on S_1, and S_2 and the edges $(p_i, q_i) \forall\, i$.

Lemma 19. *G is a 2-factor such that each input pair is contained in exactly one cycle, and each cycle contains an even number of input pairs.*

Proof. The edge set of G is the union of two disjoint perfect matchings over S. Therefore, each node has degree exactly 2 and G is by definition a 2-factor. Also, by definition of a 2-factor, each input point p_i is part of a unique cycle, and in this case, as each node p_i has an edge of the form (p_i, q_i) incident to it, therefore, the point q_i must be contained in the same cycle as $p_i \forall\, i$. Thus, each input pair is contained in the same unique cycle in G.

To see that each cycle contains an even number of input pairs, imagine coloring the nodes of S_1 red and the nodes of S_2 blue. Suppose by contradiction that some cycle in G contains an odd number of input pairs. By definition, two nodes defining an edge of $M_B(S_1) \cup M_B(S_2)$ must be of the same color, and two nodes defining an edge of the form (p_i, q_i) must be of different color. Consider a cycle in G containing an odd number of input pairs. Coloring nodes of S_1 red and S_2 blue is equivalent, in this cycle, to contracting edges of the form (p_i, q_i) and 2-coloring the edges of $M_B(S_1) \cup M_B(S_2)$, as the edges of this cycle alternate between the form (p_i, q_i) and edges in $M_B(S_1) \cup M_B(S_2)$. Since this contracted cycle has an odd number of edges, this 2-coloring is not possible, thus a contradiction. □

Using the above structure lemma we will argue that we can compute a graph with the same properties and extract a feasible partition with constant factor approximation guarantees. Let $\hat{M}_B(\mathcal{S})$ be the minimum weight (exactly) one of a pair bottleneck matching over $\mathcal{S}$; that is, a bottleneck matching on the pairs of points where the cost of an edge between two pairs of points is the cheapest of the four edges between points in the two pairs. Note that edges of this matching connect points of S. Let $M_B(S)$ be the minimum weight bottleneck matching over S (excluding the edges $(p_i, q_i) \forall\, i$).

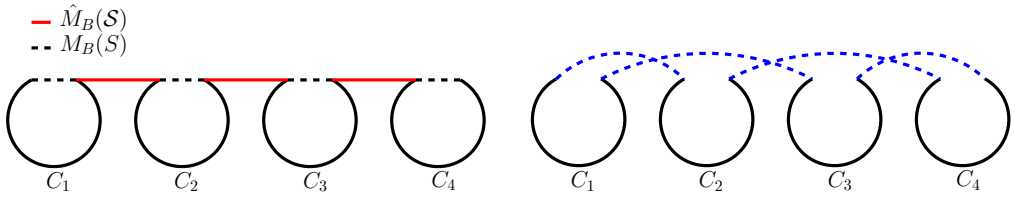
Let $\hat{\lambda}$ (resp. λ) be the heaviest edge used in $\hat{M}_B(\mathcal{S})$ (resp. $M_B(S)$) and let λ^* be the heaviest edge in a minimum weight bottleneck matching computed over each of the two sets in OPT. Note that $|\hat{\lambda}| \leq |\lambda^*|$ and $|\lambda| \leq |\lambda^*|$.

Begin by constructing a graph $G = (V = S, E = M_B(S) \cup (p_i, q_i)_{i=1}^n)$, which is a 2-factor as its edge set is the union of two disjoint perfect matchings. Note, it will be the case that each input pair exists in the same unique cycle. If each cycle contains an even number of input pairs then this graph has the same structure as that described in Lemma 19 and thus we can extract a feasible partition from G. We will describe how to obtain this partition later. As $|\lambda^*| \geq |\lambda|$, this graph induces an optimal partition. On the other hand, if there exists a cycle with an odd number of input pairs (there must be an even number of such cycles) then we "merge" cycles of G together into larger cycles until a point is reached in which each new "super-cycle" contains an even number of pairs. From this graph we can extract a constant factor approximation to an optimal partition.

