Algorithms for Wireless Communication and Sensor Networks

Thesis submitted for the degree of

Doctor of Philosophy

at University of Leicester

by

Thomas Grant

Department of Computer Science

University of Leicester

October 2012

Algorithms for Wireless Communication and Sensor Networks
Thomas Grant

Abstract

In this thesis we will address four problems concerned with algorithmic issues that arise from communication and sensor networks.

The problem of *scheduling wireless transmissions under SINR constraints* has received much attention for unicast (one to one) transmissions. We consider the scheduling problem for multicast requests of one sender to many receivers, and present a logarithmic approximation algorithm and an online lower bound for arbitrary power assignments.

We study the problem of *maximising the lifetime of a sensor network for fault-tolerant target coverage* in a setting with composite events, where a composite event is the simultaneous occurrence of one or more atomic events. We are the first to study this variation of the problem from a theoretical perspective, where each event must be covered twice and there are several event types, and we present a $(6 + \varepsilon)$-approximation algorithm for the problem.

The *online strongly connected dominating set* problem concerns the construction of a dominating set that is strongly connected at all times, and for every vertex not in the dominating set, there exists an edge to some vertex in the dominating set, and an edge from a vertex in the dominating set. We present a lower bound for deterministic online algorithms and present an algorithm that achieves competitive ratio matching the lower bound.

The *monotone barrier resilience* problem is to determine how many sensors must be removed from a sensor network, such that a monotone path can exist between two points that does not intersect any sensor. We present a polynomial time algorithm that can determine the monotone barrier resilience for sensor networks of convex pseudo-disks of equal width.

Acknowledgements

My main thanks go to Thomas Erlebach. I am forever indebted to Thomas for the opportunity of working with someone with an exceptional talent for research. He has not only been an exemplary supervisor, but has also been a pleasure to work with. He has been incredibly generous with his advice and guidance regarding research, and his unwavering moral support has been invaluable. Thomas is an irreplaceable asset, not only to the department and the university, but also to the research community as a whole. Although his overriding contribution to making my life a better one will be his supervision of my Ph.D, I also greatly treasure the memories when we were able to socialise outside of work. In particular I am very fond of our various lunches/dinners, and travelling together to Bergen for a workshop.

My time as a Ph.D student within the department of Computer Science has been very enjoyable due to the friendly atmosphere within the department. In particular, I would like to especially thank Michael Hoffmann, who has always offered sound advice as my second supervisor, has continually shown an interest in my work and been very supportive throughout the past few years. Similarly, Rick Thomas has always been very approachable and provided me with invaluable support and guidance regarding the teaching aspects of academia. Additionally, I consider myself very fortunate that during the course of my Ph.D there have been so many friendly colleagues, especially Kyriakos, Michalis, Mop, Sam and Stelios who I consider not just colleagues, but also close friends. I also wish to thank those who have collaborated with me: Frank Kammer, Torsten Tholey and Marco Chiesa, who were not only a pleasure to work with and host, but also contributed greatly to my development as a researcher.

Finally, I wish to express my gratitude to my family and friends, especially my parents to who I owe everything I have in life, my brother, and my girlfriend and our children, who have given me a sense of purpose in life, and the promise of a better tomorrow.

Contents

List of Figures

Chapter 1

Introduction

The recurring theme throughout this work is the theoretical analysis of algorithms for communication and sensor networks. Although the work within the individual chapters is to a certain degree disjoint, all the work presented here could be considered under the umbrella research theme of the theoretical analysis of approximation and exact algorithms, designed for solving issues that arise in sensor networks, and wireless ad-hoc communication networks.

A natural modelling of communication networks, in particular wireless ad-hoc networks is to consider the nodes forming the network to be represented by vertices of a graph, and then a natural analogy of the links between communicating nodes, is to represent links as edges in the graph. This observation is the motivation for our interest in real life problems arising in wireless communication and sensor networks, that can be modelled by algorithmic problems in graphs, of which we consider the theoretical analysis.

We consider in this work communication networks. A *communication network* is a collection of nodes, that wish to transmit data to each other. Such transmissions are accomplished via communication links. A communication link may be of various physical forms, e.g., wired links, fibre optic links, wireless transmissions. It may also be that such a network is comprised of a variety of the aforementioned media types. The interested reader can refer to [LGW04] for an overview of communication networks. Throughout this thesis we will consider wireless ad-hoc networks.

A wireless ad-hoc network is extremely versatile in its nature, as it is not dependent on a pre-existing infrastructure. There are however drawbacks in such networks. A common necessity of wireless ad-hoc networks is to specify a routing mechanism for the network. Such routing can be achieved by a variety of means, either by centralised governance, where all nodes in the network adhere to a specified routing organisation, or by a distributed organisation where the nodes act somewhat independently of each other. Throughout this thesis, we will be discussing routing problems of a centralised nature. For a more comprehensive description of routing problems, the reader can refer to [AKK04].

Additionally, one often considers ad-hoc networks that consist of mobile nodes that utilise a battery as a power source, and as such the nodes have limited power available. Hence, efficient routing is of paramount importance. Transmitting wireless signals will deplete power at the sending node, and consequently one wishes to avoid unnecessary transmissions with the aim of preserving energy available for the node for future transmissions. Using power unnecessarily could impair the efficiency of the network, and as such the organisation of how packets are routed through the network, is of great concern in practice. Routing can also be more of a problematic issue in wireless networks utilising batteries as a power source, as due to a limited amount of power available to the nodes, the transmission radius of the nodes is limited. This leads to an increased need for routing in such networks as most transmissions will require the use of intermediate nodes to transmit the packet over a greater distance. Routing through multiple intermediate nodes is commonly referred to as *multi-hop* routing [AY05].

There are a variety of algorithmic issues related to the routing of packets in wireless networks, two of which we discuss in this work, namely scheduling wireless transmissions such that they do not interfere with each other, and forming a routing backbone for transmissions (in Chapters 3 and 5 respectively). One can specify such a routing backbone of the network that selects a group of nodes forming the network, that are specified as nodes that will be utilised in routing packets through the network to the destination node. We discuss the concept in greater detail in Chapter 5.

A *sensor network* is a collection of nodes (predominantly such networks will be comprised of wireless nodes) that are deployed to detect events. There is an inherent need for fault tolerance in sensor networks. Deployed sensors may not be able to detect events that they are supposed to monitor due to several factors. Such sensor networks are often comprised of small, battery powered nodes and as such a sensor could fail due to a lack of power. Such a lack of power could occur due to the battery being completely discharged so the sensor cannot function, or due to reduced power available due to battery depletion leading to a reduced sensing radius, and as a result an event that should have been monitored might not detected. Additionally, failure in sensor networks may occur due to other factors such as physical damage to the nodes in the network, which could be common in networks deployed in military settings, or environmental interference preventing the sensing or communication of a node due to extreme weather phenomena.

For examples of such settings where failure to detect an event that the network is supposed to monitor is unacceptable, one can consider settings where a sensor network is deployed to detect intruders in a building or other secure area, or alternatively environmental warning systems where sensor networks are deployed to detect potential extreme weather events and warn nearby inhabited areas. We refer the reader to the survey paper of Callaway [CJ03] for more information regarding sensor networks.

A common approach to provide fault tolerance within a sensor network is to introduce redundancy. One possible example of an approach in this area would be to duplicate each sensor, i.e., for every sensor deployed in the network, deploy a second sensor in close proximity to its duplicate, such that if one of the sensors fails, the other may remain active and is able to detect and report the occurrence of an event. We will address different notions of fault tolerance in sensor networks in Chapters 4 and 6.

Outline of this Thesis

The remainder of this thesis is organised as follows:

Chapter 2. In the second chapter fundamental concepts related to the results presented in this thesis are introduced. Within this chapter we also present an introduction of terminology that is used continuously throughout the duration of the thesis.

Chapter 3. In the third chapter the first study is presented, namely the problem of scheduling wireless transmissions under Signal-to-Interference-plus-Noise Ratio (SINR) constraints. While previous work has considered the unicast case, where each transmission has one sender and one receiver, we consider the setting of multicast requests where each transmission has one sender and a set of receivers. A set of multicast transmissions can be scheduled in the same round if the SINR at all receivers is above a certain threshold. The goal is to minimise the number of rounds. Building on the relationship between SINR scheduling and unit disks colouring established by Halldórsson [Hal09], we present an $\mathcal{O}(\log \Gamma)$-approximation algorithm for multicast scheduling in the SINR model, where Γ is the ratio of the longest to the shortest link length, considering only the longest link of each multicast request. The algorithm uses uniform power assignment and can be implemented online. We also compare the model of atomic multicasts, where all receivers of a multicast must receive the transmission in the same round, to the model of splittable multicasts, where a multicast sender can transmit in several rounds, each time serving a subset of its receivers. Furthermore, we consider the throughput maximisation problem where one has to schedule the maximum number of requests when restricted to a single round and obtain an $\mathcal{O}(\log \Gamma)$-competitive randomised online algorithm, and finally we show that every deterministic online algorithm, even for unicast links and using arbitrary power assignments, has competitive ratio $\Omega(\log \Gamma)$.

Chapter 4. In the fourth chapter, we study the problem of maximising the lifetime of a sensor network for fault-tolerant target coverage in a setting with composite events. Here, a composite event is the simultaneous occurrence of a combination of atomic events, such as the detection of smoke and high temperature. We are given sensor nodes that have an initial battery level and can monitor certain event types, and a set of points at which composite events need to be detected. The points and sensor nodes are located in the Euclidean plane, and all nodes have the same sensing radius. The goal is to compute a longest activity schedule with the property that at any point in time, each event point is monitored by at least two active sensor nodes. We present a $(6 + \varepsilon)$-approximation algorithm for this problem by devising an approximation algorithm with the same ratio for the dual problem of minimising the weight of a fault-tolerant sensor cover. The algorithm generalises previous approximation algorithms for geometric set cover with weighted unit disks and is obtained by enumerating properties of the optimal solution that guide a dynamic programming approach.

Chapter 5. We proceed in the fifth chapter, and present work which is concerned with online algorithms. We study the online problem of computing an in-out dominating set that is strongly connected (every vertex in the in-out dominating set has a path to every other vertex in the in-out dominating set), where an in-out dominating set has the property that for every vertex not in the in-out dominating set, there exists an edge to some vertex in the in-out dominating set, and an edge from some vertex in the in-out dominating set. This problem is analogous to the network routing problem of specifying a routing backbone for a communication network. We first present a lower bound for the problem stating that no deterministic online algorithm can attain a competitive ratio less than $n/4 + 1/4$, where n is the number of vertices in the graph. We then complement this lower bound, by presenting an online algorithm that attains the competitive ratio $n/4 + 1/4$ for the problem.

Chapter 6. In the penultimate chapter of this work we return our focus to offline algorithms for sensor networks. Consider two points in the plane. If one does not wish to allow anyone to travel between them undetected, then one can deploy a sensor network which forms a barrier between the two points. A path is said to be detected by a sensor in the network if the path intersects the sensor region. A sensor network for which all possible paths between two points are detected is said to attain *barrier coverage.* However, there have been numerous works which note that if one sensor fails, there may exist a path which is not detected. As such, this prompts the study of robustness of such sensor networks attaining barrier coverage. Two such measures of robustness are considered namely the *thickness* of a barrier, where the aim is to calculate the minimum number of times any path from the start point to the target point intersects the sensors in the network, and the *resilience* of the barrier, where the aim is to calculate the minimum number of sensors that must be removed from the network, such that a path exists between the two points that does not intersect any sensor. Previous work has shown the problem of calculating the barrier resilience of a network to be $\mathcal{NP}$-hard for non-symmetrical sensor regions, which has prompted other works to consider approximation algorithms. We study the barrier resilience problem when restricted to monotone paths (monotone barrier resilience), and show that we can calculate the monotone barrier resilience of a sensor network in polynomial time, for networks with sensor regions of convex shape and equal width.

Chapter 7. In the seventh and final chapter we will conclude this thesis. We will give a brief overview and summary of the main results and contributions presented in the aforementioned chapters in this thesis. We will also recapitulate some of the open questions and directions for future research related to the work presented in this thesis.

Chapter 3 is based on publication [EG10] and is joint work with Thomas Erlebach. Chapter 4 is based on publications [EGK11a, EGK11b] and is joint work with Thomas Erlebach and Frank Kammer. Chapter 5 is based on joint work with Thomas Erlebach and Chapter 6 is based on joint work with Thomas Erlebach and Torsten Tholey.

Chapter 2

Preliminaries

Although the reader is likely to be familiar with some of the concepts discussed in this chapter we nevertheless include brief introductions and summaries of some fundamental aspects of complexity theory and graph theory.

2.1 Complexity Theory

The vast area of complexity theory is one we cannot hope to begin to summarise within the scope of this thesis. Indeed, it is one of the most fundamental research areas within computer science, and thus has attracted much attention from researchers. As such, there are a plethora of interesting results one could discuss. Therefore, in this section we will only present a brief introduction and summation of some of the most pertinent areas of complexity theory relevant to the work presented in this thesis. However, we do assume the reader has some familiarity, and a grasp of the basic concepts of set theory.

We first discuss decision problems and then subsequently will proceed to discuss some of the most fundamental complexity classes. A *decision problem* is a problem for which an algorithm for any given instance of the problem must determine and output a solution of either *'yes'* or *'no'*. An instance for which the solution is yes is denoted as a *yes-instance*, and similarly an instance for which the solution is no is denoted as a *no-instance*. One of the most famous decision problems is the *halting problem*, that is the problem of taking as input an algorithm and an input to that algorithm, and determining whether or not the algorithm *'halts'* (i.e., terminates) at some stage, i.e., determine whether or not the algorithm will run for an infinite amount of time. An algorithm for the halting problem should therefore output 'yes' if the algorithm does terminate, or conversely output 'no' if the algorithm does not terminate. The halting problem was shown to have no solution by Church [Chu36] and Turing [Tur36] independently.

One is interested in making algorithms efficient in terms of computation time, and in some application settings such efficient algorithms may be essential. For example, consider an air traffic control system that must dynamically re-direct aircraft safely to avoid mid-air collisions. In such a scenario an algorithm which is inefficient would be unacceptable as it could endanger lives. One class of measures of the efficiency of an algorithm are the *computational resources* that are required by the algorithm during the run time of the algorithm.

An important factor when determining the computational resources used by an algorithm is the *size* of an input to an algorithm. For an algorithm designed to sort a list of numbers in ascending order, the size of the input would be the length of the list of numbers. Typically n will denote the input size to an algorithm. Two types of computational resources are usually considered when analysing algorithms: The *time complexity* of the algorithm which is the number of steps than an algorithm requires before terminating, and the *space complexity* of an algorithm, which is the amount of memory utilised by an algorithm during execution. We first discuss time complexity.

An algorithm is said to run in *polynomial time* if given an input of size n to some problem, the algorithm terminates after at most $\mathcal{O}(n^c)$ steps, where c is some constant value, i.e., the algorithm terminates after at most a polynomial number of steps. Similarly, if an algorithm has running time $\mathcal{O}(c^{f(n)})$ where c is a constant and $f(n)$ is some polynomial function of n, then the algorithm terminates in *exponential time* (assuming the algorithm does indeed terminate).

A problem P is said to lie in the *complexity class* $\mathcal{P}$, if there exists a deterministic algorithm that solves P, and the algorithm terminates within polynomial time. $\mathcal{P}$ is also referred to as $DTIME(n^{\mathcal{O}(1)})$ in the literature.

A problem P is in the complexity class *'non-deterministic polynomial time'* ($\mathcal{NP}$), if a solution for P can be *verified* to be correct in polynomial time. For example, for the problem of sorting a list of integers into ascending order one can easily check in polynomial time if the resulting list is indeed in ascending order. It is clear that the set of problems that can be solved in polynomial time is a subset of the set of problems whose solution can be verified in polynomial time, i.e., $\mathcal{P} \subseteq \mathcal{NP}$. Note that an equivalent definition of the complexity class $\mathcal{NP}$ is that it is constituted by all the problems that can be solved non-deterministically in polynomial time.

We can also determine the relationships between computational problems. One form of relationship is to note that problems can be *reduced* to other problems. A problem P' *reduces* to a problem P if for any instance of P', there exists a mapping to an instance of P, such that the instance of P' is a yes-instance if and only if the instance of P is a yes-instance. It also follows from this definition that it must be the case that the instance of P' is a no-instance if and only if the instance of P is a no-instance. If such a reduction can be performed in at most polynomial time then it is a *polynomial reduction.*

A decision problem P is $\mathcal{NP}$*-hard* if every other decision problem $P' \in \mathcal{NP}$ can be polynomially reduced to P.

A problem P is $\mathcal{NP}$*-complete*, if P is in $\mathcal{NP}$ and there exists a polynomial reduction from every other problem $P' \in \mathcal{NP}$ to P. The first problem shown to be $\mathcal{NP}$-complete was the *Boolean Satisfiability* problem by Cook in 1971 [Coo71]. In 1972, Karp [Kar72] built on this work and showed 21 $\mathcal{NP}$-complete problems. Many of the problems shown were fundamental problems in Computer Science (such as the *set cover* problem), which provides motivation for the study of the complexity of combinatorial problems.

If one problem P is known to be $\mathcal{NP}$-complete then there exists a polynomial reduction from every other $\mathcal{NP}$ problem to P. As a result of this, if one could show a polynomial time algorithm for P, then as any instance of another problem in $\mathcal{NP}$ can be polynomially transformed into an instance of P, there would exist a polynomial time algorithm for all problems in $\mathcal{NP}$. Consequently all problems in $\mathcal{NP}$ would also belong to $\mathcal{P}$, and this would show that $\mathcal{P} = \mathcal{NP}$.

As a final remark on the subject, it is widely conjectured that $\mathcal{P} \neq \mathcal{NP}$. We proceed under the assumption that $\mathcal{P} \neq \mathcal{NP}$ throughout the course of this thesis.

Although we do not explicitly analyse the space complexity of the algorithms presented in this thesis, we do provide a brief overview. The space complexity of an algorithm is the measure of the maximum amount of memory (space) that an algorithm will use during its runtime execution. If for a problem P, an algorithm for P uses at most a polynomial amount of space (with regard to the input size) to solve P then P is in the complexity class *PSPACE*.

For more information regarding complexity theory we refer the reader to [GJ79].

2.2 Offline Algorithms

An offline *optimisation problem* is a problem where one must find the best possible solution, amongst all feasible solutions. An optimisation problem is defined by the set of *instances* I to the problem, a function that will for a given instance $i \in I$ of the problem, map i to a set of *feasible solutions*, an *objective function* that will map each pair of an instance and a feasible solution to the non-negative cost of that solution, and an aim of the problem that is usually to maximise or minimise the value of the objective function (we discuss this in more detail in subsection 2.2.1). An offline algorithm will have complete knowledge of the instance with which to make its decisions (in contrast to online algorithms that are discussed later). We remark that for every optimisation problem, there is a corresponding decision problem. The consequence of this is that every optimisation problem is at least as hard as its corresponding decision problem.