Lemma 20. *If G contains at least one cycle with an odd number of input pairs, then it is possible to merge cycles of G into super-cycles, each of which contains an even number of input pairs, such that the heaviest edge (excluding $(p_i, q_i)\ \forall\ i$) in any super-cycle has weight at most $3|\lambda^*|$.*

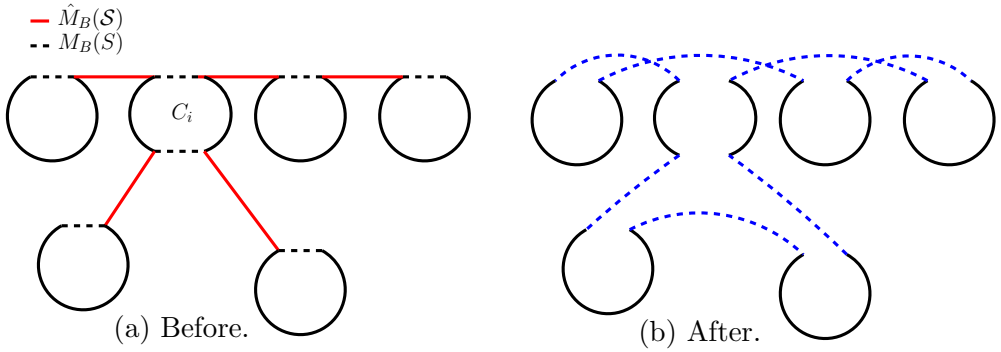
Proof. Superimpose a subset of the edges in $\hat{M}_B(\mathcal{S})$ over the nodes of G in the following way. Consider only edges in $\hat{M}_B(\mathcal{S})$ which have endpoints in different cycles. Treat each cycle in G as a node and run Kruskal's algorithm until all of the aforementioned edges of $\hat{M}_B(\mathcal{S})$ are exhausted. This yields a forrest on the cycles of G. It is easy to see that every cycle of G containing an odd number of input pairs has an edge of $\hat{M}_B(\mathcal{S})$ connecting it to some other cycle. This implies that it is possible to merge all cycles which are connected by an edge of $\hat{M}_B(\mathcal{S})$ until one reaches a point where all cycles have an even number of pairs. We give a brief outline for this merging process.

Find any maximal path P in G which alternates edges of the form $M_B(S)$ and $\hat{M}_B(\mathcal{S})$. Consider the cycles in G containing the edges of $M_B(S)$ in P (see Figure 34a). We will label the cycles in this path $C_1, C_2, \ldots, C_k$ where k is the number of cycles in the path. We will "stitch" the cycles together into a final super-cycle by making connections between pairs of cycles with odd subscripts in sorted order then by making connections between cycles with even subscripts in reverse sorted order. Now, remove all edges of P and we are left with a super-cycle (see Figure 34b). Recall that the weight of each edge of $M_B(S)$ and $\hat{M}_B(\mathcal{S})$ is a lower bound on $|\lambda^*|$. Consider the two nodes



(a) Before. (b) After.

Figure 34: Before and after stitching.



(a) Before. (b) After.

Figure 35: Before and after merging into a super-cycle.

that define some edge e created in the stitching process. Note that e can be replaced by path of at most three edges from $M_B(S) \cup \hat{M}_B(\mathcal{S})$. Thus, any edge not of the form (p_i, q_i) in the super-cycle has weight at most $3|\lambda^*|$. It is not difficult to see that the stitching process can be done while keeping the weight of the bottleneck edge at most $3|\lambda^*|$ regardless of whether k is even or odd.

It is possible that many maximal paths share an edge with the same cycle C_i in G (see Figure 35a). These edges must all be different because, when only considering edges of $M_B(S) \cup \hat{M}_B(\mathcal{S})$, the degree of each node in G is at most two. This implies that any edge created in one stitching process is not altered by another stitching process and thus stitching processes are independent of one another. All cycles associated with these paths will be merged into the same super-cycle (see Figure 35b), and since the merging processes are independent, the bottleneck edge created is still of weight no larger than $3|\lambda^*|$. □