If a problem is $\mathcal{NP}$-complete then we assume that it cannot be solved in polynomial time (following from the assumption $\mathcal{P} \neq \mathcal{NP}$). One approach to obtaining solutions to such problems is to have an efficient algorithm that does not guarantee the best possible (i.e., optimal) solution, but rather outputs an approximate solution that is not optimal, but rather is close to the optimal. Such algorithms that return approximate solutions are *approximation algorithms*. It is usually implicit that approximation algorithms terminate in polynomial time as the motivation for their use is that they run efficiently (i.e., have a small execution time) and approximate the solution.

2.2.1 Approximation Algorithms

For the analysis of approximation algorithms it is common that *worst case analysis* is employed. One compares the value of the output that the algorithm achieves for the worst possible instance in terms of the ratio of value of the solution output by the algorithm and the value of an optimal solution for the same instance. This provides a rigorous analysis as one then has the property that for any possible instance given to the algorithm, the algorithm will output a solution with cost at most (assuming a minimisation problem) a certain factor worse than an optimal solution for that instance, yielding a provable performance guarantee of the algorithm. Indeed we utilise worst case analysis throughout this thesis to yield performance guarantees for our algorithms.

As we briefly mentioned in this section the aim of an algorithm is usually to either minimise or maximise the value of the objective function. A *minimisation problem* is an optimisation problem where one wishes to minimise the value of the objective function. An algorithm for a minimisation problem obtains *approximation ratio* ρ if for any feasible instance i of the problem it outputs a solution with value at most a factor of ρ larger than an optimal solution:

$$alg(i) \leq \rho \cdot opt(i)$$

where $alg(i)$ denotes the objective function value of the algorithm, for the instance i, and similarly $opt(i)$ denotes the objective function value of an optimal solution, for the instance i. It is common in the literature to refer to the value of the objective function for a minimisation problem as the *cost* of the solution.

Similarly, a *maximisation problem* is an optimisation problem where the aim is to maximise the value of the objective function. An algorithm for a maximisation problem obtains approximation ratio ρ if for any feasible instance i of the problem, the algorithm outputs a solution with value at most a factor of ρ smaller than an optimal:

$$alg(i) \geq opt(i)/\rho$$

Analogous to minimisation problems, it is common in the literature for the objective function for a maximisation problem to be referred to as the *profit* of the solution.

One can also consider randomised algorithms for a minimisation problem where the expected cost of the algorithm (denoted by $\mathbb{E}[alg(i)]$ for the instance i) is analysed over all possible instances. Formally, a randomised algorithm has expected approximation ratio ρ if its expected outcome for any instance i is at most a factor of ρ larger than the cost of an optimal algorithm for the same instance:

$$\mathbb{E}[alg(i)] \leq \rho \cdot opt(i)$$

Similarly, for a maximisation problem, a randomised algorithm has expected approximation ratio ρ if for any instance i of a maximisation problem its expected outcome is at least a factor of ρ smaller than the profit of an optimal algorithm for the same instance:

$$\mathbb{E}[alg(i)] \geq opt(i)/\rho$$

A *constant-factor* approximation algorithm is an algorithm that has approximation ratio c, for some positive constant $c \in \mathbb{R}$. If for a problem P there exists an algorithm that is a polynomial-time constant-factor approximation algorithm, P belongs to the complexity class $\mathcal{APX}$ which is the set of all problems, for which there exists a polynomial-time constant-factor approximation algorithm.

A *polynomial-time approximation scheme* (PTAS) is a set of approximation algorithms that are given as input an instance of the problem, and a constant $\varepsilon > 1$, and that output a solution with approximation ratio $1 + \varepsilon$, and terminate in $\mathcal{O}(n^{f(1/\varepsilon)})$ time, where $f(1/\varepsilon)$ is some function depending on $1/\varepsilon$. Similarly, a *fully polynomial-time approximation scheme* (FPTAS) is a set of approximation algorithms that attain approximation ratio $1 + \varepsilon$, and have running time $\mathcal{O}(n^c \cdot (1/\varepsilon)^c)$ where c is some constant value.

An *exact algorithm* for an optimisation problem P is an algorithm that can determine an optimal solution for P, i.e., has approximation ratio 1. Alternate but equivalent terminology that appears commonly in the literature is that the algorithm *solves* the problem P.

A standard reference for the a reader interested in approximation algorithms is [Vaz04].

2.3 Online Algorithms

In many application settings one does not have complete information of a problem instance. For example, real time systems (such as stock trading algorithms) do not have knowledge of future inputs (price changes). In such a setting an algorithm must make decisions while being aware of only the input that has been received thus far, without knowledge of what future input it will receive (if any). A problem where the instance is revealed to an algorithm piece-by-piece and the algorithm must dynamically act when presented with a portion of the input, is an *online problem.*

In contrast to offline problems where the instance is represented as a set, and the algorithm is aware of the entire set before its execution, for an online problem, an *online algorithm* will receive an instance as an ordered set (sequence) and the algorithm only becomes aware of the input as each element of the sequence is presented to the algorithm piecewise. Additionally, in most online settings (including all the online problems discussed in this thesis) the online algorithm will also not know the length of the input sequence.

Competitive analysis is a recognised and commonly used approach for the analysis of online algorithms, introduced by Sleator and Tarjan [ST85] for the analysis of paging algorithms. One can can determine the quality (or *competitiveness*) of a solution produced by an online algorithm, with respect to an optimal solution, that could be constructed by an omnipotent optimal offline algorithm, that has the input in its entirety with which to inform its decisions, in contrast to the online algorithm that has only information about the current request and previous requests with which to make decisions.

A deterministic online algorithm *alg* is *ρ-competitive* (or has *competitive ratio ρ*) if for every instance i of a minimisation problem, the quality of the solution produced by *alg* is within a factor of ρ of an optimal offline algorithm (*opt*), with some additive constant c. Formally we have the following:

$$alg(i) \leq \rho \cdot opt(i) + c$$

where $alg(i), opt(i)$ denote the cost of the online algorithm and offline algorithm respectively, for the instance i. Similarly, for an online maximisation problem, an online algorithm has competitive ratio ρ if for any instance i, it produces an output of

$$alg(i) \geq opt(i)/\rho - c$$

Additionally, analogous to offline algorithms a randomised online algorithm *alg* has expected competitive ratio ρ if for any instance i of a minimisation problem, it satisfies:

$$\mathbb{E}[alg(i)] \leq \rho \cdot opt(i) + c$$

where $\mathbb{E}[alg(i)]$ denotes the expected output of the randomised online algorithm *alg* for an instance i. Similarly, for any instance i of a maximisation problem if the randomised online algorithm satisfies:

$$\mathbb{E}[alg(i)] \geq opt(i)/\rho - c$$

then the randomised online algorithm attains expected competitive ratio ρ.

For all four of the above definitions of competitive ratio, if c is less than or equal to 0, then the online algorithm *alg* is *strictly ρ-competitive*.

We remark for the reader a subtle point that although the concepts of competitive analysis and approximation ratio are similar, the definition of an online algorithm does not specify that it must terminate in polynomial time. Consequently one cannot directly compare the performance of an online algorithm with an offline algorithm. Within, the scope of this thesis the online algorithms presented do indeed terminate in polynomial time. Obviously, in practice one is always interested in efficient algorithms.

The interested reader can refer to [BEY98] for an in depth introduction and overview of online algorithms.

2.4 Graph Theory

All the problems presented in this work are graph problems motivated by real life issues arising in communication and sensor networks. As such, graph theory is a predominant theme throughout our work. To alleviate congestion and avoid repetition in the following chapters we will now present concepts, notation and terminology that are not particular to any one chapter, but are relevant in numerous sections of this thesis.

Throughout the following, we will assume some rudimentary knowledge of set theory and its fundamental operators. A *graph* G is defined to be a pair, constituted by a set of *vertices* V, and a set of *edges* E, i.e., $G = (V, E)$. V is the set of $n = |V|$ vertices in the graph. The notion of adjacency with regard to vertices in a graph is represented by the set E of edges. There are various different concepts of adjacency in graphs, two of which we will discuss in this section as they are both prevalent in our work. A graph can be viewed differently if one considers sets of symmetric edges, or asymmetric edges.

2.4.1 Undirected Graph

A graph $G = (V, E)$ where the set of edges E is symmetric, i.e., all pairs of edges in E are symmetric, is said to be an *undirected graph.* A set of symmetric edges, is formed by pairs $\{u, v\}$ of vertices $u, v \in V$, where $\{u, v\} = \{v, u\}$. For an edge $\{u, v\}$, the vertices u and v are *adjacent* to each other. Additionally, for an edge $\{u, v\}$, u and v are *endpoints* of the edge. Vertices that are adjacent can be referred to as *neighbours*, and the number of neighbours a vertex v has is the *degree* of that vertex. The *degree of a graph* G, is the maximum of the degree of all vertices $v \in V$.

A *path* within a graph is a sequence of vertices $(v_1, \ldots, v_k)$, such that there exists an edge between a vertex and the immediately following vertex in the sequence, i.e., there exists an edge $\{v_i,\ v_{i+1}\}$ for all vertices $v_1, \ldots, v_{k-1}$.

A graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if all vertices in G' appear in G, i.e., $V' \subseteq V$, and all edges in G' are in G, i.e., $E' \subseteq E$. A subgraph G' of G with vertex set $V' \subseteq V$ is *connected* if for every pair of vertices $u, v \in V'$, there exists a path from u to v in G'. A graph G is *connected*, if for every pair of vertices $u, v \in V$, there exists a path from u to v in G. Additionally, if for a graph G, for every vertex $v \in V$, there exists an edge $\{v, u\}$ for all $u \in V \setminus \{v\}$, then G is a *clique*. Note that in the literature a clique is also known as a *complete graph*.

2.4.2 Directed Graph

A *directed graph* is formed by a set of asymmetric edges, i.e., all pairs in E are asymmetric. A set of asymmetric edges, is formed by pairs (u, v) of vertices $u, v \in V$, where the pair $(u, v) \neq (v, u)$.

Note that in an undirected graph the notion of adjacency is unambiguous, i.e., for an edge $u, v \in E$, u and v are neighbours, however, in a directed graph for an edge (u, v) this concept is not clear. In a directed graph the vertices adjacent to a vertex v may not be equal to the vertices adjacent from a vertex v. For an edge (u, v), let u be *adjacent to* v, and v *adjacent from* u. The number of vertices adjacent to a vertex v is the *in-degree* of v, and similarly the number of vertices adjacent from v is the *out-degree* of v.

A *directed path* is a sequence of vertices $(v_1, \ldots, v_k)$, such that there exists the directed edges $(v_i,\ v_{i+1})$ for all vertices $v_1, \ldots, v_{k-1}$.

A subgraph G' of G with vertex set $V' \subseteq V$ is *strongly connected*, if for every vertex $v \in V'$, there exists a directed path from v to every other vertex $v' \in V' \setminus \{v\}$ in G'. Similarly, a directed graph is strongly connected if the subgraph of all vertices V and all edges E is strongly connected.

2.4.3 Graph Problems

Graph colouring is a fundamental graph optimisation problem. There are two main classes of colouring problems for graphs. Firstly, the aim of *vertex colouring* is to assign to each vertex a colour, such that adjacent vertices do not have the same colour assignment. The second main variant of graph colouring is the *edge colouring* problem. For edge colouring the aim is to assign colours to all edges of the graph, such that for all vertices, no vertex is an endpoint of two different edges that have the same colour assignment. For both problems, edge and vertex colouring, the objective function is to minimise the colours used for the assignment. The optimal (minimum) number of colours needed to colour the vertices of a graph G is known as the *chromatic number* of G. Similarly, the optimal (minimum) number of colours needed to colour the edges of a graph is known as the *chromatic index.* The decision version of the problem of computing the chromatic number of a graph was shown to be $\mathcal{NP}$-complete as one of Karp's 21 problems [Kar72].

An *independent set* (equivalently known as a *stable set*) is a subset of vertices such that for every pair of vertices in the independent set, the pair of vertices are not adjacent, i.e., at most one endpoint of every edge in the graph appears in the independent set. An independent set is a *maximal independent set* if it is not a proper subset of another independent set of the graph. Another equivalent definition of a maximal independent set is to consider the subset of vertices not in the independent set. If every one of these vertices has an edge to a vertex in the independent set, then the independent set is maximal. The *maximum independent set problem* is to find an independent set of maximum size, amongst all independent sets. The decision version of the maximum independent set problem is also $\mathcal{NP}$-complete [GJ79].

The *node weighted Steiner tree* problem is defined as the following: Given as input is an undirected graph $G = (V, E)$, where each vertex $v \in V$ is associated with a non-negative integer weight, and a subset of vertices $T \subseteq V$ referred to as *terminals*. A *Steiner tree* for T in G is a subset of vertices $V' \subseteq V$ such that $T \subseteq V'$ and V' induces a connected subgraph. The objective of the problem is to specify a Steiner tree such that the sum of the weights of the vertices in the Steiner tree is minimised amongst all possible Steiner trees for T in G. We will refer to the node weighted Steiner tree problem in Chapter 4.

For a more comprehensive overview of graph theory we refer the reader to [Die05] or [Wes01].

2.5 Linear Programming

In Chapter 4 we utilise *linear programming* (equivalently known as *linear optimisation*) as a subroutine of our approach. The aim of a *linear program*

(LP) is to maximise or minimise a linear objective function $c^T \cdot x$, where c is an n-dimensional column vector of constants, and x is an n-dimensional column vector or variables, and c^T denotes the transpose of the vector c. The variables must satisfy the *constraints* of the program, which take the form of linear inequalities $Ax \leq b$, where A is an $m \times n$ matrix, and b is an m-dimensional vector, and constraints that prevent non-negative variables $x \geq 0$.

A *feasible solution* is a solution for which all constraints of the linear program are satisfied. If such a solution maximises or minimises the value of the objective function over all possible solutions, then the solution is optimal. If the number of constraints is polynomial in the input size, a linear program can be solved in polynomial time [Sch98]. However, if the number of the constraints of a linear program is exponential, the linear program can be solved in polynomial time under the condition that there exists a *separation oracle* that can determine in polynomial time if any of the linear constraints are violated.

Integer linear programming (ILP) can be viewed as a constrained version of linear programming. An integer linear program is a linear program with the additional constraint that every variable must be an integer. Contrary to linear programs, integer linear programs cannot be solved in polynomial time unless $\mathcal{P} = \mathcal{NP}$. We do however note than an often used technique for optimisation problems is to formulate the problem as an integer linear program. One can then relax the integer constraints of the ILP, and obtain a resulting linear program. This linear program can then be solved optimally in polynomial time (under the condition that the number of constraints is polynomial), yielding a fractional solution. It then remains to perform rounding to obtain an integral solution from the optimal fractional solution.

For more information on the topic on linear and integer linear programming the reader, we refer the reader to [Sch98].

2.6 Geometry

In Chapter 3 concerned with scheduling wireless transmissions under SINR constraints, Chapter 4 on target coverage and Chapter 6 on barrier resilience, we consider problems embedded in a plane. We note for the reader that in all three studies, we only consider the two-dimensional Euclidean plane and as such, do not discuss preliminaries for spaces of greater dimension.

Let the two-dimensional Euclidean plane be represented by the set $\mathbb{R}^2$. Each element $p \in \mathbb{R}^2$ represents an individual point in the plane. Each point $p = (x_p, y_p)$ is identified as a pair of co-ordinates, where x_p represents the x-coordinate of p, and similarly let y_p be the y-coordinate of p. We denote the Euclidean distance between two points p and p' in the plane by $\delta(p, p')$.

2.6.1 Unit Disk Graphs

A natural class of graphs with which to model wireless networks is that of *unit disk graphs.* Unit disk graphs feature prominently in our work (in Chapters 3 and 4) and we introduce them here.

Consider a set of disks, all of the same radius, placed in the two-dimensional Euclidean plane. Two disks d and d' intersect if and only if there exists a point of intersection between the boundaries of d and d' (i.e., there exists a point p that appears on both the boundary of d and d'). In the resulting unit disk graph $G = (V, E)$, for every disk $d \in D$, there exists a corresponding vertex $v \in V$, and there is an edge $(v, v') \in E$ between two vertices v and v', if their corresponding disks d and d' intersect.

Chapter 3

Scheduling With Interference

In this chapter we consider the problem of scheduling multicast (one sender to many receivers) wireless transmissions under Signal-to-Interference-plus-Noise Ratio (SINR) constraints.

An important aspect of a wireless transmission schedule is the power that is assigned to the nodes for their transmission. The simplest approach is to utilise a *uniform power assignment*, where all nodes are assigned the same transmission power. Uniform power assignments may be favoured in practice due to their simpler implementation. Additionally, uniform power assignments must be used in homogeneous networks, where power control functionality is not available due to hardware constraints. Other power assignments that have been considered in previous literature are *oblivious* power assignments, where the power assigned to the sender of a transmission, is a function of the distance to the receiver, and *arbitrary* power assignment where the power assigned to a sender can be set arbitrarily without restriction (for example, depending on the interference caused by the simultaneous transmissions by other senders).

In this work we consider the SINR which is the ratio of the reception strength at the intended receiver (calculated by the power of a transmission, divided by the length between the sender and receiver of a link to a path loss exponent) to the reception strength of transmissions from other senders transmitting concurrently, plus ambient noise. SINR constraints stipulate that all intended receivers have SINR ratio above a pre-determined threshold. We define this formally later in Section 3.3.1.

The majority of work on transmission scheduling under SINR constraints, especially that preceding our work, has considered *unicast transmissions*, i.e., the case where each transmission has a single sender, and a single receiver. However, a fundamental property of wireless transmissions is that a single transmission can be received by several nodes that are within the transmission range of the sender. Furthermore, there are many scenarios where nodes forming the network may want to transmit the same message to a set of other nodes, e.g., in the exchange of routing information between neighbours in a virtual topology maintained on top of the physical network, or in the flooding of information across the network. Therefore, in this work we consider the wireless transmission scheduling problem for *multicast transmissions*.

Our aim is to investigate how existing methods developed for the unicast setting can be adapted to the multicast setting. We consider algorithms that use uniform power assignment. Although the lack of power control has been shown to be sub-optimal when compared with a schedule utilising power control by a factor logarithmic in the maximum power used [ALP09, Hal09], we argue that the study of uniform power schedules is indeed meaningful. The predominant reason for this is that solutions based on uniform power are simpler for implementation, both from the perspective of hardware and software. Thus, such methods are more likely to be adopted and utilised by practitioners. Similar justifications also appear in the literature [ALP09, Hal09]. Furthermore, to achieve stronger results, we measure the approximation ratio of our algorithms in comparison with optimal solutions with arbitrary power assignments.