Now that each cycle contains an even number of input pairs, all that is left to show is how to create a feasible partition from these cycles so that the

weight of the heaviest edge in the bottleneck matching computed on either side of the partition is at most $3|\lambda^*|$. Notice that for each cycle, every other edge is of the form (p_i, q_i). We will 2-color the nodes of each cycle red and blue so that if two nodes share an edge of the form (p_i, q_i), they must be of different color, and if they share an edge not of the form (p_i, q_i) they must be of the same color. Assign the red nodes to S_1 and the blue nodes to S_2. This partition is clearly feasible and the weight of the heaviest edge in the matchings created on either side is equal to the weight of the heaviest edge not of the form (p_i, q_i) among all of the cycles we have created; this edge has weight at most $3|\lambda^*|$.

Theorem 39. *There exists a 3-approximation to the Bottleneck 2-Matching problem in general metric spaces.*

6.4 TSP Tours

Given a set $\mathcal{S}$ of n pairs of points in a metric space, partition the point set S into two sets, S_1 and S_2, such that for each pair $\{p_i, q_i\}$, $p_i \in S_1$ if and only if $q_i \in S_2$. Let $TSP(X)$ be a TSP tour on point set X and $|TSP(X)|$ be the cost of the tour. Let $\lambda(X)$ be the bottleneck edge in a TSP tour on point set X and $|\lambda(X)|$ be the cost of the bottleneck edge.

In this section we consider partitioning a set of input pairs of points so that the cost function of a pair of TSP tours, one built on each set of the partition, is minimized. It is interesting to note the complexity difference emerging here. In prior sections, the structures to be computed on each set of the partition were computable exactly in polynomial time. Thus, the decision versions of these problems, which ask if there exists a partition such that some cost function over the pair of structures is at most k, are easily seen to be in NP. However, when the cost function is over a set of TSP tours or bottleneck TSP tours, this is no longer the case. That is, suppose that a non-deterministic Touring machine could in polynomial time, for a point set S and $k \in \mathbb{R}$, return a partition for which it claimed the cost of the TSP tours generated over both sets is at most k. Unless $P = NP$, the verifier cannot in polynomial time confirm that this is a valid solution, and therefore the problem is not in NP. Thus, the problems considered in this section are all NP-hard.

6.4.1 Minimum Sum

In this section the objective is to minimize $|TSP(S_1)| + |TSP(S_2)|$.

Algorithm 5 Algorithm $A(\mu, \beta)$. $0 < \mu < 1$ and $\beta > 1$.

Let a β-factor approximate TSP tour on set X be denoted $\overline{TSP}(X)$.

1. Compute $\overline{TSP}(S)$.
2. Let $2k$ be the maximum even number not exceeding $(2+\frac{1}{\mu})\beta$. Enumerate all ways of decomposing $\overline{TSP}(S)$ into $2k$ connected components: for each decomposition, assign the nodes from the components to S_1 and S_2 alternately (i.e. assign all nodes in component one to S_1, all nodes in component two to S_2, etc.). If this partition is infeasible, then skip to the next decomposition; otherwise compute $\overline{TSP}(S_1)$ and $\overline{TSP}(S_2)$.
3. Compute a random feasible partition, $S = \hat{S}_1 \cup \hat{S}_2$, and compute $\overline{TSP}(\hat{S}_1)$ and $\overline{TSP}(\hat{S}_2)$.
4. Among all pairs of tours produced in steps 2 and 3, choose the pair of minimum sum.

We will show for $\beta > 1$ and for the proper choice of μ, that Algorithm 5 gives a 3β-approximation for the Min-Sum 2-TSP problem. Fix a constant $\mu < 1$. Let OPT be the optimal partition $S = S_1^* \cup S_2^*$. Let $d(S_1, S_2)$ be the minimum point-wise distance between sets S_1 and S_2. We call an instance of the problem μ-separable if there exists a feasible partition $S = S_1 \cup S_2$: $d(S_1, S_2) \geq \mu(|TSP(S_1)| + |TSP(S_2)|)$.

Let APX be the partition returned by our approximation. We will show that if S is not μ-separable, then $|APX| \leq \frac{2}{1-4\mu}\beta|OPT|$ (see Lemma 21) and that if S is μ-separable, then $|APX| \leq \frac{1}{4\mu}\beta|OPT|$ (see Lemma 22). Supposing both of these are true, then the approximation factor of our algorithm is $max\{\frac{1}{4\mu}, \frac{2}{1-4\mu}\}\beta$. One can easily verify that $\mu = 1/12$ is the minimizer which gives the desired 3β factor. The following lemma states that if S is not μ-separable, then any feasible partition yields a "good" approximation.

Lemma 21. *If S is not μ-separable, then $|APX| \leq \frac{2}{1-4\mu}\beta|OPT|$.*

Proof. If S is not μ-separable, then for any feasible partition $S = S_1 \cup S_2$ we have $d(S_1, S_2) \leq \mu(|TSP(S_1)| + |TSP(S_2)|)$. In particular, for the partition induced by the optimal solution, $S = S_1^* \cup S_2^*$, $d(S_1^*, S_2^*) \leq \mu(|TSP(S_1^*)| + |TSP(S_2^*)|)$. Then, $|TSP(S)| \leq |OPT| + 2d(S_1^*, S_2^*) \leq |OPT| + 2\mu(|TSP(S_1^*)| + |TSP(S_2^*)|) \leq |OPT| + 4\mu|TSP(S)|$.

Hence, when $\mu < \frac{1}{4}$, $|TSP(S)| \leq \frac{1}{1-4\mu}|OPT|$. Let $S = \hat{S}_1 \cup \hat{S}_2$ be the random feasible partition computed by $\mathcal{A}(\mu, \beta)$. Then, as we are returning the best partition between $\hat{S}_1 \cup \hat{S}_2$ and all $O(n^{2k})$ partitions of $\overline{TSP}(S)$, we have $|APX| \leq \beta(|TSP(\hat{S}_1)| + |TSP(\hat{S}_2)|) \leq 2\beta|TSP(S)| \leq \frac{2\beta}{1-4\mu}|OPT|$. □

The following lemma states that if S is μ-separable, then any witness partition to the μ-separability of S gives a "good" approximation.

Lemma 22. *If S is μ-separable, then $|APX| \leq \frac{1}{4\mu}\beta|OPT|$.*

Proof. Suppose we successfully guessed a partition $X_0 = S_1^0 \cup S_2^0$ that is a "witness" to the μ-separability of S (we will show how to guess X_0 later).

- Case 1: $OPT = X_0$. Then $|APX| \leq \beta(|TSP(S_1^0)| + |TSP(S_2^0)|) = \beta(|TSP(S_1^*)| + |TSP(S_2^*)|) = \beta|OPT|$.
- Case 2: $OPT \neq X_0$. Then for $i = 1, 2$, $S_i^* \neq S_i^0$, which means each tour in OPT must contain at least 2 edges crossing the cut (S_1^0, S_2^0), hence the optimal solution must contain at least 4 edges crossing the cut (S_1^0, S_2^0). So $|OPT| \geq 4d(S_1^0, S_2^0) \geq 4\mu(|TSP(S_1^0)| + |TSP(S_2^0)|) \geq \frac{4\mu}{\beta}|APX|$. Equivalently, $|APX| \leq \frac{\beta}{4\mu}|OPT|$.

□

The next two lemmas show how to guess a witness partition X_0 in polynomial time. First, we show that if S is μ-separable with a witness partition X_0, then $\overline{TSP}(S)$ cannot cross this partition "too many" times.

Lemma 23. *Let $\overline{TSP}(S)$ be an β-factor approximation for $TSP(S)$. Also, suppose S is μ-separable with witness X_0. Then $\overline{TSP}(S)$ crosses the cut (S_1^0, S_2^0) at most $(2 + \frac{1}{\mu})\beta$ times.*

Proof. One can construct a TSP tour for S by adding two bridges to $TSP(S_1^0)$ and $TSP(S_2^0)$, thus we have $|TSP(S)| \leq |TSP(S_1^0)| + |TSP(S_2^0)| + 2d(S_1^0, S_2^0) \leq (2 + \frac{1}{\mu})d(S_1^0, S_2^0)$. Also, suppose $\overline{TSP}(S)$ crosses the cut (S_1^0, S_2^0) $2k$ times. Then, $2kd(S_1^0, S_2^0) \leq |\overline{TSP}(S)| \leq \beta|TSP(S)|$. Combining the above two inequalities, we obtain $2k \leq (2 + \frac{1}{\mu})\beta$. □

The next lemma completes our proof.