We consider the scheduling and throughput maximisation of multicast transmission requests. For the scheduling problem one has to assign all the links into rounds, such that the SINR ratio at all receivers is above the desired threshold. For the throughput maximisation problem one has to chose a maximum subset of links for a single round, such that the SINR ratio at all receivers is above the desired threshold.

For both the scheduling and throughput problems, the unicast case was proven $\mathcal{NP}$-complete by Goussevskaia et al. [GOW07] for uniform power. As the unicast case is a special case of the multicast case, the multicast variant is also $\mathcal{NP}$-complete for both scheduling and throughput, under uniform power assignment. Furthermore, Andrews and Dinitz [AD09] show that even with arbitrary power assignment, the throughput problem is $\mathcal{NP}$-hard. These results provide motivation for studying approximation algorithms for these problems.

3.1 Related Work

A starting point for the theoretical analysis of the capacity of wireless networks was the seminal work by Gupta and Kumar [GK00], who studied the throughput problem in a setting when the nodes are distributed uniformly at random. There then followed an extensive study of offline scheduling and throughput problems in the SINR model with respect to arbitrary networks, with particular focus on worst-case analysis and approximation algorithms, including but not limited to [Mos07, MW06, AD09, FKV09, HW09, Hal09, GHWW09, ALP09].

The uniform power assignment has also attracted attention motivated by its practical applications, in addition to its theoretical interest. In the case where the optimal algorithm is also restricted to uniform power, algorithms achieving constant competitive ratio have been shown in [HW09, GHWW09].

The difference between uniform and non-uniform power assignments was initially investigated by Moscibroda and Wattenhofer [MW06].

Halldórsson [Hal09] studies the wireless scheduling problem in comparison with an optimal solution that is permitted arbitrary power assignment. He shows that the scheduling problem can be related to the colouring of unit disk graphs. In the case of links with similar lengths, it is shown that one can bound the loss in the approximation ratio to a constant factor. This leads to simple online algorithms utilising uniform power assignment that achieve competitive ratio $\mathcal{O}(1)$ for similar link lengths and $\mathcal{O}(\log \Gamma)$ for the general case where Γ is defined to be the ratio between the maximum and the minimum link lengths. Additionally, in this paper a lower bound on any algorithm using oblivious power assignment is presented. Specifically, no algorithm under oblivious power assignment can achieve a better approximation ratio than $\Omega(\log\log\Gamma)$ in the worst case. Furthermore, these results hold for the throughput problem.

For the throughput maximisation problem in fading metrics, where the optimal algorithm is allowed arbitrary power assignment, the first constant factor approximation algorithm was presented by Kesselheim [Kes11]. The same algorithm can be applied to general metrics with approximation ratio $\mathcal{O}(\log n)$.

Broadcast transmissions can be seen as an extreme case of multicast transmissions, in which one sender must transmit to all other nodes in the network, and have been studied in [RS09, GMW08].

Furthermore, the online variant of the throughput problem has also received attention by Fanghänel et al. [FGHV10]. They assume that requests can have duration in the interval $[1, T]$ and consider the problem in Euclidean space of constant dimension d. They show that no deterministic algorithm, utilising an oblivious power assignment, can achieve competitive ratio better than $\Omega(T \cdot \Gamma^{d/2})$. Furthermore, they present an $\mathcal{O}(T \cdot \Gamma^{(d/2)+\varepsilon})$-competitive deterministic online algorithm, and a randomised $\mathcal{O}(\log\Gamma \cdot \log T)$-competitive online algorithm. They also consider a generalisation of the problem, where requests have to be assigned to one of k channels.

Work has also been conducted on other problems related to the SINR model of interference. In addition to the challenges of power assignment and scheduling of transmissions, the *cross layer latency minimisation problem* proposed by Chafekar et al. [CKM$^+$07], also requires an algorithm to decide routing paths. Work under SINR constraints has also been performed for topology control by Gao et al. [GHN08] and Moscibroda et al. [MWZ06], and in sensor networks by Moscibroda [Mos07].

3.2 Contributions

We consider the scheduling problem for multicast requests under SINR constraints, assuming that the nodes are points in the two-dimensional Euclidean plane and the received signal strength is proportional to the power of the transmitted signal, divided by the distance between the sender and the intended receiver to the power of α, where α is the path loss exponent and assumed to be strictly greater than 2. In practice, it is typically assumed that α is between 2 and 6).

We consider multicast requests to be *atomic*, which is to say that in a given multicast group, all receivers must successfully receive the transmission in the same round. Building on Halldórsson's [Hal09] results for the unicast case, we present an $\mathcal{O}(\log \Gamma)$-approximation algorithm for the multicast scheduling under SINR constraints. In the multicast case, Γ denotes the ratio of the longest to the shortest link length, considering only the longest link of each multicast request. We remark that Γ can be much smaller than the ratio of the longest to the shortest link length, amongst all unicast requests that are part of a multicast request. As our algorithm is based on partitioning the requests into length classes, and application of unit disk graph colouring to schedule each length class, it can also be used as a simple online algorithm.

Additionally, we discuss the relationship between schedules for such atomic multicasts, and schedules for *splittable* multicasts, where the sender of a multicast can transmit in several rounds, serving a subset of its receivers in each round.

As with previous work, we show that our approach is also applicable to the case of throughput maximisation. Moreover, we present a lower bound showing that every deterministic online algorithm has competitive ratio $\Omega(\log \Gamma)$, even for the unicast variant of the problem. This complements previous lower bound constructions, where the algorithm was restricted to oblivious power assignments.

3.3 Preliminaries

Consider the two-dimensional Euclidean plane within which all senders and receivers are represented with points. We denote the Euclidean distance between any two such points p, q as $\delta(p, q)$. For an undirected graph $G = (V, E)$, $\Delta(G)$ denotes the maximum degree of any vertex in V. It is well known that any graph G can be coloured with $\Delta(G) + 1$ colours utilising a greedy colouring algorithm.

A graph $G = (V, E)$ is a *unit disk graph* (for disks with radius r) if each vertex $v \in V$ can be associated with a disk of radius r centred at point p_v in the plane such that two vertices $u, v \in V$ are adjacent if and only if the two corresponding disks intersect (or equivalently if the distance between p_u and p_v is at most $2r$). We state the following well known property of unit disks graphs which we utilise later in the analysis:

Lemma 3.1. *If a unit disk graph with disks of radius r has maximum degree $k \geq 1$, increasing the radius of all disks from r to cr for some $c > 1$ increases the maximum degree of the graph to at most $\mathcal{O}(kc^2)$.*

Proof. Consider a set of disks embedded in the plane of radius r, with a corresponding disk graph of maximum degree k. If one increases the radius of each disk to cr then we observe the following. Consider a single disk d. All neighbours of d must have centre within distance $2 \cdot cr$ to intersect d. Consider a grid overlaying d consisting of squares with side length r. The disk d can be covered by $(4 \cdot cr)^2/r^2 = 16c^2$ such squares. Note that for each $r \times r$ square, it must be that case that at most $k+1$ disks have centre in that square, or it would violate the condition that the unit disk graph for disks of radius r has maximum degree k. As such, the maximum degree of the unit disk graph when the radius of each disk is increased to cr is at most $(k+1) \cdot 16c^2$ and the claim follows. $\square$

3.3.1 Unicast Requests

In the following subsections we present notation for unicast and multicast requests and proceed to formally define the problems we are studying. As we extend previous work, in particular [Hal09], we strive to keep similar notation where possible for clarity.

A unicast request is a transmission request ℓ_v from a single sender s_v to a single sender r_v. As is common practice in related literature, we also refer to requests as *links*.

We override notation where the context is unambiguous and denote the *length* of a link ℓ_v by $\ell_v = \delta(s_v, r_v)$. Additionally for brevity of notation we also use δ_{uv} to refer to the distance $\delta(s_u, r_v)$ between the sender s_u of a link ℓ_u and the receiver r_v of another link ℓ_v. A set L of links is *nearly equilength* if the lengths of all links in L are within a factor of 2 of each other. Specifically when discussing nearly equilength links there exists some value D such that the lengths of all links lie in the interval $[D, 2D]$.

For a set L of nearly equilength unicast links with lengths in $[D, 2D]$, analogous to [Hal09], $G'_q(L)$ denotes the unit disk graph formed by disks with radius $qD/2$ and with unicast receivers as the centres of the disks.

For a set L of unicast links, Γ denotes the ratio of the maximum link length to the minimum link length, specifically $\Gamma = \max_v \ell_v / \min_u \ell_u$.

Let the power assigned to a sender s_v be $\mathcal{P}_v$. We assume however throughout this work that all senders utilise uniform power and that there is no upper bound on the power that can be assigned.

To model the degradation of a transmission over the distance from the sender, let α represent the path loss exponent and we adopt the common assumption made in previous literature that $2 < \alpha$ (e.g. [GHWW09]). The received signal strength of a transmission on link ℓ_v from s_v to r_v is $\mathcal{P}_v/\ell_v^\alpha$.

The model of interference we use is the SINR model, interchangeably referred to as the *physical model*. Let $\mathcal{S}_v$ be the set of senders that are transmitting concurrently with a sender s_v and let N represent the ambient background noise in the network. A unicast transmission ℓ_u from s_u to r_u can be received if the following constraint (referred to as the SINR constraint) is satisfied:

$$\frac{\mathcal{P}_u/\ell_u^\alpha}{N + \sum_{s_v \in \mathcal{S}_u \setminus \{s_u\}} \mathcal{P}_v/\delta_{vu}^\alpha} > \beta \tag{3.1}$$

$\beta \geq 1$ denotes the minimum SINR required for successful reception of a transmission. As is common in work preceding this study we assume that the background ambient noise $N = 0$. This assumption can be justified by the observation that the effect of the noise can be made arbitrarily small by scaling up the power assigned to all senders. Moreover, one can assume that $\beta = 1$, as via Lemma 3.9 any desired constant $\beta > 1$ can be achieved while losing only a constant factor in the approximation ratio.

3.3.2 Multicast Requests

A *multicast request* or *multicast group* is a set of unicast links with a common sender. A multicast group m_v is represented as a pair (s_v, R_v) where s_v is the sender and $R_v = \{r_{v_1}, r_{v_2}, \ldots, r_{v_k}\}$ is a set of $k \geq 1$ receivers. Intuitively a multicast request m_v asks for a single transmission by the sender s_v that is successfully received by all receivers in R_v simultaneously. For $1 \leq i \leq k$ we use ℓ_{v_i} to refer to the link with sender s_v and receiver r_{v_i}. Again, we override notation and also denote by ℓ_{v_i} the length of that link. Without loss of generality, we assume that the receiver with index 1 is a receiver that is furthest from the sender of m_v, i.e., $\ell_{v_1} = \max_{1 \leq i \leq k} \ell_{v_i}$. The distance between the sender of the multicast group m_u and a given receiver r_{v_i} in multicast group m_v is denoted by $\delta_{uv_i} = \delta(s_u, r_{v_i})$.

For a set M of multicast groups, Γ denotes the ratio of the maximum link length to the minimum link length amongst all longest links of the multicast groups in M, i.e., $\Gamma = \max_v \ell_{v_1} / \min_u \ell_{u_1}$.

Analogous to the unicast setting we also define multicast groups as being *nearly equilength* if the lengths of the longest link in each group are within a factor of 2 of each other, i.e., there exists a D such that $\ell_{v_1} \in [D, 2D]$ for all $m_v \in M$.

A multicast transmission from s_u to R_u is received successfully if inequality (3.1) holds for all $r_{u_i} \in R_u$.

A *schedule* for a set M of multicast links is a partitioning of M into subsets, referred to as *rounds* or *slots*. For each round we require that all transmissions are received successfully by all intended receivers.

Following from Halldórsson [Hal09] the *affectance* on a receiver r_v of a unicast link ℓ_v is defined to be the ratio of the interference received from concurrent transmissions by other senders to the received signal strength at r_v from s_v. Specifically, the affectance $a_L(\ell_v)$ on a unicast link ℓ_v from a set of links L is:

$$a_L(\ell_v) = \sum_{\ell_u \in L \setminus \{\ell_v\}} \frac{P_u / \delta_{uv}^\alpha}{P_v / \ell_v^\alpha}$$

In the multicast context the affectance on a receiver r_{v_i} in a multicast group m_v is the ratio of the interference received from concurrent transmissions by other senders to the received signal strength at r_{v_i} from s_v. Note that a receiver r_{v_i} successfully receives a transmission from s_v if and only if the affectance of concurrent transmissions on r_{v_i} is at most $1/\beta$.

For $p \geq 1$ a *p-signal schedule* is one for which the affectance of any multicast request is at most $1/p$, i.e., the SINR at every receiver is at least p. Additionally a *p-signal slot* (interchangeably referred to as a *p-signal set*) refers to one round of a p-signal schedule.

3.3.3 Problem Definitions

Work presented in this chapter is concerned with the two following optimisation problems. The multicast scheduling problem which we denote by *M-Scheduling*, is to compute, for a given set M of multicast requests, and a given $p \geq 1$, a p-signal schedule with a minimum number of rounds. The multicast throughput problem, denoted by *M-Throughput*, is to compute, for a given set M of multicast requests, a largest subset of M that forms a p-signal slot. The corresponding problems where the input consists only of unicast requests are denoted by *Scheduling* and *Throughput* respectively.

In the online versions of these problems introduced above, the requests are presented to the algorithm one by one, and the algorithm must process each request without knowledge of future requests. In the scheduling problems, this means that the algorithm must assign a round and a power to the request, and in the throughput problem, the algorithm must accept or reject the request, and in the case that a request is accepted, the algorithm must additionally assign a power to the sender. In both problems, decisions of the algorithm are irrevocable, and the solution must be feasible at all times. In the online variant, we compare the quality of the solution produced by an algorithm with the quality of an optimal offline solution for the same input.

Even though our algorithms utilise uniform power assignment, we compare their solutions with an optimal solution that can use arbitrary power assignments which strengthens our approximation results. For a given set L of unicast or multicast requests, we denote by $opt_p(L)$ the optimal solution. For convenience where it does not cause ambiguity, when discussing the scheduling problem, we overload notation and also use $opt_p(L)$ to refer to the length of the optimal schedule. In the case that $p = 1$, we also write opt as shorthand for opt_p.

It is assumed that the multicast requests are atomic, i.e., the sender of a multicast transmission can only send the signal once and all its receivers must successfully receive the transmission in the same round. However, one can also consider the variant of the scheduling problem with *splittable* multicast requests. In this variant, the sender of a multicast request can transmit in several rounds, and each of its receivers must successfully receive the transmission in at least one of these rounds. It is clear that the optimal splittable schedule length cannot be longer than the optimal atomic schedule.

3.4 Algorithm for Multicast Scheduling

In this section we will present an $\mathcal{O}(\log \Gamma)$-approximation algorithm for the problem of scheduling multicast requests under SINR constraints (*M-Scheduling*). We follow the approach of Halldórsson [Hal09] and show that a constant-factor approximation for nearly equilength multicast groups can be achieved by a greedy colouring of a suitably defined unit disk graph. The difficulty we are presented with is that a set of nearly equilength multicast groups may contain unicast links that are much shorter than the longest links of the multicast groups, and hence it does not suffice to argue about nearly equilength unicast links. We will proceed by first recapitulating some results shown by Halldórrson [Hal09] that we require.

As introduced in [Hal09] as a measure of separation between a pair of links in relation to their length, two unicast links ℓ_u, ℓ_v are said to be q-*independent* for some constant q if and only if they satisfy the following constraint:

$$\delta_{uv} \cdot \delta_{vu} \geq q^2 \cdot \ell_v \ell_u \tag{3.2}$$

A set L of links is said to be q-independent if any pairs of links in the set is q-independent. For a given set L of unicast requests, the *link graph* $G_q(L)$ is a graph with a vertex for each request in L, and an edge between two vertices if the corresponding requests are not q-independent.

Allow us to recall that $G'_q(L)$ denotes the unit disk graph of a link set L with a disk of radius $qD/2$ centred at each receiver r_v of a link $\ell_v \in L$.

It was established by Halldórsson [Hal09] that for a set of nearly equilength unicast links L there exists a close relationship between link graphs $G_q(L)$ and unit disk graphs $G'_q(L)$.

Lemma 3.2. [Hal09] *For any $q \geq 1$ and a set L of nearly equilength unicast links $G'_q(L) \subseteq G_{q+1}(L)$ and $G_q(L) \subseteq G'_{2(q+1)}(L)$.*

It was shown that unicast links which belong to the same q^α-signal slot are q-independent, and as such a weaker version of the converse of this statement is established with the following result:

Lemma 3.3. **[Hal09]** *A z-independent set S of nearly equilength unicast links under uniform power assignment S is an $\Omega(z^{\alpha})$-signal set.*

If the link set is nearly equilength then the following lower bound on *opt* applies:

Lemma 3.4. **[Hal09]** *Given a set L of nearly equilength unicast links and a constant q, opt$(L) = \Omega(\Delta(G_q(L)))$.*

Combining the above results shows the existence of a constant approximation ratio for a set L of nearly equilength unicast links. Haldórsson observes that one can greedily colour either $G_q(L)$ or $G'_{q'}(L)$, for suitable constants q, q'. This yields the following theorem:

Theorem 3.5. **[Hal09]** *Let $p \geq 1$ be an arbitrary constant. There exists $q = q(p) \geq 1$ such that for any set L of nearly equilength unicast links, any colouring of $G'_q(L)$ with $\mathcal{O}(\Delta(G'_q(L)))$ colours yields a p-signal schedule with uniform power that is within a constant factor of the optimal p-signal schedule for L.*

Proof. Following from Lemma 3.3, for every p there is exists z, such that any z-independent set of nearly equilength links is a p-signal set (with uniform power). Let $q = 2(z+1)$. By Lemma 3.2, $G_z(L) \subseteq G'_q(L)$, and hence any independent set in $G'_q(L)$ is z-independent and therefore a p-signal slot. Thus, any colouring of $G'_q(L)$ constitutes a p-signal schedule. By Lemma 3.4, the optimal 1-signal schedule for L has length $\Omega(\Delta(G_{q+1}(L)))$. As the optimal p-signal schedule is at least as long as the optimal 1-signal schedule, the optimal p-signal schedule for L has length $\Omega(\Delta(G_{q+1}(L)))$ and thus, by Lemma 3.2, length $\Omega(\Delta(G'_q(L)))$. $\square$

Consider the following algorithm, denoted by alg, for scheduling a set of nearly equilength multicast groups. We create a unit disk graph H that has one disk for every multicast group, with centre at the receiver of the longest link in that multicast group. The radius of all disks is $(q/2+4)D$, where q is suitably chosen in accordance with Theorem 3.5. Then a greedy colouring of that unit disk graph is returned as the schedule, i.e., each multicast group is assigned to be scheduled in the round given by the colour assigned to the corresponding disk. The algorithm concludes by assigning uniform power to all senders.

Algorithm 1: Algorithm (alg) for nearly equilength multicast requests

```
Data: Input: A set $M$ of nearly equilength multicast requests
Let $q = q(p)$ in accordance with Theorem 3.5.;
Construct the unit disk graph $H$, where each disk has radius
$(q/2+4)D$ centred at the receivers $r_{v_1}$ of all $m_v \in M$.;
Compute a greedy colouring of $H$, with $\mathcal{O}(\Delta(H))$ colours.;
Return the colouring of $H$ as a $p$-signal schedule $S_M$ with uniform
power assignment.;
```

For a set M of multicast groups, let L_M be the set of unicast links obtained by taking the longest link ℓ_{v_1} from every multicast group $m_v \in M$.

Lemma 3.6. *Let M be a set of nearly equilength multicast groups. For any constant $p \geq 1$, any greedy colouring of the unit disk graph H constructed by the algorithm gives a p-signal schedule for M whose length is at most a constant factor longer than the optimal p-signal schedule for L_M.*

Proof. Let D be an integer, such that the length of the longest link in each multicast group lies in the interval $[D, 2D]$. The algorithm chooses $q =$

$q(p) \geq 1$ in accordance with Theorem 3.5. Hence, we have that any colouring of $G'_q(L_M)$ assigning $\mathcal{O}(\Delta(G'_q(L_M)))$ colours yields a p-signal schedule with uniform power for L_M, that is within a constant factor of $\text{OPT}_p(L_M)$.

The unit disk graph H constructed by the algorithm is the unit disk graph obtained from $G'_q(L_M)$ by increasing the radius of the disks from $qD/2$ to $(q/2+4)D$. As $q \geq 1$, this will increase the radius of the disks by at most a factor of 9. Following from Lemma 3.1 the maximum degree of H is within a constant factor of the maximum degree of $G'_q(L_M)$. Hence, a greedy colouring of H with $\mathcal{O}(\Delta(H))$ colours uses only $\mathcal{O}(\Delta(G'_q(L_M)))$ colours. Note that any greedy colouring of H is also a colouring of $G'_q(L_M)$ with $\mathcal{O}(\Delta(G'_q(L_M)))$ colours and thus, by Theorem 3.5, constitutes a p-signal schedule S_{L_M} for L_M, that is a constant factor approximation of $opt_p(L_M)$.

Observe that for any multicast group m_v, the disks in H with centre r_{v_1} contains the disks with radius $qD/2$ centred at any receiver r_{v_j} of the multicast group m_v, as the distance between the two receivers of the same multicast group, is at most $4D$ by the triangle inequality. This follows from the fact that each unicast link in m_v has length at most $2D$.

We continue by claiming that a greedy colouring of H yields a p-signal schedule S_M for M with uniform power, that is within a constant factor of $opt_p(L_M)$. Since the maximum degree of H is within a constant factor of $G'_q(L_M)$, the number of colours in any greedy colouring of H is a constant factor approximation of $opt_p(L_M)$.

It remains to show that the schedule S_M derived from any greedy colouring of H constitutes a p-signal schedule. Consider an arbitrary link ℓ_{v_i} of a multicast group m_v. Let U be the set of all multicast groups $m_u \neq m_v$

that are scheduled in the same round as m_v in S_M. If $i = 1$, i.e., if l_{v_i} is the longest link of m_v, we can argue with the following observations. The received signal strength from s_v at r_{v_1} and the total strength of interfering signals received at r_{v_1} are the same in S_M and in S_{L_M}, so the affectance at r_{v_1} is at most $1/p$ in S_M.

If $i \neq 1$ the argumentation becomes more involved. Let s'_v be an arbitrary point in the plane that has distance $2D$ from r_{v_i}, and let ℓ'_v be the unicast link with sender s'_v and receiver r_{v_i}. Consider the unit disks of radius $qD/2$ centred at r_{v_i}, and at all r_{u_1} for $m_u \in U$. Observe that these unit disks are the disks that constitute $G'_q(L')$, where L' is the set of unicast links containing the link ℓ'_v and the links ℓ_{u_1} for all $m_u \in U$. Furthermore, these unit disks are disjoint since they are contained in the respective disks of radius $(q/2 + 4)D$ that have been assigned the same colour in H.

By Theorem 3.5, the links in L' constitute a p-signal set $S_{L'}$. In the round in which m_v is scheduled in S_M, the received signal strength from s'_v at r_{v_i} in $S_{L'}$, and the total strength of interfering signals received at r_{v_i} is the same in S_M and in $S_{L'}$. Therefore, the affectance at r_{v_i} is at most $1/p$ in S_M. □

As any p-signal schedule for M is also a p-signal schedule for L_M, it is clear that $opt_p(L_M) \leq opt_p(M)$. Thus, we obtain the following corollary.

Corollary 3.7. *For nearly equilength multicast groups and any constant $p \geq 1$, the algorithm alg attains a constant-factor approximation ratio for M-Scheduling.*

We can now consider arbitrary sets M of multicast requests using the standard approach of partitioning the requests into a logarithmic number of length classes. Assume without loss of generality that the lengths of the longest links of all multicast groups $m_v \in M$, lie in the interval $[1,\ \Gamma]$. Partition the set M of multicast groups into $\lceil \log \Gamma \rceil$ classes M_i, where M_i consists of all multicast groups whose corresponding longest link has a length that lies in the interval $[2^i, 2^{i+1})$. Apply *alg* to each class M_i separately and obtain a schedule for M by concatenating the schedules for all such classes of multicast groups M_i. As the schedule for each class M_i is a constant factor approximation of $opt_p(M_i)$ and therefore also of $opt_p(M)$, we obtain a p-schedule that is an $\mathcal{O}(\log \Gamma)$-approximation of $opt_p(M)$.

Theorem 3.8. *For every constant $p \geq 1$, there exists an $\mathcal{O}(\log \Gamma)$-approximation algorithm for the problem of computing a shortest p-signal schedule for a given set of multicast requests.*

It follows that because partitioning into length classes and the greedy colouring of the corresponding unit disk graphs for each length class can be performed online, the same approach yields an $\mathcal{O}(\log \Gamma)$-competitive online algorithm for multicast scheduling. Furthermore, as pointed out by Halldórsson [Hal09], approaches based on colouring of unit disk graphs are amenable to distributed implementation.

3.5 Signal Strengthening

It is also interesting to relate the length of the optimal p-signal schedule to the optimal 1-signal schedule. The following result from [HW09] shows that in the unicast case, a larger SINR for all receivers can be achieved at a constant-factor loss in the schedule length.

Lemma 3.9. **[HW09]** *There is a polynomial time algorithm that takes a p-signal schedule for a set of unicast links and refines it to a p'-signal schedule for some $p' > p$, increasing the length of the schedule by a factor of at most $\lceil 2p'/p \rceil^2$.*

Extending this lemma to the multicast setting does not seem straightforward, however we are able to establish an analogous result at least for nearly equilength multicast links.

Lemma 3.10. *If there exists a p-signal schedule for a set of nearly equilength multicast groups M with length k, then there also exists a p'-signal schedule for M for any constant $p' > p$ of length $\mathcal{O}(k)$. Furthermore, such a schedule can be computed within polynomial time.*

Proof. Let $\mathcal{A}$ be a p-signal schedule for a set M of multicast groups. Let L_M be the set of unicast links obtained by taking the longest link ℓ_{v_1} from every multicast group m_v in M.

We note that $\mathcal{A}$ can also be viewed as a p-signal schedule for L_M. As L_M is a set of unicast links, we can transform $\mathcal{A}$ into a p'-signal schedule $\mathcal{A}'$ for L_M by Lemma 3.9, such that the length of the schedule increases by only a constant factor. As the optimal p'-signal schedule for L_M cannot be longer than $\mathcal{A}'$, we have that the length of the optimal p'-signal schedule for L_M, $opt_{p'}(L_M)$, is within a constant factor of the length of $\mathcal{A}$.

Consider the p'-signal schedule S_M computed for M by *alg* in polynomial time. Following from Lemma 3.6, the length of S_M is within a constant factor of $opt_{p'}(L_M)$, and therefore also within a constant factor of the length of $\mathcal{A}$. Thus, S_M is a p'-signal schedule for M, that is within a constant factor of the length of $\mathcal{A}$. $\square$

For an optimal 1-signal schedule for M, from application of Lemma 3.10 we obtain the following corollary:

Corollary 3.11. *Given a set of nearly equilength multicast groups M and a constant $p \geq 1$, $opt_p(M)$ is at most a constant factor longer than $opt(M)$.*

Intuitively the above corollary shows that if one requires a stronger SINR in a schedule, this can be achieved with a loss of only a constant factor in the schedule length for nearly equilength multicast groups.

3.6 Splittable versus Atomic Multicast

We proceed in this section to discuss the relationship between splittable and atomic multicast requests. For ease of presentation we consider only 1-signal schedules. For a given set M of multicast requests, denote by $opt^{\mathcal{S}}(M)$, the length of an optimal 1-signal schedule that is allowed to split a multicast request. As before, $opt(M)$ denotes the length of an optimal 1-signal schedule with atomic multicast requests.

As every atomic schedule is also a splittable schedule, it is clear that $opt^{\mathcal{S}}(M) \leq opt(M)$ for any set M of multicast requests. Furthermore we establish the following lemmas.

Lemma 3.12. *Given a set M of nearly equilength multicast groups, $opt(M) = \mathcal{O}(opt^{\mathcal{S}}(M))$.*

Proof. Recall that L_M is the set of unicast links obtained by taking the longest link ℓ_{v_1} from every multicast group $m_v \in M$. We have $opt(L_M) \leq opt^{\mathcal{S}}(M)$ since even a splittable schedule must schedule the longest link of each multicast group in some round. Moreover, by Lemma 3.6 there is an atomic 1-signal schedule for M that is within a constant factor of $opt(L_M)$. □

Lemma 3.13. *For any set M of multicast groups, $opt(M) = \mathcal{O}(\log \Gamma) \cdot opt^{\mathcal{S}}(M)$.*

Proof. Consider the partitioning of M into $\mathcal{O}(\log \Gamma)$ length classes M_i, such that each length class is nearly equilength. The maximum of $opt(L_{M_i})$ over all i is a lower bound of $opt^{\mathcal{S}}(M)$. Furthermore, the algorithm presented in Section 3.4 computes an atomic schedule for M that is within a constant factor of the sum of values $opt(L_{M_i})$ over all i. □

3.7 Throughput Maximisation

3.7.1 Algorithms

In this section, we discuss how the approach described in Section 3.4 can be adapted to the problem *M-Throughput.* For the following we consider only atomic multicast groups, i.e., requests/groups for which all receivers within the group must receive the transmission in the same round. For a given set M of nearly equilength multicast groups one can construct the unit disk graph H as in alg, and then compute a maximal independent set I in H, via standard techniques (for example using a greedy algorithm). The set

I forms a 1-signal set. It is known ([MBHI$^+$95]) that in unit disk graphs, the size of any maximal independent set is within a factor of 5 of that of a maximum independent set.

Let $I^* \subseteq M$ be a 1-signal set of largest size. By Lemma 3.6, the unit disks in H corresponding to the multicast groups in I^* can be coloured with a constant number of colours. Hence, I^* contains a set of multicast groups that corresponds to an independent set in H of size $\Omega(|I^*|)$. The set I computed by alg then also has size $\Omega(|I^*|)$ and therefore constitutes a constant-factor approximation of the largest 1-signal schedule.

For general sets of multicast requests, we can partition the multicast requests into $\mathcal{O}(\log \Gamma)$ length classes, compute a 1-signal set for each length class as described above, and output the largest of these 1-signal sets. This gives an $\mathcal{O}(\log \Gamma)$-approximation for *M-Throughput*. The same approach can be used to obtain a randomised $\mathcal{O}(\log \Gamma)$-competitive online algorithm. We note that the randomisation is only needed to select one of the length classes at the beginning of the algorithm.

3.7.2 Online Lower Bound

We now present a lower bound showing that no deterministic online algorithm for the throughput problem can achieve competitive ratio better than $\mathcal{O}(\log \Gamma)$ even in the case of unicast requests. We make use of the following property stated by Avin et al. [ALP09].

Lemma 3.14. [ALP09] *Two senders s_1 and s_2 cannot transmit simultaneously if their respective receiver is closer to the other sender, i.e., if*

$\delta(s_1, r_1) > \delta(s_1, r_2)$ and $\delta(s_2, r_2) > \delta(s_2, r_1)$.

Utilising Lemma 3.14 we provide a construction bounding the performance of any arbitrary deterministic algorithm against an adversary that uses arbitrary power assignment, for the online variant of *Throughput*.

Theorem 3.15. *The competitive ratio of any deterministic online algorithm, even when permitted arbitrary power assignment, is $\Omega(\log \Gamma)$ for Throughput.*

Proof. Consider the following construction. Let n be an arbitrary positive integer and let $\Gamma = 2b^n$ for a sufficiently large constant $b > 1$. All senders and receivers are located on the x-axis, so we identify a node solely with its x-coordinate.

Let A be an arbitrary deterministic deterministic online algorithm. The adversary first presents a request ℓ_0 with sender $s_0 = 2b^n$ and receiver $r_0 = 0$. The algorithm must accept ℓ_0 as otherwise the competitive ratio is unbounded if no further requests are presented. Next, the adversary presents requests $\ell_1, \ldots, \ell_n$ where for each $1 \leq i \leq n$ the sender and receiver of ℓ_i are $s_i = -b^i$ and $r_i = b^i$ respectively.

By Lemma 3.14, the algorithm cannot accept any of the requests $\ell_1, \ldots, \ell_n$ as none of them can transmit at the same time as ℓ_0. It remains to show that an optimal solution can reject ℓ_0 and accept all other requests.

Let $\ell_1, \ldots, \ell_n$ transmit simultaneously using the square root power assignment, i.e., assign power $\sqrt{b^i}$ to s_i for all $1 \leq i \leq n$. The strength of the signal received at r_i from s_i is:

$$\frac{\sqrt{b^i}}{(2b^i)^\alpha} = \frac{b^{(0.5-\alpha)i}}{2^\alpha}$$

The total interference received at r_i is:

$$\sum_{j<i} \frac{\sqrt{b^j}}{(b^j + b^i)^\alpha} + \sum_{j>i} \frac{\sqrt{b^j}}{(b^j + b^i)^\alpha} \tag{3.3}$$

We can bound the first sum in (3.3) as follows:

$$\sum_{j<i} \frac{\sqrt{b^j}}{(b^j + b^i)^\alpha} \leq \sum_{j<i} \frac{\sqrt{b^j}}{(b^i)^\alpha} \leq \frac{\sqrt{b^i}}{(\sqrt{b} - 1) \cdot (b^i)^\alpha} = \frac{b^{(0.5-\alpha)i}}{\sqrt{b} - 1}$$

The second sum in (3.3) can be bounded as follows:

$$\sum_{j>i} \frac{\sqrt{b^j}}{(b^j + b^i)^\alpha} \leq \sum_{j>i} \frac{\sqrt{b^j}}{(b^j)^\alpha} = \sum_{j>i} b^{(0.5-\alpha)j} \leq 2 \cdot b^{(0.5-\alpha)(i+1)}$$

where the last inequality holds for a sufficiently large b. The SINR at r_i is greater than:

$$\frac{b^{(0.5-\alpha)i} \cdot 2^{-\alpha}}{b^{(0.5-\alpha)i} \cdot (\frac{1}{\sqrt{b}-1} + 2 \cdot b^{0.5-\alpha})} = \frac{2^{-\alpha}}{\frac{1}{\sqrt{b}-1} + 2 \cdot b^{0.5-\alpha}}$$

For a sufficiently large constant (b that is chosen depending on α), the SINR is larger than 1 (or any other desired constant SINR threshold). $\square$

We remark that the lower bound of Theorem 3.15 will still apply, even if one allows an algorithm to retrospectively change the power assigned to previously accepted requests, upon acceptance of a new request.

3.8 Conclusion

In this chapter we have made contributions to both wireless scheduling and throughput problems in the context of multicast requests in the SINR model. Building upon the relationship between SINR scheduling and the colouring of unit disk graphs, which was established in earlier work by Halldórsson [Hal09], we have presented $\mathcal{O}(\log \Gamma)$-competitive algorithms for both settings.

We have shown that the approach of reducing SINR scheduling problems for nearly equilength links to unit disk graph colouring extends to multicast requests as well, provided that the longest links in the multicast groups are nearly equilength.

We also have shown a $\Omega(\log \Gamma)$ lower bound on the competitive ratio of any deterministic online algorithm for the throughput problem, even in the case of unicast links and arbitrary power assignments, and additionally have discussed the relationship between scheduling with atomic multicast requests, and splittable multicast requests.

It would be interesting to know whether signal strengthening can be applied for arbitrary multicast requests, while losing only a constant factor in the approximation ratio. We know for arbitrary unicast requests this is possible (Lemma 3.9), however, for multicast requests it has only been proved for the case where the links are nearly equilength (Lemma 3.10).

Additionally, it would also be interesting to know whether one can reduce the factor of $\mathcal{O}(\log \Gamma)$ which bounds the difference in the length of a schedule between atomic and splittable schedules, presented in Lemma 3.13.

In recent work, Kesselheim [Kes11] has shown the existence of a constant approximation algorithm for the throughput problem for unicast links. The overriding open problem related to SINR is to obtain a constant factor approximation algorithm for the scheduling problem for unicast links. An interesting open problem related to our extension to the multicast setting would be to see if the results yielding a constant approximation algorithm for the unicast throughput problem, can be generalised to the multicast case.

Chapter 4

Target Coverage

In this chapter we consider the problem of maximising the lifetime of a network for fault-tolerant target coverage of composite events, with several event types.

Consider a sensor network whose task is to detect the occurrence of events at a given set of event points. This is also known as the *target coverage problem.* Since the nodes in a sensor network often have a limited battery supply that cannot be replenished, it is important to address the problem of maximising the lifetime of the network, i.e., the length of time during which the network can carry out its monitoring task successfully. The lifetime of the network can be prolonged by calculating an activity schedule in which only a subset of the sensor nodes is active at any point in time and the remaining sensors are in a sleep mode that saves energy. The active nodes must be sufficient for performing the required monitoring task. Following [VBL07, MYC09], we consider the setting where the events to be detected are *composite events*, i.e., events comprising several simultaneous *atomic* events at the same location detected by different sensor types, and the sensor coverage is required to be *fault-tolerant*, i.e., the failure of any one sensor does not affect the sensing task.