Lemma 24. *Suppose S is μ-separable. Let X_0 be any partition which serves as a "witness". Then, in step 2 of $\mathcal{A}(\mu, \beta)$, we will encounter X_0 at some stage.*

Proof. Given a nonnegative integer k and a TSP tour P, define $\Pi(P, k) = \{X$: X is a feasible partition and P crosses X at most k times$\}$. By Lemma 23, we know $X_0 \in \Pi(\overline{TSP}(S), (2 + \frac{1}{\mu})\beta)$. Since step 2 of $\mathcal{A}(\mu, \beta)$ is actually enumerating all partitions in $\Pi(\overline{TSP}(S), (2 + \frac{1}{\mu})\beta)$, we are done. □

Note that step 2 considers $O(n^{2k}) = O(n^{14\beta})$ decompositions and for each partition that is feasible, we compute two approximate TSP tours. Suppose the running time to compute a β-factor TSP tour on n points is $h_\beta(n)$. Then the worst case running time of Algorithm 5 is $O(h_\beta(2n)n^{14\beta})$. Thus, we have the following Theorem.

Theorem 40. *For any $\beta > 1$, the algorithm $\mathcal{A}(\frac{1}{12}, \beta)$ is a 3β-approximation for the Min-Sum 2-TSP problem with running time $O(h_\beta(2n)n^{14\beta})$.*

Remark: If S is in the Euclidean plane then $\beta = 1 + \epsilon$ for some $\epsilon > 0$ [73] yielding a $(3 + \epsilon)$-approximation and if S is in a general metric space then $\beta = 3/2$ [32] yielding a 4.5-approximation. In both cases $h_\beta(2n)$ is polynomial.

6.4.2 Min-max

In this section the objective is to $min - max\{|TSP(S_1)|, |TSP(S_2)|\}$.

Theorem 41. *There exists a 6β-approximation to the Min-Max 2-TSP problem, where β is the approximation factor for TSP in a certain metric space.*

Proof. We use the same algorithm, and return the same partition, $S_1 \cup S_2$, as in Section 6.4.1. Let APX be the partition returned, and let $|APX|$ be the cost of the larger of the two TSPs computed on APX. Let OPT be the optimal solution. Note that $|APX| \leq \beta(|TSP(S_1)| + |TSP(S_2)|) \leq 3\beta(|TSP(S_1^*)| + |TSP(S_2^*)|) \leq 6\beta|OPT|$ since $|TSP(X_1)| + |TSP(X_2)| \leq 2max\{TSP(X_1), TSP(X_2)\}$ for any $X = X_1 \cup X_2$.

□

6.4.3 Bottleneck

In this section the objective is to $min - max\{|\lambda(S_1)|, |\lambda(S_2)|\}$.

Theorem 42. *There exists an 18-approximation algorithm for the Bottleneck 2-TSP problem.*

Proof. We remarked in Section 6.2.3 that there exists a 9-approximation to the problem of finding a partition that minimizes the weight of the bottleneck edge on two Hamilton paths built on the partition. A Hamilton path can be converted into a Hamilton cycle by at most doubling the weight of the bottleneck edge in the Hamilton path. This yields an 18-approximation to the Bottleneck 2-TSP problem. □

Chapter 7

Conclusion

In this dissertation we considered many algorithmic problems with applications in sensor networks. We considered that the placement of an input set of sensors is in a general metric space and that the placement of the sensors is in Euclidean space. For many of these problems we either designed efficient exact polynomial-time algorithms or showed that the problems are NP-hard. For most of the problems that were shown to be NP-hard, we designed $O(1)$-approximation algorithms and very frequently achieved better constants when the sensors were assumed to be located in Euclidean space. We improved on some known results and introduced and solved many new important problems. This work, with a focus on geometric optimization, advances the field of sensor networks.