An atomic event is a physical change in the environment such as temperature rising above a pre-defined threshold, and a composite event is a combination of several atomic events, all occurring concurrently at the same location. As an example motivating the study of composite events, one can consider the scenario described by Vu et al. [VBL07] where fire is detected by the composite event of temperature being above a given threshold and in addition the density of smoke being above a pre-defined value.

The settings described in [VBL07, MYC09] also require that each event is covered by k sensors as a fault tolerant mechanism. One may wish to cover each event by k sensors, such that if up to $k-1$ sensors fail, there will still be at least one active sensor that can detect and report the event. Obviously, the higher the value of k, the more robust the network will be against failing sensor nodes. In this work, we mainly consider the case of $k = 2$. This case is of interest in many application settings, because higher levels of fault tolerance are often considered to consume too many resources. Moreover, handling larger values of k is more complicated.

We assume that the sensor nodes and the event points are located in the two-dimensional Euclidean plane, and that all sensor nodes have uniform sensing radius. Each sensor node can monitor a certain set of event types, and the composite event to be detected at each event point is a combination of atomic events, corresponding to different event types. We remark that the presence of multiple event types adds a non-geometric, combinatorial aspect to the problem.

A common approach to the lifetime maximisation is to formulate the problem as a linear program and obtain an approximate solution by approximating the dual problem of computing a sensor cover of minimum weight (see Section 4.6 for details). We follow the same approach and hence mainly consider the dual problem of minimising the weight of a fault-tolerant sensor cover. We model the latter problem as a weighted multi-T-cover with unit disks, where T is the set of event types.

We remark that in the case of atomic events, where all such events are of the same event type (i.e., $|T| = 1$), and no fault-tolerance requirements are imposed (i.e., $k = 1$), the minimum weight sensor cover problem is a standard geometric set cover problem with unit disks, where the aim is to cover a given set of points, using disks of minimum total weight. This standard problem is well understood and is known to be $\mathcal{NP}$-hard, even in the unweighted case [CCJ90], motivating the study of approximation algorithms. In the weighted case of geometric set cover with unit disks, the best known approximation ratio is $4 + \varepsilon$ [EM09, ZWX$^+$11].

Our setting poses the additional challenges of having to cover every point twice (turning the problem into a *multi-cover* problem), while avoiding the loss of a factor of two in the approximation ratio. Additionally, we must deal with different event types and consequently composite events. Addressing these challenges requires us to refine the techniques that have been developed for the standard geometric set cover problem with unit disks.

4.1 Related Work

Sensor cover problems have been studied in several variants, including target coverage problems where a discrete set of points that need to be monitored is specified in the input, and region coverage problem where the area to be monitored is specified as a (typically convex) region in the plane. We refer the interested reader to the survey by Thai et al. [TWDJ08] for an overview.

Some of the work on the lifetime maximisation variant (where the nodes in the network will have limited battery power) of sensor cover problems has focussed on models where the sets of sensors that are active at different times must be disjoint [CD05]. However, it was observed in [BCSZ04, BCSZ05, CTLW05] that the use of non-disjoint sensor covers can yield a significantly extended lifetime of the network.

Berman et al. [BCSZ04, BCSZ05] show that the region coverage problem can be reduced to the target coverage problem and present an algorithm attaining logarithmic competitive ratio. They also show that a minimum cost sensor cover algorithm with approximation ratio ρ implies an approximation algorithm with ratio $\rho(1+\varepsilon)$ for the lifetime maximisation problem using the Garg-Könemann algorithm [GK98a].

Dhawan et al. [DVZ$^+$06] study a target coverage problem where the sensor nodes can adjust their sensing range, and the aim is to maximise the network lifetime. They propose a greedy algorithm for the minimum cost sensor cover problem that yields a logarithmic competitive ratio. Zhao and Gurusamy [ZG08] study the target coverage problem with the additional requirement that the sensors that are active at any time are connected. They obtain an algorithm with logarithmic competitive ratio and also present a performance evaluation based on simulation experiments. Sanders and Schieferdecker [SS10] show that the target coverage problem for sensors represented by unit disks with the objective of lifetime maximisation is $\mathcal{NP}$-hard. They further provide a $(1+\varepsilon)$-approximation algorithm using resource augmentation, i.e., their algorithm needs to increase the sensing range of every sensor node by a factor of $1+\delta$, for some fixed $\delta > 0$.

Much of the previous work on target coverage problems has not considered fault-tolerance requirements, composite events or different sensor types. Vu et al. [VBL07] and Marta et al. [MYC09] consider fault-tolerant sensor cover problems with composite events. They present centralised and distributed heuristics and evaluate them in simulations. Contrary to their setting, in this work we aim at designing approximation algorithms with provable performance guarantees for fault-tolerant sensor cover problems with composite events.

A special case of the minimum cost sensor cover problem is the weighted geometric set cover problem with unit disks. Given as input a set of points in the plane, and a set of weighted unit disks, the aim is to compute a cheapest set of disks that covers all points. This problem has received considerable attention as it includes the weighted dominating set problem for unit disk graphs, which is relevant for routing backbone construction in wireless networks. This relationship also shows that the problem is $\mathcal{NP}$-hard as the minimum dominating set problem for unit disks is known to be $\mathcal{NP}$-hard [CCJ90].

The first constant factor approximation for weighted set cover with unit disks was a 72-approximation by Ambühl et al. [AEMN06]. The plane is partitioned into squares of constant size, and the algorithm computes a 2-approximation for each square separately and outputs the union of the solutions for all squares. As a subroutine, they show that the problem can be solved optimally in polynomial time by dynamic programming if the points to be covered are located in a (horizontal or vertical) strip of the plane, and all disks have centre outside of that strip. This inaugural constant approximation result was then improved to a $(6 + \varepsilon)$-approximation by Huang et al. [HGZW09] by considering blocks of the plane, consisting of a bounded number of squares, and applying a geometric shifting strategy [HM85]. They compute separate solutions for each horizontal and vertical strip of squares inside a block. They also introduce a *sandglass* technique by which an algorithm 'guesses' properties of the optimal solution. This allows the algorithm to decide which of the points in a square should be covered by disks with centre above or below the square the point appears in, and which points are covered by disks with centre to the left or right of the square. This approach employed by Huang et al. [HGZW09] was further amended by Dai and Yu [DY09], yielding a $(5 + \varepsilon)$-approximation. The improvement is obtained by calculating solutions over pairs of strips of squares simultaneously, combined with techniques from [HGZW09]. A further improvement to a $(4+\varepsilon)$-approximation was then attained by Erlebach and Mihalák [EM09] and, independently by Zou et al. [ZWX$^+$11]. The main idea in [EM09] is to compute solutions for K adjacent strips of squares simultaneously using a 'split sweepline' technique that ensures that disks that cover points in different strips are met by the corresponding sweepline pieces at the same time.

All the aforementioned results in the previous paragraph apply to weighted geometric set cover with unit disks, and consequently also to the minimum-weight dominating set problem in unit disk graphs. For this latter problem, the variant where the dominating set has the constraint that it must be connected, is also of interest, i.e., the minimum-weight connected dominating set problem. The approach followed by several authors in previous work [AEMN06, DY09, EM09, HGZW09, ZWX$^+$11] is to first compute a dominating set of minimum cost, and then to apply an algorithm solving the node-weighted Steiner tree problem, in order to connect the dominating set. The node-weighted Steiner tree problem admits a $2.5 \cdot \alpha$-approximation algorithm in unit disk graphs [ES09, ZLGW09], where α is the approximation ratio of the best known approximation algorithm for edge-weighted Steiner trees, which is used as a subroutine.

In 2010 a result by Byrka et al. [BGRS10] showed that for node-weighted Steiner trees in unit disk graphs we have $\alpha < 1.39$, and thus an approximation algorithm with ratio less than 3.475 for node weighted Steiner trees in unit disk graphs. In combination with the best known approximation algorithm for minimum-weight dominating sets, which to the extent of our knowledge is from Erlebach and Mihalák [EM09], and independently Zou et al. [ZWX$^+$11], who show a $(4 + \varepsilon)$-approximation algorithm, this yields a 7.475-approximation algorithm for minimum-weight connected dominating sets in unit disk graphs.

The unweighed set multi-cover problem has been studied in geometric settings by Chekuri et al. [CCHP09]. They present an $\mathcal{O}(\log opt)$-

approximation algorithm for set systems of bounded VC dimension where *opt* is the size of an optimal cover, and constant factor approximation algorithms for covering points by half spaces in three dimensions or covering points with pseudo-disks (disks that intersect each other in at most two points) in the Euclidean plane. Their results only apply to the unweighed variant.

4.1.1 Contributions

We model the fault-tolerant target coverage problem with composite events as a generalised geometric multi-cover problem with unit disks and present a $(6+\varepsilon)$-approximation algorithm. This approximation ratio holds for both the minimum cost sensor cover variant of the problem and the lifetime maximisation variant.

On a high level, we solve the minimum cost sensor cover problem by providing a 6-approximation algorithm for the case where all event points are located in a square of the plane of bounded size, and employing a geometric shifting strategy [HM85, HGZW09]. To obtain this 6-approximation of a solution for a block, we 'guess' a number of properties of an optimal solution by enumeration, and then apply dynamic programming along horizontal and vertical strips of smaller squares. As a result of this enumeration procedure, we need only handle the case where disks with centre outside a strip are used to cover points inside the strip, which makes a dynamic programming approach feasible.

Our algorithm requires significant adaptations compared with previous work. Due to the multi-cover aspect a more involved 'guessing' step is required, and the algorithm must be able to handle different event types. Using our approximation algorithm for minimum cost sensor cover as a subroutine in the Garg-Könemann algorithm [GK98a], we obtain the same approximation ratio $6 + \varepsilon$ for the lifetime maximisation problem. Furthermore, provided that the communication radius is at least twice the sensing radius, we can use the known approximation algorithm for the node-weighted Steiner tree problem in unit disks graphs, and achieve approximation ratio 9.475 for the problem variants where the set of active sensors is required to form a connected communication network.

The remainder of the chapter is structured as follows. In Section 4.2, we formally introduce and define the problems we study and briefly describe the aspects of our approach that are reasonably standard, e.g., partitioning the plane into squares of bounded size, and then applying a geometric shifting strategy. In Section 4.3, we present a high-level description of our algorithm for approximating the minimum cost fault-tolerant sensor cover problem with composite events, for a block containing a set number of squares in the plane. In Sections 4.4 and 4.5 we then continue by discussing the details of two key components of our algorithm, namely the enumeration procedure that 'guesses' certain properties of a fixed optimal solution and the dynamic programming approach that, based on these guessed properties, will solve subproblems of the block, namely a horizontal or vertical strip of squares in the block, to optimality. In Section 4.6, we describe how the Garg-Könemann algorithm [GK98a] can be applied in our setting, using a ρ-approximation algorithm for the minimum cost sensor cover problem as a subroutine, to yield a $\rho(1 + \varepsilon)$-approximation algorithm for the lifetime maximisation variant of the problem. We show that this general approach, which has already been employed in previous work, can be adapted to generalise to our setting as well. In Section 4.7 we show how our results can be adapted to the problem variants where sensor covers are required to form a connected communication graph. Finally, we conclude in Section 4.8 by providing a summary of the presented material and indicate some directions for future work that could be of interest.

4.2 Preliminaries

In this section, we introduce basic notation, formally define the problems considered in this chapter, and describe the partitioning of the plane that is used by our algorithms.

Consider the two-dimensional Euclidean plane. The x-coordinate and y-coordinate of a point p is denoted by x_p and y_p, respectively. The Euclidean distance between two points p and q is denoted by $\delta(p,\ q)$. If d is a disk, we also use d to refer to the centre of the disk, such that we can write $\delta(d,\ p)$ to denote the Euclidean distance between the centre of a disk d and a point p in the plane. Note that, against conventional notation, $\delta(d,\ p)$ does not denote the minimum distance between p and any point in d, but rather the Euclidean distance between the centre of d and a specific point p. We say that a point p is *in a disk* d of radius r if $\delta(d,\ p) \leq r$. Additionally, we say that p *is on a disk* d if it lies on the boundary of the disk, i.e., $\delta(d,\ p) = r$.

4.2.1 Problem Definitions

An instance of the *weighted (geometric) set cover problem with unit disks* is given by a set P of points in the two-dimensional Euclidean plane and a set D of weighted unit disks. All disks have uniform radius r, and without loss of generality we assume $r = 2$ throughout this chapter. We note that this choice of $r = 2$ is consistent with previous work, e.g., [EM09], where the dominating set problem for unit disks of radius 1 was transformed into an equivalent geometric set cover problem, where all disks have radius 2. The weight of a disk $d \in D$ is non-negative and denoted by $w(d)$ or w_d. The total weight of a set $D' \subseteq D$ of disks is denoted by $w(D') = \sum_{d \in D'} w(d)$. The goal of the weighted set cover problem with unit disks is to select a set of disks of minimum total weight such that every point p is in at least one of the selected disks.

In the context of the target coverage problem, the disks in D correspond to sensor nodes (with r representing the sensing radius) and the points in P correspond to targets (event points) that need to be monitored.

To model fault-tolerance requirements, we consider the following extension of the set cover problem. Every target $p \in P$ specifies a positive integer k_p as its coverage requirement, and a set $D' \subseteq D$ of disks is a *feasible multi-cover* if every $p \in P$ is in at least k_p distinct disks of D'. The goal of the *weighted multi-cover problem with unit disks* is to compute a feasible multi-cover D' of minimum total weight.

Furthermore, to model different event types, we assume that there is a (small) set T of different event types (e.g., smoke, temperature, etc.) and every sensor $d \in D$ has sensing components for a subset $T_d \subseteq T$ of event types. Moreover, each target $p \in P$ needs to be monitored with respect to a subset $T_p \subseteq T$ of event types, corresponding to the occurrence of a composite event comprised of atomic events for each type in T_p. A set $D' \subseteq D$ of disks is a *feasible multi-T-cover* if, for each p in P and each $t \in T_p$, p is in at least k_p distinct disks d' in D' with $t \in T_{d'}$, i.e., if

$$\forall p \in P \ : \ \forall t \in T_p \ : \ |\{d' \in D' \ : \ \delta(d',\ p) \leq r,\ t \in T_{d'}\}| \geq k_p \tag{4.1}$$

To simplify matters, we reduce composite events to equivalent atomic events as follows. We replace every point p, that needs to be monitored with respect to a set $|T_p|$ of event types, by $|T_p|$ copies of p. Every copy of p has the requirement to be monitored with respect to a distinct $t \in T_p$, and with the same coverage requirement k_p. For every new copy p' of p, we denote by $t_{p'}$ its associated event type. We note that a set D' of disks is a feasible multi-T-cover of the original points with composite monitoring requirements if and only if it is a feasible multi-T-cover of the modified points with only atomic monitoring requirements.

Therefore, without loss of generality, we can now assume that it is required that every point $p \in P$ is monitored only with respect to one event type t_p. We remark that after this procedure, P may contain points with identical coordinates. We now have that a disk $d \in D$ *covers* a point $p \in P$, if p is in d, and $t_p \in T_d$. A set $D' \subseteq D$ of disks *meets the coverage requirements* of a point $p \in P$, if p is covered by at least k_p distinct disks in

D'.

We now formally define the weighted multi-cover problem under consideration:

Definition 4.1. *For a set T of event types, the weighted multi-T-cover problem with unit disks is denoted by WMCUD-T. We are given a set D of disks of radius $r = 2$, each disk $d \in D$ associated with a non-negative weight w_d and a set T_d of event types, and a multi-set P of points, each point $p \in P$ associated with one event type $t_p \in T$ and a coverage requirement k_p. A subset $D' \subseteq D$ is a feasible multi-T-cover if:*

$$\forall p \in P \ : \ |\{d' \in D' \ : \ \delta(d', \ p) \leq r, t_p \in T_{d'}\}| \geq k_p \tag{4.2}$$

The objective is to compute a feasible multi-T-cover of minimum total weight. The restriction of WMCUD-T to the case where $k_p \leq 2$, for all $p \in P$ is denoted by W2CUD-T.

Furthermore, we remark that the assumption that the cardinality of the set of event types T is small, i.e., bounded above by a fixed constant value, is natural in this application setting. If the set of event types T is allowed to be of arbitrarily large size, then even covering a single target would be analogous to solving a general set cover problem and cannot be approximated to a constant factor unless $\mathcal{P} = \mathcal{NP}$, [AMS06, Fei98].

We also note that the running time of our algorithm is exponential in $|T|$, but polynomial in general as $|T|$ is bounded above by a fixed constant.

Finally, let us define the lifetime maximisation variant of the problem.

Definition 4.2. *We are given points and disks as in an instance of WMCUD-T, but additionally each disk $d \in D$ specifies an initial battery level b_d, expressed in suitable units such that b_d is the total duration during which d can be active before its battery runs out. A* schedule *is a set of pairs $(D_i,\ x_i)$ where $D_i \subseteq D$ is a feasible multi-T-cover and $x_i \geq 0$ is the number of units for which the sensors in D_i are active. A schedule is* feasible *if for each disk $d \in D$, the sum of all x_i values of all pairs $(D_i,\ x_i)$, where $d \in D_i$ does not exceed b_d. The lifetime of a schedule is the sum of all the x_i values of all its associated pairs $(D_i,\ x_i)$. The goal is to compute a feasible schedule of maximum lifetime. We refer to this problem as the maximum lifetime multi-T-cover problem with unit disks (MLMCUD-T), and the restricted version where $k_p \leq 2$, for all $p \in P$, as ML2CUD-T.*

4.2.2 Plane Partition

As in previous work (e.g., [HGZW09]), our algorithms employ a partition of the plane. Consider an infinite grid that partitions the plane into squares, each with side length 1.4 (we remark that any number sufficiently close to, but strictly less than $\sqrt{2}$ would suffice). Consider an arbitrary such square S. Note that any disk of radius 2 with centre in S contains the entire square. We can assume without loss of generality that no point, or disk centre lies exactly on the boundary between two adjacent squares.

The neighbouring infinite regions of a square S are referenced as in Figure 4.1, where UM is *'upper middle'*, CL represents *'centre left'* and LR stands for *'lower right'*. The other regions surrounding the square S_{ij} shown in Figure 4.1 are referenced analogously.