References

[1] National traveling salesman problems. `http://www.math.uwaterloo.ca/tsp/world/countries.html`. Accessed: 2014-12-15.

[2] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristn. Smallest color-spanning objects. In F. auf der Heide, editor, *Algorithms ESA 2001*, volume 2161 of *Lecture Notes in Computer Science*, pages 278–289. Springer Berlin Heidelberg, 2001.

[3] K. Almiani, A. Viglas, and L. Libman. Tour and path planning methods for efficient data gathering using mobile elements. *IJAHUC*, 21(1):11–25, 2016.

[4] G. Anastasi, M. Conti, and M. Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 1096–1102, July 2008.

[5] A. Archer, M. Bateni, M. Hajiaghayi, and H. J. Karloff. Improved approximation algorithms for prize-collecting steiner tree and TSP. *SIAM J. Comput.*, 40(2):309–332, 2011.

[6] C. Archetti, M. Speranza, and D. Vigo. Vehicle routing problems with profits. Technical report, Tech. Report WPDEM2013/3, University of Brescia, 2013.

[7] E. Arkin, J. S. B. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *In Symposium on Computational Geometry*, pages 307–316, 1998.

[8] E. M. Arkin, A. Banik, P. Carmi, G. Citovsky, M. J. Katz, J. S. B. Mitchell, and M. Simakov. Choice is hard. In *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 318–328, 2015.

[9] E. M. Arkin, A. Banik, P. Carmi, G. Citovsky, M. J. Katz, J. S. B. Mitchell, and M. Simakov. Conflict-free covering. In *Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015*, 2015.

[10] E. M. Arkin, J. M. Díaz-Báñez, F. Hurtado, P. Kumar, J. S. B. Mitchell, B. Palop, P. Pérez-Lantero, M. Saumell, and R. I. Silveira. Bichromatic 2-center of pairs of points. *Comput. Geom.*, 48(2):94–107, 2015.

[11] E. M. Arkin, M. M. Halldórsson, and R. Hassin. Approximating the tree and tour covers of a graph. *Information Processing Letters*, 47(6):275–282, 1993.

[12] E. M. Arkin and R. Hassin. Minimum-diameter covering problems. *Networks*, 36(3):147–155, 2000.

[13] E. M. Arkin, R. Hassin, and A. Levin. Approximations for minimum and min-max vehicle routing problems. *J. Algorithms*, 59(1):1–18, 2006.

[14] R. Aschner, G. Citovsky, and M. J. Katz. Exploiting geometry in the SINR _k model. In *Algorithms for Sensor Systems - 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2014, Wroclaw, Poland, September 12, 2014, Revised Selected Papers*, pages 125–135, 2014.

[15] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[16] G. Ausiello, V. Bonifaci, and L. Laura. The online prize-collecting traveling salesman problem. *Inf. Process. Lett.*, 107(6):199–204, Aug. 2008.

[17] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela. On salesmen, repairmen, spiders, and other traveling agents. In *Algorithms and Complexity, 4th Italian Conference, CIAC 2000, Rome, Italy, March 2000, Proceedings*, pages 1–16, 2000.

[18] C. Avin, Y. Emek, E. Kantor, Z. Lotker, D. Peleg, and L. Roditty. SINR diagrams: Convexity and its applications in wireless networks. *J. ACM*, 59(4):18, 2012.

[19] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *SIAM JOURNAL ON COMPUTING*, pages 277–283, 1995.

[20] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[21] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 166–174, 2004.

[22] L. Barba, S. Durocher, R. Fraser, F. Hurtado, S. Mehrabi, D. Mondal, J. Morrison, M. Skala, and M. A. Wahid. On k-enclosing objects in a coloured point set. In *Proceedings of the 25th Canadian Conference on Computational Geometry, CCCG 2013, Waterloo, Ontario, Canada, August 8-10*, 2013.

[23] B. K. Bhattacharya, A. Custic, A. Rafiey, A. Rafiey, and V. Sokol. Approximation algorithms for generalized MST and TSP in grid clusters. In *Combinatorial Optimization and Applications - 9th International Conference, COCOA 2015, Houston, TX, USA, December 18-20, 2015, Proceedings*, pages 110–125, 2015.

[24] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. P. Williamson. A note on the prize collecting traveling salesman problem. *Math. Program.*, 59:413–420, 1993.

[25] A. Biere, M. Heule, and H. van Maaren. *Handbook of satisfiability*, volume 185. IOS Press, 2009.

[26] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 46–55, Oct 2003.

[27] P. S. Bonsma, T. Epping, and W. Hochstättler. Complexity results on restricted instances of a paint shop problem for words. *Discrete Applied Mathematics*, 154(9):1335–1343, 2006.

[28] D. Chafekar, V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Approximation algorithms for computing capacity of wireless networks with SINR constraints. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint*

Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA, pages 1166–1174, 2008.

[29] T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003.

[30] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 661–670, 2008.

[31] K. Chen and S. Har-Peled. The euclidean orienteering problem revisited. *SIAM Journal on Computing*, 38(1):385–397, 2008.

[32] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[33] G. Citovsky, J. Gao, J. S. B. Mitchell, and J. Zeng. Exact and approximation algorithms for data mule scheduling in a sensor network. In *Algorithms for Sensor Systems - 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2015, Patras, Greece, September 17-18, 2015, Revised Selected Papers*, pages 57–70, 2015.

[34] M. Claverol, D. Garijo, C. I. Grima, A. Mrquez, and C. Seara. Stabbers of line segments in the plane. *Computational Geometry*, 44(5):303 – 318, 2011.

[35] M. Claverol Aguas et al. Problemas geométricos en morfología computacional. 2004.

[36] S. Coene and F. C. R. Spieksma. Profit-based latency problems on the line. *Oper. Res. Lett.*, 36(3):333–337, 2008.

[37] M. E. Consuegra and G. Narasimhan. Geometric avatar problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013*, volume 24 of *LIPIcs*, pages 389–400, 2013.

[38] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[39] S. Das, P. P. Goswami, and S. C. Nandy. Smallest color-spanning object revisited. *International Journal of Computational Geometry and Applications*, 19(05):457–478, 2009.

[40] G. A. Dirac. Some theorems on abstract graphs. *Proc. London Mathematical Society, series 3*, 2(1):69–81, 1952.

[41] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984.

[42] B. Eksioglu, A. V. Vural, and A. Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[43] G. Even, N. Garg, J. Knemann, R. Ravi, and A. Sinha. Covering graphs using trees and stars. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, volume 2764 of *Lecture Notes in Computer Science*, pages 24–35. 2003.

[44] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.

[45] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete . *Information Processing Letters*, 12(3):133 – 137, 1981.

[46] H. N. Gabow, S. N. Maheshwari, and L. J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, 2(3):227–231, 1976.

[47] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

[48] O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer. Complexity in geometric SINR. In *Proceedings of the 8th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2007, Montreal, Quebec, Canada, September 9-14, 2007*, pages 100–109, 2007.

[49] O. Goussevskaia, R. Wattenhofer, M. M. Halldórsson, and E. Welzl. Capacity of arbitrary wireless networks. In *INFOCOM 2009. 28th IEEE*

International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil, pages 1872–1880, 2009.

[50] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, August 2002.

[51] P. Gupta and P. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, Mar 2000.

[52] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[53] P. Hall. On representatives of subsets. *J. London Math. Soc*, 10(1):26–30, 1935.

[54] M. M. Halldórsson. Wireless scheduling with power control. *ACM Transactions on Algorithms*, 9(1):7, 2012.

[55] M. M. Halldórsson and P. Mitra. Wireless capacity with oblivious power in general metrics. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1538–1548, 2011.

[56] M. M. Halldórsson and R. Wattenhofer. Wireless communication is in APX. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 525–536, 2009.

[57] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics*, 45(6):615–627, 1998.

[58] O. Hudec. On alternative p-center problems. *Zeitschrift fur Operations Research*, 36(5):439–445, 1992.

[59] D. Ismailescu and J. Park. Improved upper bounds for the steiner ratio. *Discrete Optimization*, 11:22–30, 2014.