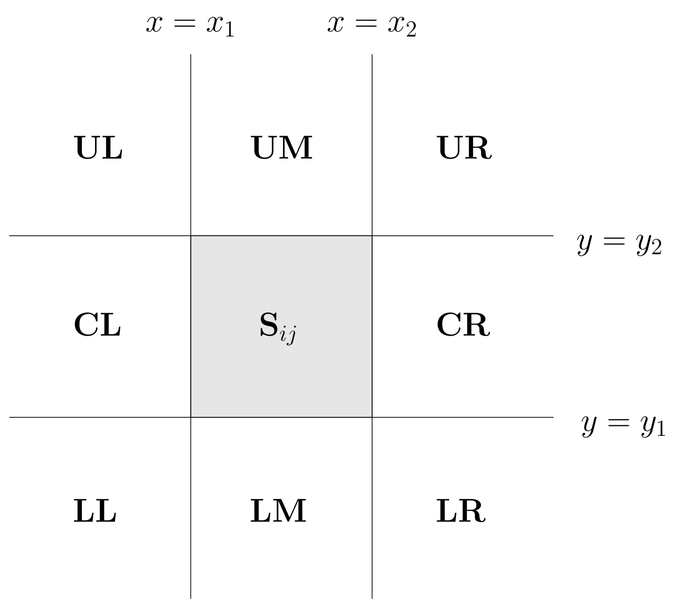

Figure 4.1: Square S_{ij} and neighbouring regions.

Furthermore, let UPPER be the union of the regions UL, UM, UR, and similarly, let LOWER be the union of the regions LL, LM and LR, let LEFT be the union of the regions UL, CL and LL, and finally let RIGHT be the union of the regions UR, CR and LR.

For an integer constant $K \geq 0$ (which determines the ε term in the final approximation ratio (see Theorem 4.11), consider a partition of the plane into *blocks* so that each block B consists of $K \times K$ squares S. For convenience of presentation, we index the squares in a block from the square $S_{0,\ 0}$ in the top left corner, through to the square $S_{K-1,\ K-1}$ in the bottom right corner. We remark for clarity that the indices of a square S_{ij} are local with regard to the block B containing the square. The first index i refers to the row of the block the square appears in, and the second index j to the column of the block, in which S_{ij} is located within the block B. Let P_{ij} be the set of points that lie in the square S_{ij}.

If we have a ρ-approximation for *W2CUD-T* instances whose points all lie in one block, we can obtain a $\rho(1 + \mathcal{O}(1/K))$-approximation for general instances of *W2CUD-T* using a standard geometric shifting strategy [HGZW09]. A sketch of this approach is as follows. The algorithm calculates a ρ-approximate solution for each block and amalgamates these solutions for individual blocks into a solution for the entire plane. Each block is then shifted up and right by four squares, and a new solution is calculated for this new set of blocks. This process is repeated for $K/4$ iterations. The best of the $K/4$ solutions, i.e., the one of minimum cost, is then output as the overall solution.

The analysis of this approach is based on the observation that a disk can cover points in different blocks for only two of the $K/4$ shifted cases, because each disk overlaps at most four horizontal or vertical strips of squares. For any given optimal solution *opt*, there is a shifted problem for which the total weight of disks in *opt* that overlap block boundaries is at most $8 \cdot w(\mathit{opt})/K$, and thus the union of ρ-approximate solutions in the blocks is within a factor of $\rho(1 + \mathcal{O}(1/K))$ of the total weight of the optimal solution. The details of this process are fairly standard and can be found for a similar setting, e.g., in [HGZW09]. Thus, the key to obtaining a good approximation algorithm for *W2CUD-T* is to achieve a good approximation ratio for instances where the points are located in one block.

4.3 A $(6+\varepsilon)$-Approximation Algorithm

In this section, we present a 6-approximation algorithm for *W2CUD-T* in a block. At the end of the section we conclude that there exists a $(6+\varepsilon)$-approximation algorithm for general instances of *W2CUD-T*.

Let an instance of *W2CUD-T* in a $K \times K$ block B of squares be given by a set of points P_B (each point $p \in P_B$ is associated with an event type t_p, and a coverage requirement $k_p \leq 2$) and a set D of disks. Our approach to solve this instance of *W2CUD-T* consists of two stages. In the first stage, using enumeration we 'guess' properties of a fixed optimal solution, denoted by opt_B. We remark that both here and in the following, in any reference to *'the optimal solution'*, we refer to that particular fixed optimal solution. In the second stage, we approximate the best solution with these properties via the use of dynamic programming.

As motivation for our approach to guessing properties of an optimal solution, observe that an optimal solution may contain an arbitrarily large number of disks with centre in the same square. As an example, consider two horizontally adjacent squares S_1 and S_2. Assume that there is a single event type and no fault-tolerant requirement (i.e., we must cover each point (target) with only one disk). Place n targets on a vertical line l_1 in the square S_1. Place the centre of n disks on a vertical line l_2 in the square S_2, in such a way that the distance between l_1 and l_2 is 2. Furthermore, for each target t_p, place a disk on l_2 whose centre has the same y-coordinate as t_p. It follows that each of the n disks covers exactly one of the n targets, and consequently the only solution that covers all n targets as required, consists of all n disks. This shows that an approach that enumerates all disks that are in the optimal solution and have centre in a specific square is not feasible. We overcome this obstacle by enumerating guesses only for certain properties of the optimal solution.

The enumeration stage produces a polynomial number of guesses of the properties of opt_B. Each such guess π specifies a set of disks D_π that are contained in opt_B, which may reduce the coverage requirements of some points in P_B accordingly. For example, the coverage requirement of a point p, with $k_p = 2$ that is contained in one disk $d \in D_\pi$ with $t_p \in T_d$ is reduced to $k'_p = 1$. Furthermore, each point p whose remaining coverage requirement k'_p is not zero is classified according to the regions UL, UM, ... (with respect to the square in which p is contained) where the centres of k'_p disks in the optimal solution that cover p lie.

Definition 4.3. *For a point p the symbol* $\Updownarrow$ *specifies that p must be covered by a disk with centre in* UPPER $\cup$ LOWER*, the symbol* $\updownarrow$ *that the point must be covered by a disk with centre in* UM $\cup$ LM*, the symbol* $\Leftrightarrow$ *that the point must be covered by a disk with centre in* LEFT $\cup$ RIGHT*, and the symbol* $\leftrightarrow$ *that the point must be covered by a disk with centre in* CL $\cup$ CR*. Here, all regions are specified with respect to the square in which the point lies. If a point p has remaining coverage requirement* $k'_p = 1$*, the* classified coverage requirement π_p *can be* $\{\Leftrightarrow\}$ *or* $\{\Updownarrow\}$*. If a point has remaining coverage requirement* $k'_p = 2$*, the classified coverage requirement* π_p *can be* $\{\Leftrightarrow,\ \Leftrightarrow\}$, $\{\Updownarrow,\ \Updownarrow\}$, $\{\Leftrightarrow,\ \updownarrow\}$*, or* $\{\leftrightarrow,\ \Updownarrow\}$*.*

Definition 4.4. *A set* D' *of disks* meets *the classified coverage requirement*

$\pi_p = \{\Leftrightarrow, \Leftrightarrow\}$ of a point p, if the point p is covered by two distinct disks $d_1, d_2 \in D'$ with centre in LEFT$\cup$RIGHT. *A set D' of disks meets the classified coverage requirement $\pi_p = \{\Leftrightarrow, \ \Updownarrow\}$ of a point p, if the point p is covered by one disk $d_1 \in D'$ with centre in* LEFT $\cup$ RIGHT *and by one disk $d_2 \in D'$ with centre in* UM $\cup$ LM. *For the other possible classified coverage requirements, the conditions are analogous.*

A 'guess' consists of the set of disks D_π, and the classified coverage requirements π_p.

Definition 4.5. *A guess π is* consistent *with the optimal solution opt_B if $D_\pi \subseteq opt_B$, and $opt_B \setminus D_\pi$ meets the specified coverage requirement π_p of every point p.*

The important property of the enumeration stage of our algorithm is stated in the following lemma, whose proof is deferred to Section 4.4.

Lemma 4.6. *There exists a polynomial time algorithm that enumerates a set of guesses, such that at least one of the guesses is consistent with opt_B.*

We remark that if a point p is covered in opt_B by one or two disks whose centres lie in the square in which p lies, then our enumeration stage will ensure that the required disks covering p are contained in D_π for the guess π that is consistent with the optimal solution. Therefore, it suffices to consider the case that the remaining coverage requirement of each point p needs to be satisfied by disks with centres outside the square in which p lies.

For every guess π, the aim is to find a solution that is consistent with π

and has small weight. The solution of minimum weight, amongst the feasible solutions for all guesses, is output as the solution.

Lemma 4.7. *There exists a polynomial time algorithm that takes as input a guess π, and either outputs a solution that is consistent with π, or asserts that no solution consistent with π exists. If the guess π is consistent with the optimal solution* $\textsc{opt}_B$*, the solution output by the algorithm has weight at most* $6 \cdot w(opt_B)$*.*

Before providing a proof of Lemma 4.7 we first require some additional definitions. On a high level, the algorithm of Lemma 4.7 is obtained by applying dynamic programming to each horizontal and each vertical strip of squares contained in the block B. As such, we define two sub-problems.

Definition 4.8. *(Horizontal Strip Problem) Consider a horizontal strip H of squares within a block, that consists of K squares S_{ij} for a fixed i, and $0 \leq j \leq K-1$. We are given as input a set P_H of points in the strip, and a set $D_{\bar{H}}$ of disks with centre above or below the strip (i.e., all the disks with centres in the union of the regions* $\textsc{upper} \cup \textsc{lower}$ *for all squares S_{ij} in the strip H). Each disk $d \in D_{\bar{H}}$ is associated with a weight $w(d)$ and a set T_d of event types. Each point $p \in P_H$ has an associated event type t_p, and a classified coverage requirement π_p that can be $\{\Updownarrow\}, \{\updownarrow\}$ or $\{\Updownarrow, \ \Updownarrow\}$. A set $D' \subseteq D_{\bar{H}}$ is a feasible solution if it meets the classified coverage requirements of all points in P_H. The objective function of the* horizontal strip problem *is to compute a feasible solution of minimum weight.*

The *vertical strip problem* is defined analogously. In Section 4.5 we prove the following result.

Lemma 4.9. *There exists a polynomial-time algorithm that computes a minimum-weight solution to any (horizontal or vertical) strip problem (or asserts that no feasible solution exists).*

Using this lemma, we are now in a position to prove Lemma 4.7.

Proof. (of Lemma 4.7)

Let π be an arbitrary guess. One can check whether π admits a feasible solution, because it suffices to check whether the set of all disks D meets the classified coverage requirements of all points in P_B. Therefore, we can assume in the following that there exists a solution, that is consistent with π.

The algorithm solves a vertical strip problem for each of the K vertical strips of K squares contained in the block B, and similarly solves a horizontal strip problem for each of the K horizontal strips of K squares contained within block B. The inputs to the $2K$ strip problems are constructed as follows. For each horizontal strip H, the set of disks in the input to the horizontal strip problem for H consists of all disks from $D \setminus D_\pi$ with centre outside H. Similarly, for every vertical strip of squares V in the block B, the set of disks to the vertical strip problem for V consists of all disks from $D \setminus D_\pi$ with centre outside V.

The points that form the input of a (horizontal or vertical) strip problem are determined as follows. Consider a point $p \in P_B$ that lies in some square S_{ij}, that belongs to a horizontal strip H_p, and a vertical strip V_p. If is it the case that $\pi_p = \{\Leftrightarrow\}$, then p is added with classified coverage requirement $\{\Leftrightarrow\}$ to the vertical strip problem for V_p. Similarly, if $\pi_p = \{\Updownarrow\}$ then p

is added with classified coverage requirement $\{\Updownarrow\}$ to the horizontal strip problem for H_p.

In the case that $\pi_p = \{\Updownarrow, \ \Updownarrow\}$, then p is added with the classified coverage requirement $\{\Updownarrow, \ \Updownarrow\}$ to the horizontal strip problem for H_p. If $\pi = \{\Leftrightarrow, \ \Leftrightarrow\}$, then p is added with the classified coverage requirement $\{\Leftrightarrow, \ \Leftrightarrow\}$ to the horizontal strip problem for V_p.

Additionally, if $\pi_p = \{\Leftrightarrow, \ \updownarrow\}$, then p is added with classified coverage requirement $\{\Leftrightarrow\}$ to the vertical strip problem for V_p, and with classified coverage requirement $\{\updownarrow\}$ to the horizontal strip problem for H_p. Finally, if $\pi_p = \{\leftrightarrow, \ \Updownarrow\}$, then p is added with classified coverage requirement $\{\leftrightarrow\}$ to the vertical strip problem for V_p, and with classified coverage requirement $\{\Updownarrow\}$ to the horizontal strip problem for H_p.

The algorithm of Lemma 4.9 is applied to each of the $2K$ strip problems. If one of the strip problems does not admit a feasible solution, the algorithm outputs that the given guess π does not admit a feasible solution. If all $2K$ strip problems admit feasible solutions, then the union of the $2K$ solutions, together with the set of disks D_π that has been determined to be in the solution by the guess, is then output as the solution. One can observe the algorithm outputs a feasible solution if one exists for the given guess π, and otherwise asserts correctly that no feasible solution exists.

Now consider a guess π, that is consistent with opt_B. Let $opt_B(H)$ be the subset of $opt_B \setminus D_\pi$ consisting of disks that are in UPPER $\cup$ LOWER for a horizontal strip H, and overlap H. Similarly, let $opt_B(V)$ be the subset of $opt_B \setminus D_\pi$ consisting of disks that are in LEFT $\cup$ RIGHT for a horizontal strip V, and overlap V.

Observe that $opt_B(V)$ and $opt_B(H)$ form feasible solutions to the strip problems for the strips V and H, respectively. Hence, the solutions $A(V)$ and $A(H)$ calculated by the algorithm from Lemma 4.9, for the strips V and H respectively, have costs at most $w(opt_B(V))$ and $w(opt_B(H))$. The cost of the solution output by the algorithm is therefore at most

$$\begin{aligned} alg_B &= w(D_\pi) + \sum_H w(A(H)) + \sum_V w(A(V)) \\ &\leq w(D_\pi) + \sum_H w(opt_B(H)) + \sum_V w(opt_B(V)) \\ &\leq 6 \cdot w(opt_B) \end{aligned}$$

The last inequality follows as each disk d in $opt_B \setminus D_\pi$ is in $opt_B(H)$ for at most three horizontal strips, and in $opt_B(V)$ for at most three vertical strips. For vertical strips, this is because a disk of diameter 4 intersects at most four (consecutive) vertical strips of width 1.4, and in one of them the disk has centre inside the strip, and therefore cannot be in LEFT $\cup$ RIGHT. For horizontal strips the reasoning is analogous. Hence, the weight of each disk in $opt_B \setminus D_\pi$ is counted at most 6 times on the left-hand side of the last inequality, and the weight of each disk in $D_\pi \subseteq opt_B$ only once. $\square$

By combining Lemmas 4.6 and 4.7, we obtain the following lemma.

Lemma 4.10. *There is a 6-approximation algorithm for instances of W2CUD-T where all points lie in a $K \times K$ block.*

As discussed in Section 4.2.2, a 6-approximation for *W2CUD-T* in a block implies a $(6+\varepsilon)$-approximation algorithm for general instances of *W2CUD-T*. Thus, Lemma 4.10 implies the following result.

Theorem 4.11. *For every fixed $\varepsilon > 0$, there is a $(6+\varepsilon)$-approximation algorithm for the problem W2CUD-T.*

4.4 Guessing Properties of the Optimal Solution by Enumeration

In this section we prove Lemma 4.6. Recall that opt_B denotes an arbitrary fixed optimal solution to the given instance of *W2CUD-T* in a block B. We present the enumeration technique using the notion of 'guessing'. When we write that the algorithm '*guesses*' a property of opt_B, this means that the algorithm enumerates all possibilities for that property in such a way that one of the possibilities is guaranteed to be the desired property of the optimal solution. The enumeration is done separately for each of the K^2 squares S_{ij}. Let m denote the number of disks that overlap S_{ij}. Let the left and right boundary of S_{ij} lie on the line $x = x_1$ and $x = x_2$, respectively, and the bottom and top boundary on the line $y = y_1$ and $y = y_2$, respectively. See Figure 4.1 for an illustration.

4.4.1 Disks with Centre in S_{ij}

First, for every event type $t_l \in T$, we guess whether opt_B contains $0, 1$ or 2 disks with centre in S_{ij} that monitor event type t_l. Furthermore, in the second and third case we also guess one or two such disks, respectively. Note that a disk with centre in S_{ij} must contain the whole square S_{ij}, as the disk has radius $r = 2$ and the square has side length less that $\sqrt{2}$. Furthermore, two disks with centre in S_{ij} that cover event type t_l are sufficient to meet the coverage requirements of all points $p \in P_{ij}$ with $t_p = t_l$. Hence, for these points, it is not necessary to guess more than two disks. All the disks that are guessed in this way form the set D_π, and the coverage requirements of all points that are covered by disks in D_π are reduced accordingly. In the following, we use k_p to denote the remaining coverage requirement of a point p.

4.4.2 Overview of Square Partition

Next, we aim at guessing a partition of the square S_{ij} into areas such that for points in the same area, we know whether or not they are covered twice by UPPER ∪ LOWER (here and in the following, all regions are specified with respect to the square S_{ij}), twice by LEFT∪RIGHT, or once by UPPER∪LOWER (possibly restricted to UM∪LM) and once by LEFT∪RIGHT (possibly restricted to CL∪CR). In other words, we want to guess the classified coverage requirements of all points. We guess a separate such square partition for each event type $t_l \in T$. For each $t_l \in T$, the steps involved in guessing the partition are as follows. First, we determine areas called *2-watching t_l sandglasses* (the terminology is motivated be a similar sandglass concept in [HGZW09]) for the four regions UM, LM, CL and CR. For each of these areas, we can require that points are covered only by UPPER ∪ LOWER (classified coverage requirement $\{\Updownarrow\}$ if $k_p = 1$ or $\{\Updownarrow, \quad \Updownarrow\}$ if $k_p = 2$) or only by LEFT ∪ RIGHT (classified coverage requirement $\{\Leftrightarrow\}$ if $k_p = 1$ or $\{\Leftrightarrow, \quad \Leftrightarrow\}$ if $k_p = 2$). Secondly, we consider 1-watching envelopes, i.e., envelopes of the disks in

opt_B with centre in one of the regions UM, LM, CL and CR, and guess the four points where adjacent envelopes intersect. Based on the locations of these intersection points, we can segment the square into smaller areas and deduce for each of the smaller areas whether the points located in the area are to be watched from UPPER $\cup$ LOWER, or from LEFT $\cup$ RIGHT (or from both). The details of the partition of the square into areas for one specific event type t_l are presented in the following subsections.

4.4.3 2-Watching Sandglasses

We define the *2-watching t_l sandglass* for the region LM. The sandglasses for the regions UM, CR and CL are defined similarly by rotation.

Let P'_{ij} be the set of points in P_{ij} that have event type t_l, i.e., the set of all points p in the square S_{ij}, where $t_p = t_l$. Consider the set $P^2_{\text{LM}} \subseteq P'_{ij}$ of all points in P'_{ij} that are covered by two distinct disks, with centre in LM in opt_B, but that are not covered by any disk from opt_B that does not have centre in LM. For each $p \in P^2_{\text{LM}}$, consider a line l_p through p with slope 1 and let p' be the point where l_p intersects the line $y = y_1$. Let p_l be the point in P^2_{LM} for which p' is leftmost. Similarly, let l'_p be the line through p with slope -1 and let p'' be the intersection point of l'_p and $y = y_1$. Let p_r denote the point in P^2_{LM} for which p'' is rightmost. The 2-watching t_l sandglass for LM is now defined as the area that is obtained as the intersection of the halfplane below l_{p_l}, the halfplane below l'_{p_r}, and the square S_{ij}. The reader can refer to Figure 4.2 for an illustration (we remark for the reader that in order to better illustrate the concepts described, the figures presented in this chapter are not to scale). Note that this sandglass is uniquely determined by p_l and

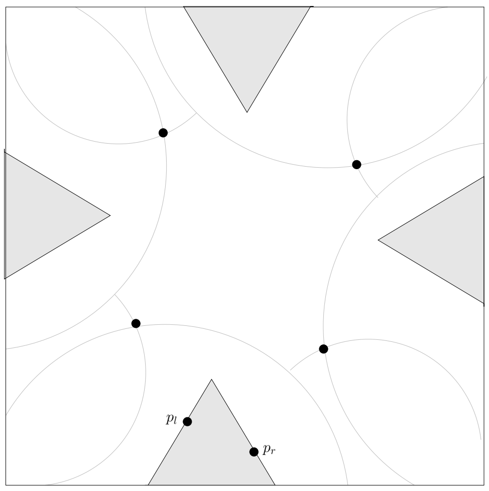

Figure 4.2: 2-watching sandglasses.

p_r and there are $\mathcal{O}(|P'_{ij}|^2)$ possibilities for guessing p_l and p_r.

We show that the coverage requirements of any point of P'_{ij} located in the 2-watching t_l sandglass for LM are met by disks from UPPER∪LOWER. Define the *lower-shadow* of a point p to be the region that is the intersection of the half-plane below the line with slope -1 through p, the half-plane below the line with slope 1 through p, and the square S_{ij}. The *left-shadow*, *up-shadow*, and *right-shadow* of a point are defined analogously. We state the following lemma due to Huang et al. [HGZW09].

Lemma 4.12. [HGZW09] *If a point $p \in P'_{ij}$ is covered by a disk d from* LM*, then any point in the lower-shadow of p, is also covered by the same disk from* LM*.*

The lemma directly implies the following corollary.

Corollary 4.13. *If a point $p \in P'_{ij}$ is covered by two disks d_1, d_2 from* LM*, then any point in the lower-shadow of p is also covered by the same two disks*

d_1, d_2 from LM.

We also require the following lemma, which we prove by application of Lemma 4.12.

Lemma 4.14. *The coverage requirement for any point $p \in P'_{ij}$ that lies inside the 2-watching t_l sandglass for* LM *is met by disks in opt_B from* UPPER $\cup$ LOWER*.*

Proof. Consider the points p_l and p_r defining the 2-watching t_l sandglass for LM. For points in the lower-shadow of p_l or in the lower-shadow of p_r, the lemma follows from Corollary 4.13. Let p be a point that lies in the 2-watching t_l sandglass for LM, but not in the shadows of p_l or p_r. Assume that p is covered by a disk d with centre in CL or CR by opt_B. Then, by Lemma 4.12 applied to the left-shadow or right-shadow of p, respectively, we find that p_l or p_r is also covered by d, a contradiction to the choice of p_l and p_r. $\square$

It follows that the coverage requirements of all points in the 2-watching t_l sandglasses for LM and for UM are satisfied by disks in opt_B from the regions UPPER $\cup$ LOWER, and the coverage requirements of all points in the 2-watching t_l sandglasses for CL and CR are satisfied by disks in opt_B from LEFT $\cup$ RIGHT. Hence, all the points from P'_{ij} that lie in 2-watching t_l sandglasses can be classified accordingly (classified coverage requirements $\{\Leftrightarrow\}$, $\{\Leftrightarrow, \ \Leftrightarrow\}$, $\{\Updownarrow\}$, or $\{\Updownarrow, \ \Updownarrow\}$). These points are ignored for the classifications of points described in the following subsections, i.e., their classification is not changed if they are contained in one of the areas under consideration there.

4.4.4 1-Watching Envelopes

It remains to describe the case with points from P'_{ij} that do not lie in any of the 2-watching t_l sandglasses. For each of the four regions UM, LM, CL and CR, we define a 1-watching t_l envelope as follows (see Figure 4.3 for an illustration). Note that we have separate envelopes for each $t_l \in T$.

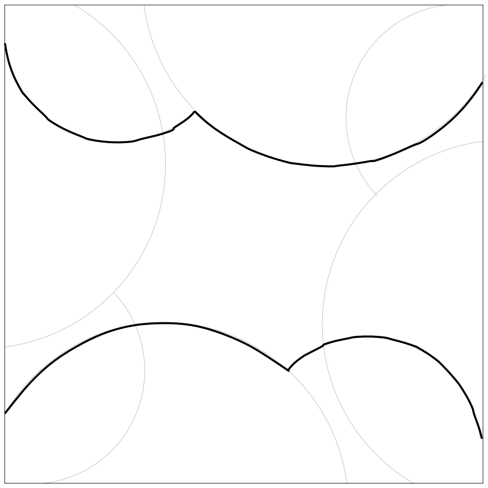

Figure 4.3: 1-watching envelopes for LM and UM.

Definition 4.15. *Consider a square S_{ij}, and let R be one of the regions* UM*,* LM*,* CL *and* CR *with respect to S_{ij}. The* 1-watching t_l envelope for region R *is the intersection of the square S_{ij} and the boundary of the union of the disks in opt_B that monitor event type t_l and have centre in the region R. The respective boundary of the square (i.e., the top boundary for the 1-watching t_l envelope for* UM*, the right boundary for the 1-watching t_l envelope for* CR*, etc.) is used to fill in parts of the envelope where no disk from the respective region intersects the square.*

4.4.5 Intersection Points of 1-Envelopes

We call the 1-watching envelopes of CL and UM *adjacent* and similarly those of UM and CR, etc. Allow us to define *intersection points* of adjacent 1-watching envelopes and describe how they are used to partition the remainder of the square S_{ij} into areas such that we can specify for the points in each area, whether they are covered by opt_B using disks in UPPER $\cup$ LOWER, or disks in LEFT $\cup$ RIGHT.

Lemma 4.16. *Each pair of adjacent 1-watching t_l envelopes intersects in exactly one point.*

Proof. By the dimension of the square S_{ij} and the radius of the disks, it follows that the direction of any tangent to the 1-watching t_l envelope for UM or LM is in the open interval $(-\pi/4, \pi/4)$, and the direction of any tangent to the 1-watching t_l envelope for CL or CR is in the open interval $(\pi/4, 3\pi/4)$. Therefore, it is impossible that two adjacent 1-watching t_l envelopes have more than one intersection point. As each envelope is a curve connecting points on opposite sides of the square, it follows that two adjacent envelopes always intersect. $\square$

Hence there are four, not necessarily distinct, intersection points between adjacent 1-watching t_l envelopes. Note that each of these intersection points is uniquely specified by the two disks whose boundaries intersect at that point. Hence, it suffices to guess eight disks in order the determine the four intersection points in a fixed optimal solution.

Let the intersection point of the 1-watching t_l envelopes for CL and UM be denoted by $i_{\mathrm{CL}}^{\mathrm{UM}}$. Similarly, let $i_{\mathrm{CR}}^{\mathrm{UM}}$ be the intersection point of the 1-watching t_l envelopes for UM and CR, $i_{\mathrm{CL}}^{\mathrm{LM}}$ the intersection point of the 1-watching t_l envelopes for LM and CL, and $i_{\mathrm{CR}}^{\mathrm{LM}}$ the intersection point of the 1-watching t_l envelopes for LM and CR.

4.4.6 Areas in a Square

Assume that $i_{\mathrm{CL}}^{\mathrm{UM}}$ is to the left of $i_{\mathrm{CR}}^{\mathrm{UM}}$ (i.e., has smaller x-coordinate), $i_{\mathrm{CR}}^{\mathrm{UM}}$ is above $i_{\mathrm{CR}}^{\mathrm{LM}}$ (i.e., has larger y-coordinate), $i_{\mathrm{CR}}^{\mathrm{LM}}$ is to the right of $i_{\mathrm{CL}}^{\mathrm{LM}}$, and $i_{\mathrm{CL}}^{\mathrm{LM}}$ is below $i_{\mathrm{CL}}^{\mathrm{UM}}$. We call this the *standard configuration*, and we will describe alternative configurations in subsequent sections.

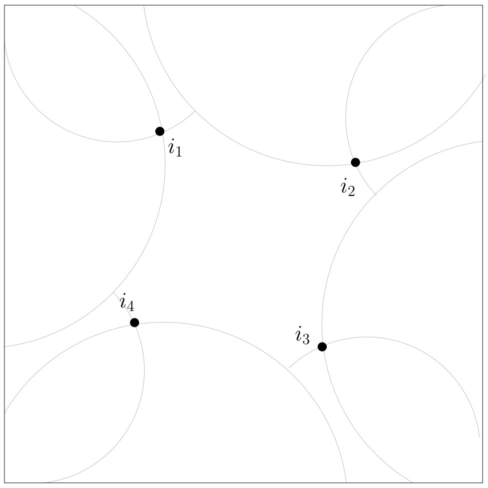

Figure 4.4: Standard configuration of intersection points.

For the standard configuration define i_1 to be $i_{\mathrm{CL}}^{\mathrm{UM}}$, let i_2 be $i_{\mathrm{CR}}^{\mathrm{UM}}$, let i_3 be $i_{\mathrm{CR}}^{\mathrm{LM}}$ and let i_4 be $i_{\mathrm{CL}}^{\mathrm{LM}}$, as illustrated in Figure 4.4. Note that when we discuss alternative configurations of intersection points in later section, the correspondence of $i_1, \ldots, i_4$ to intersection points will vary. For an arbitrary intersection point i_s, where $1 \leq s \leq 4$, let l_s and l'_s be the two lines through i_s with slope 1 and -1 respectively. These lines allow us to define the following areas, for which we can make further deductions regarding the disks covering points in these areas.

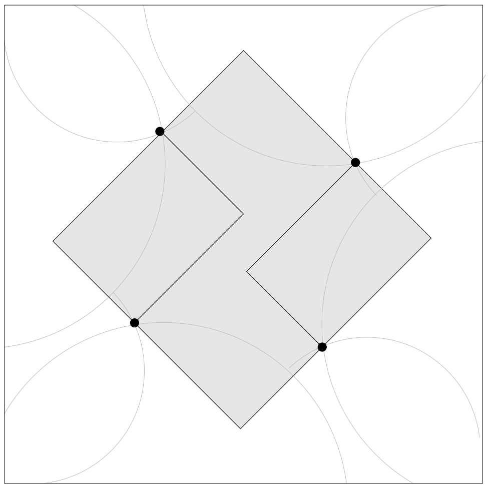

Figure 4.5: MIDDLE region.

As shown in Figure 4.5, define MIDDLE to be the area that is the intersection of the halfplanes below l_1 and l'_2 and the halfplanes above l_3 and l'_4. As illustrated in Figure 4.6, let MIDDLE-L be the area that is the intersection of the halfplanes below l_1 and l'_1, and the halfplanes above l_4 and l'_4. Similarly, let MIDDLE-R be the area that is the intersection of the halfplanes below l_2 and l'_2 and the halfplanes above l_3 and l'_3.

We now make the claim that the coverage requirements of all points in the areas MIDDLE-L and MIDDLE-R are met in opt_B by disks with centre in LEFT $\cup$ RIGHT. Without loss of generality, we state the arguments only for points within the area MIDDLE-L as identical arguments apply for MIDDLE-R.

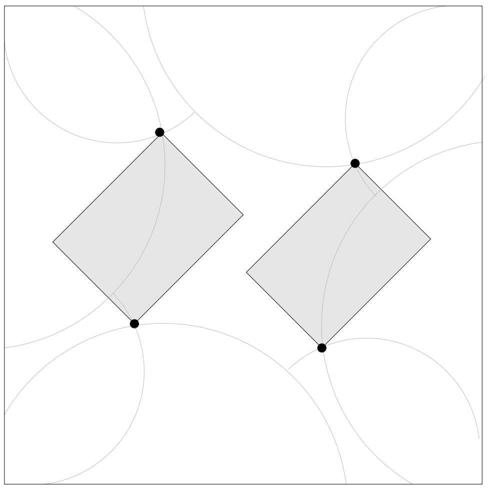

Figure 4.6: MIDDLE-L and MIDDLE-R regions.

Observe that the area MIDDLE-L lies entirely below the 1-watching t_l envelope for UM. This follows as MIDDLE is contained within the 90 degree cone below i_1 that lies between l_1 and l'_1, while the 1-watching t_l envelope for UM lies in the union of the halfplanes above l_1, and above l'_1. Similarly, MIDDLE-L lies entirely above the 1-watching t_l envelope for LM. Therefore, it is not possible that a point in MIDDLE-L is covered by a disk with centre in UM or LM in opt_B. As such, the coverage requirements of all points in MIDDLE-L are indeed met in opt_B by disks from LEFT$\cup$RIGHT. Thus, we can classify the points from P'_{ij} that lie in MIDDLE-L and MIDDLE-R accordingly (i.e., such a point p is assigned classified coverage requirement $\{\Leftrightarrow\}$ if $k_p = 1$ and $\{\Leftrightarrow, \ \Leftrightarrow\}$, in the case that $k_p = 2$).

The areas MIDDLE-U and MIDDLE-D are defined analogously to MIDDLE-L and MIDDLE-R, with MIDDLE-U enclosed by the lines $l_1, \ l'_1, \ l_2$ and l'_2, and similarly the area MIDDLE-D is enclosed by the lines l_3, l'_3, l_4 and l'_4. This

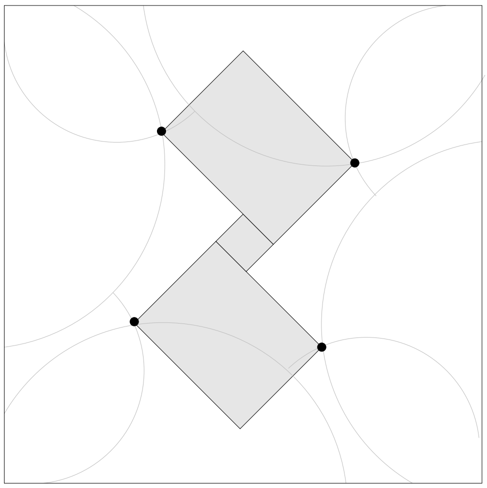

Figure 4.7: MIDDLE-U and MIDDLE-M and MIDDLE-D regions.

is illustrated in Figure 4.7. Furthermore, the region obtained by removing MIDDLE-L, MIDDLE-U, MIDDLE-R and MIDDLE-D from MIDDLE (that may be empty) is denoted by MIDDLE-M. By analogous arguments to those given above for the case of MIDDLE-L, the coverage requirements of points in areas MIDDLE-U and MIDDLE-D are met in opt_B by disks from UPPER $\cup$ LOWER. Hence, the points in these areas can be classified accordingly (i.e., such a point p is assigned classified coverage requirement $\{\Updownarrow\}$ if $k_p = 1$, and $\{\Updownarrow, \ \Updownarrow\}$ if $k_p = 2$).

Finally, the area MIDDLE-M lies outside all four 1-watching t_l envelopes, and so the points in that area can only be covered in opt_B by disks from regions UL, UR, LL and LR. These regions are in UPPER $\cup$ LOWER and in LEFT$\cup$RIGHT, so we can (arbitrarily giving preference to the former) classify these points as points that have to be covered by disks in UPPER $\cup$ LOWER (i.e., such a point p is assigned classified coverage requirement $\{\Updownarrow\}$ if $k_p = 1$,

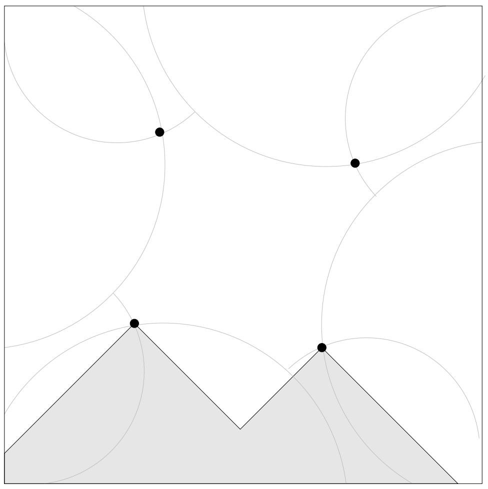

Figure 4.8: SOUTH area of S_{ij}.

and $\{\Updownarrow, \;\; \Updownarrow\}$ if $k_p = 2$).

We refer to the part of S_{ij} that is not in MIDDLE as the *peripheral part of* S_{ij}. Consider the peripheral area SOUTH shown in Figure 4.8. This is the area that is the lower-shadow of i_3, and the lower-shadow of i_4. Recall that points in SOUTH that are also in a 2-watching t_l sandglass have already been classified and are not considered further. For any remaining point p located in SOUTH we know that p is covered by a disk from LM (as SOUTH lies below the 1-watching t_l envelope for LM). Furthermore, if p has to be covered by a second disk, we know that p is covered by at least one disk whose centre is not in LM because otherwise p would lie in the 2-watching t_l sandglass for LM and would have been classified already. That second disk covering p cannot have centre in UM, because SOUTH lies entirely below the 1-watching t_l envelope for UM.

Hence if $k_p = 1$, we specify that p must be covered by UPPER $\cup$ LOWER, i.e., p has classified coverage requirement $\{\Updownarrow\}$, and in the case $k_p = 2$, we specify that p must be covered once by LEFT $\cup$ RIGHT and once by LM $\cup$ UM, i.e., has classified coverage requirement $\{\Leftrightarrow, \updownarrow\}$.

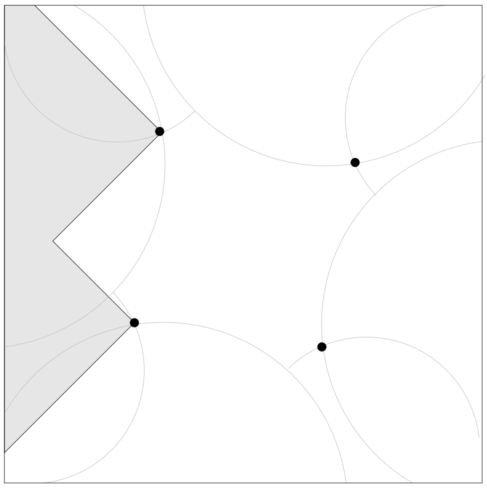
Figure 4.9: WEST area of S_{ij}.