[60] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Proceedings of the First IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'05, pages 244–257, 2005.

[61] A. Kansal, M. H. Rahimi, D. Estrin, W. J. Kaiser, G. J. Pottie, and M. B. Srivastava. Controlled mobility for sustainable wireless sensor networks. In *Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, SECON 2004, October 4-7, 2004, Santa Clara, CA, USA*, pages 1–6, 2004.

[62] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[63] T. Kesselheim. A constant-factor approximation for wireless capacity maximization with power control in the SINR model. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1549–1559, 2011.

[64] M. R. Khani and M. R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69(2):443–460, 2014.

[65] P. Khanteimouri, A. Mohades, M. Abam, and M. Kazemi. Computing the smallest color-spanning axis-parallel square. In L. Cai, S.-W. Cheng, and T.-W. Lam, editors, *Algorithms and Computation*, volume 8283 of *Lecture Notes in Computer Science*, pages 634–643. Springer Berlin Heidelberg, 2013.

[66] D. Kim, R. Uma, B. Abay, W. Wu, W. Wang, and A. Tokuta. Minimum latency multiple data mule trajectory planning in wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 13(4):838–851, April 2014.

[67] D. Knuth. SAT11 and SAT11k. *Proceedings of SAT Competition 2013; Solver and Benchmark Descriptions*, page 32, 2013.

[68] D. E. Knuth. Nested satisfiability. *Acta Informatica*, 28(1):1–6, 1990.

[69] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345 – 358, 1992.

[70] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[71] M. Ma and Y. Yang. Data gathering in wireless sensor networks with mobile collectors. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–9. IEEE, 2008.

[72] G. L. Miller, S. Teng, W. P. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997.

[73] J. S. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[74] T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*, 2006.

[75] Y. Myung, C. Lee, and D. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995.

[76] V. Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012.

[77] O. Ore. Note on hamilton circuits. *The American Mathematical Monthly*, 67(1):p. 55, 1960.

[78] E. M. Palmer. The hidden algorithm of Ore's theorem on hamiltonian cycles. *Computers & Mathematics with Applications*, 34(11):113–119, 1997.

[79] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[80] P. C. Pop. New models of the generalized minimum spanning tree problem. *J. Math. Model. Algorithms*, 3(2):153–166, 2004.

[81] P. C. Pop, W. Kern, G. Still, and U. Faigle. Relaxation methods for the generalized minimum spanning tree problem. *Electronic Notes in Discrete Mathematics*, 8:76–79, 2001.

[82] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, pages 30–41, 2003.

[83] P. Slavik. Approximation algorithms for set cover and related problems. 1998.

[84] A. Somasundara, A. Kansal, D. Jea, D. Estrin, and M. Srivastava. Controllably mobile infrastructure for low energy embedded networks. *Mobile Computing, IEEE Transactions on*, 5(8):958–973, Aug 2006.

[85] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling with dynamic deadlines. *Mobile Computing, IEEE Transactions on*, 6(4):395–410, 2007.

[86] S. L. Tanimoto, A. Itai, and M. Rodeh. Some matching problems for bipartite graphs. *Journal of the ACM (JACM)*, 25(4):517–525, 1978.

[87] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

[88] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[89] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[90] Z. Vincze and R. Vida. Multi-hop wireless sensor networks with mobile sink. In *CoNEXT'05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 302–303, New York, NY, USA, 2005. ACM Press.

[91] P. Wan, X. Jia, and F. F. Yao. Maximum independent set of links under physical interference model. In *Wireless Algorithms, Systems, and Applications, 4th International Conference, WASA 2009, Boston, MA, USA, August 16-18, 2009. Proceedings*, pages 169–178, 2009.

[92] F. Wu and Y. Tseng. Energy-conserving data gathering by mobile mules in a spatially separated wireless sensor network. *Wireless Communications and Mobile Computing*, 13(15):1369–1385, 2013.

[93] W. Yu and Z. Liu. Vehicle routing problems with regular objective functions on a path. *Naval Research Logistics (NRL)*, 61(1):34–43, 2014.