The areas WEST (illustrated in Figure 4.9), NORTH and EAST are defined and handled analogously. As each point in P'_{ij} is contained in one of the areas defined above, all these points are classified, i.e., for every point p, we have determined and assigned a classified coverage requirement π_p.

4.4.7 Areas in a Square - Other Configurations

Recall that $i^{\text{UM}}_{\text{CL}}$ is the intersection point between the 1-watching t_l envelopes for UM and CL, and $i^{\text{LM}}_{\text{CL}}$, $i^{\text{LM}}_{\text{CR}}$ and $i^{\text{UM}}_{\text{CR}}$ are defined analogously. In the previous subsection, we assumed the standard configuration as shown in Figure 4.4, i.e., $i^{\text{UM}}_{\text{CL}}$ is to the left of $i^{\text{UM}}_{\text{CR}}$, $i^{\text{UM}}_{\text{CR}}$ is above $i^{\text{LM}}_{\text{CR}}$, $i^{\text{UM}}_{\text{CL}}$is above $i^{\text{LM}}_{\text{CL}}$, and $i^{\text{LM}}_{\text{CL}}$ is to the left of $i^{\text{LM}}_{\text{CR}}$. In the following, we discuss how to handle all other possible configurations.

We note that the definition of 2-watching t_l sandglasses does not depend on the configuration of intersection points, so 2-watching sandglasses are handled as before. The definition of the peripheral areas is adapted as follows. The area SOUTH is the union of the lower-shadow of the lower of the two points $i_{\text{CL}}^{\text{UM}}$ and $i_{\text{CL}}^{\text{LM}}$ and the lower-shadow of the lower of the two points $i_{\text{CR}}^{\text{UM}}$ and $i_{\text{CR}}^{\text{LM}}$. The area WEST is the union of the left-shadow of the point that is further left among the two points $i_{\text{CL}}^{\text{UM}}$ and $i_{\text{CR}}^{\text{UM}}$ and the left-shadow of the point that is further left among the two points $i_{\text{CL}}^{\text{LM}}$ and $i_{\text{CR}}^{\text{LM}}$. The definitions of EAST and NORTH are analogous.

We observe that each of the four peripheral areas can be handled in the same way as in the standard configuration. For example, it is still the case that every point in SOUTH is covered by a disk with centre in LM, and if the point is required to be covered by a second disk, there exists a disk covering it with centre in LEFT $\cup$ RIGHT.

Let us introduce some additional terminology. We say that the 1-watching t_l envelopes for CL and CR are *opposite*, and similarly the envelopes for UM and LM are opposite. Furthermore, we say that the 1-watching t_l envelopes for CL and CR *overlap* if $i_{\text{CL}}^{\text{UM}}$ is to the right of $i_{\text{CR}}^{\text{UM}}$ and $i_{\text{CL}}^{\text{LM}}$ is to the right of $i_{\text{CR}}^{\text{LM}}$.

Similarly, we say that the 1-watching t_l envelopes for CL and CR *cross* if only one of the two relations is reversed compared to the standard configuration. For the 1-watching t_l envelopes for UM and LM, the notions of overlapping and crossing are defined analogously by considering the relations $i_{\text{CL}}^{\text{UM}}$ and $i_{\text{CL}}^{\text{LM}}$ and between $i_{\text{CR}}^{\text{UM}}$ and $i_{\text{CR}}^{\text{LM}}$. Additionally, let in the following areas

MIDDLE, MIDDLE-L, etc. be defined in the terms of the respective choices for i_1, i_2, i_3 and i_4 in the same way as the standard configuration in Figure 4.4.

A Pair of Opposite Envelopes Overlap

The first alternative configuration that we consider is the case where at least one pair of opposite envelopes overlap. Without loss of generality, assume that the 1-watching t_l envelopes for UM and LM overlap.

Note that the case where the 1-watching t_l envelopes for CL and CR overlap is symmetric. It is the case that $i_{\mathrm{CL}}^{\mathrm{LM}}$ is above $i_{\mathrm{CL}}^{\mathrm{UM}}$, and $i_{\mathrm{CR}}^{\mathrm{LM}}$ is above $i_{\mathrm{CR}}^{\mathrm{UM}}$. In each of the following cases, the choice of i_1, i_2, i_3 and i_4 will ensure that the areas MIDDLE-M MIDDLE-L and MIDDLE-R are enclosed within the 1-watching t_l envelopes for both UM and LM i.e., the coverage requirement for points in these areas can be classified as $\{\Updownarrow\}$ or $\{\Updownarrow, \ \Updownarrow\}$.

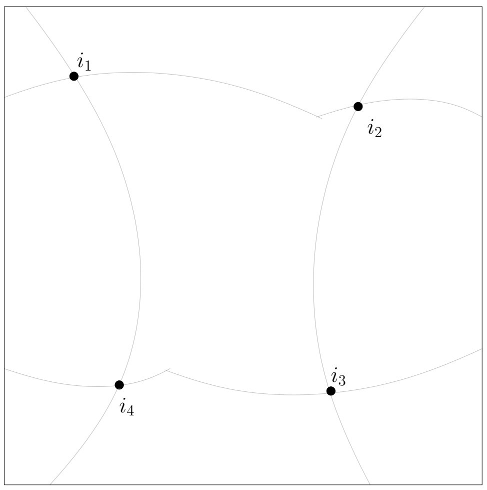

Figure 4.10: The UM and LM envelopes overlap.

Case 1: The other pair of envelopes are standard. In this case the envelope for CL and the envelope for CR are standard, i.e., they neither cross nor overlap. This case is illustrated in Figure 4.10. Let i_1 denote $i^{\text{LM}}_{\text{CL}}$ i_2 denote $i^{\text{LM}}_{\text{CR}}$, i_3 denote $i^{\text{UM}}_{\text{CR}}$ and let i_4 denote $i^{\text{UM}}_{\text{CL}}$. The areas MIDDLE-U and MIDDLE-D are always outside the 1-watching t_l envelopes for CL and CR, and therefore, as is the case in the standard configuration, the points in these areas can be classified by the coverage requirement $\{\Updownarrow\}$ or $\{\Updownarrow, \ \Updownarrow\}$.

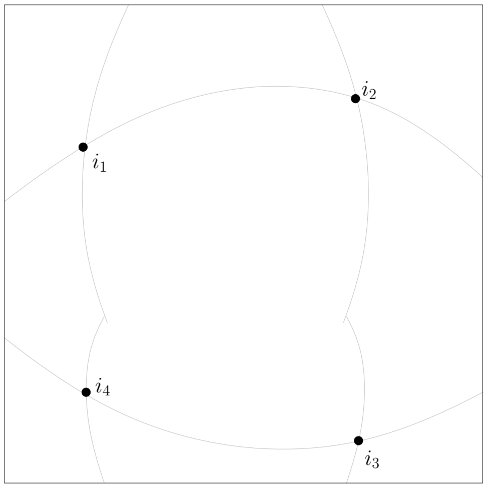

Figure 4.11: The UM and LM envelopes overlap, and CL and CR envelopes overlap.

Case 2: The other pair of envelopes also overlap. As illustrated in Figure 4.11, we have that the 1-watching envelopes for CL and CR overlap, and the 1-watching t_l envelopes for UM and LM also overlap. In this case let i_1 denote $i^{\text{LM}}_{\text{CR}}$, i_2 denote $i^{\text{LM}}_{\text{CL}}$, i_3 denote $i^{\text{UM}}_{\text{CL}}$ and let i_4 denote $i^{\text{UM}}_{\text{CR}}$. The points within the areas MIDDLE-U and MIDDLE-D are covered twice by disks in LEFT $\cup$ RIGHT as those areas are enclosed within the 1-watching t_l envelopes for both CL and CR. These points can be classified by the coverage requirement $\{\Leftrightarrow\}$ or $\{\Leftrightarrow, \ \Leftrightarrow\}$.

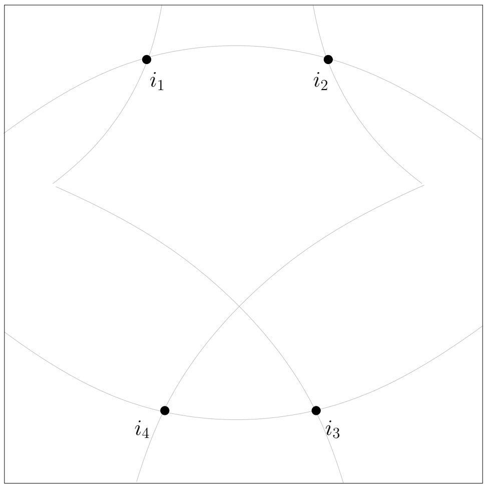

Figure 4.12: The UM and LM envelopes overlap, and CR and CL envelopes cross.

Case 3: The other pair of envelopes cross. Without loss of generality, assume that $i^{\mathrm{UM}}_{\mathrm{CL}}$ is to the right of $i^{\mathrm{UM}}_{\mathrm{CR}}$ and $i^{\mathrm{LM}}_{\mathrm{CL}}$ is to the left of $i^{\mathrm{LM}}_{\mathrm{CR}}$, as illustrated in Figure 4.12. Let i_1 denote $i^{\mathrm{LM}}_{\mathrm{CL}}$, i_2 denote $i^{\mathrm{LM}}_{\mathrm{CR}}$, i_3 denote $i^{\mathrm{UM}}_{\mathrm{CL}}$ and let i_4 denote $i^{\mathrm{UM}}_{\mathrm{CR}}$. The points in MIDDLE-U can be handled as in case 1 where the other pair of envelopes are standard, and the points in MIDDLE-D as in case 2 where the other pair of envelopes also overlap. Note that in this case it must be that the areas MIDDLE-U and MIDDLE-D are disjoint.

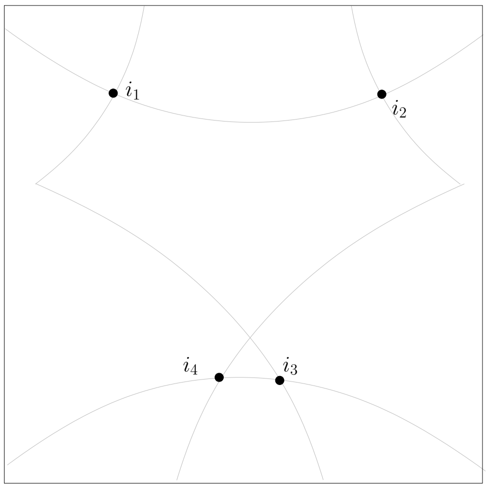

Figure 4.13: CL envelope crosses CR envelope.

One Pair of Opposite Envelopes Cross, The Other Pair of Envelopes are Standard

Without loss of generality, assume that the 1-watching t_l envelopes for CL and CR cross, and that $i_{\text{CL}}^{\text{UM}}$ is to the left of $i_{\text{CR}}^{\text{UM}}$ and $i_{\text{CL}}^{\text{LM}}$ is to the right of $i_{\text{CR}}^{\text{LM}}$ as illustrated in Figure 4.13. Note that the other cases will be symmetric. Let i_1 denote $i_{\text{CL}}^{\text{UM}}$, i_2 denote $i_{\text{CR}}^{\text{UM}}$, i_3 denote $i_{\text{CL}}^{\text{LM}}$ and finally let i_4 denote $i_{\text{CR}}^{\text{LM}}$. As in the standard configuration the areas MIDDLE-L, MIDDLE-R and MIDDLE-M are outside the 1-watching t_l envelopes for UM and LM, i.e., we can classify the coverage requirements for the points in these areas as $\{\Leftrightarrow\}$ or $\{\Leftrightarrow, \ \Leftrightarrow\}$. For a similar reason, the coverage requirements for the points in MIDDLE-U can be classified as $\{\Updownarrow\}$ or $\{\Updownarrow, \ \Updownarrow\}$. The area MIDDLE-D is within the 1-watching envelopes for both CL and CR, so the coverage requirements for points in MIDDLE-D can be classified as $\{\Leftrightarrow\}$ or $\{\Leftrightarrow, \ \Leftrightarrow\}$.

Both Pairs of Opposite Envelopes Cross

The only configuration that has not been considered is the case where both pairs of opposite envelopes cross. One such configuration would have $i^{\text{UM}}_{\text{CL}}$ strictly to the left of $i^{\text{UM}}_{\text{CR}}$, $i^{\text{LM}}_{\text{CL}}$ strictly to the right of $i^{\text{LM}}_{\text{CR}}$, $i^{\text{UM}}_{\text{CL}}$ strictly above $i^{\text{LM}}_{\text{CL}}$, and $i^{\text{UM}}_{\text{CR}}$ strictly below $i^{\text{LM}}_{\text{CR}}$. We remark that one can assume strict relationships since in the case where two points coincide, we are free to arbitrarily choose the relation and thus arrive at a previously considered configuration, where it is not the case that both pairs of envelopes cross.

To show that the aforementioned configuration cannot occur, consider the line l of slope -1 through $i^{\text{UM}}_{\text{CL}}$. Since $i^{\text{UM}}_{\text{CL}}$ is above $i^{\text{LM}}_{\text{CL}}$ and both points lie on the CL envelope, we get that $i^{\text{LM}}_{\text{CL}}$ is strictly below the line l. Since, $i^{\text{UM}}_{\text{CL}}$ is on the line l and $i^{\text{UM}}_{\text{CR}}$ is to the right of $i^{\text{UM}}_{\text{CL}}$ and also lies on the UM envelope, we get that $i^{\text{UM}}_{\text{CR}}$ is above l. Since $i^{\text{UM}}_{\text{CR}}$ is below $i^{\text{LM}}_{\text{CR}}$ and both points are on the CR envelope, it follows that $i^{\text{LM}}_{\text{CR}}$ is above l. Since, $i^{\text{LM}}_{\text{CL}}$ is to the right of $i^{\text{LM}}_{\text{CR}}$ and both points are on the LM envelope, we obtain that $i^{\text{LM}}_{\text{CL}}$ is strictly above the line l. This is a contradiction to the conclusion that $i^{\text{LM}}_{\text{CL}}$ is strictly below l that we derived above. Hence, this configuration is not possible. The other configurations where both envelopes cross can be excluded similarly.

4.4.8 Complexity of Enumeration

For each of the K^2 squares S_{ij} in a block B, and for each event type $t_l \in T$, we enumerate at most two disks with centre in S_{ij} (these form the set D_π of disks that are determined to be in the solution by the guessing stage), eight points defining the 2-watching sandglasses, and four intersection points (each identified by two disks) of the 1-watching envelopes. Hence, if there are m disks and n points in total, there are $\mathcal{O}((m^2n^8m^8)^{|T|}) = (m+n)^{\mathcal{O}(|T|)}$ choices per square. Thus, in total there are $(m+n)^{\mathcal{O}(K^2\cdot|T|)}$ choices for the whole block B. Since K and $|T|$ are constants, this is a polynomial number of choices. This concludes the proof of Lemma 4.6.

4.5 Dynamic Programming Algorithm

In this section we prove Lemma 4.9 by providing a dynamic programming algorithm for the (horizontal and vertical) strip problems. Vertical strip problems can be solved analogously as horizontal strip problems, via rotation of the plane by 90 degrees. As such, we only describe the algorithm with respect to the horizontal strip problems in this section.

4.5.1 Input to the Dynamic Program

Let an instance of the horizontal strip problem for strip H be given as described in Definition 4.8. We let n_H denote the number of points in P_H. For convenience of presentation, we extend the strip H by two empty squares on the left side, and an additional two empty squares immediately to the right of H. This is for technical reasons, specifically such that now every disk in $D_{\bar{H}}$ that covers some point in P_H has its corresponding centre in the region UM or in the region LM, with respect to some square S in the strip H.

We note for the reader that we do not consider instances for which a feasible solution does not exist. Such an instance could easily be detected by an algorithm and as such, is superfluous to our analysis. In the following we only consider the case that there exists a feasible solution, i.e, a set of disks $D' \subseteq D_{\bar{H}}$ that meets the classified coverage requirements of all points in P_H. The goal is now to compute a feasible solution of minimum weight for the given instance of the horizontal strip problem.

Let the points in $P_H = \{p_1, \ldots, p_{n_H}\}$ be ordered by non-decreasing x-coordinates, i.e., for every $1 \leq i \leq n_H$ we have $x_{p_{i-1}} \leq x_{p_i}$. If two points have the same x-coordinate then we order them arbitrarily.

4.5.2 Outer and Inner Envelopes

For a given square S forming part of a strip H, let UM(S) denote the region UM with respect to S. Similarly, LM(S) denotes the region LM with respect to S. Let $T' \in \mathcal{P}(T) \setminus \{\emptyset\}$ be an arbitrary non-empty combination of event types in T.

We now continue by defining outer and inner envelopes formed by those disks forming part of a solution, that have centres in UM(S) or LM(S), and monitor event types in T'. The purpose of these envelopes is to represent the disks lying in a particular region that cover some point from P_H, in the sense that any point in P_H that is covered once or twice by disks in that region, is also covered at least that same number of times by disks that are part of the two envelopes of that region. Our dynamic programming algorithm then has the objective of computing envelopes corresponding to a solution of minimum cost.

Definition 4.17. (Outer T' envelope for UM(S)) Let S be a square in the horizontal strip H, let $T' \in \mathcal{P}(T) \setminus \{\emptyset\}$, and let D be a set of disks. The set of disks $d \in D$ that have centre in $\textsc{um}(S)$ and satisfy $T_d = T'$ is denoted by $D^{T'}_{\textsc{um}(S)}$. The *outer T' envelope for* $\textsc{um}(S)$ (for the set of disks D) is the intersection of the boundary of the union of all disks in $D^{T'}_{\textsc{um}(S)}$ with the strip H.

If at some x-coordinate there is no disks from $D^{T'}_{\textsc{um}(S)}$ that overlaps H, we let the upper boundary of H form a part of the envelope. A disk is said to be *on the envelope* if the boundary of the disk forms part of the envelope that consists of more than a single point. The set of disks on the envelope is denoted by $\bar{D}^{T'}_{\textsc{um}(S)}$.

We also require inner envelopes, which we define as follows.

Definition 4.18. (Inner T' envelope for UM(S)) The *inner T' envelope for* $\textsc{um}(S)$, for a set of disks D, is defined to be the outer T' envelope for $\textsc{um}(S)$ for the set of disks $D \setminus \bar{D}^{T'}_{\textsc{um}(S)}$. The set of disks on the inner T' envelope for $\textsc{um}(S)$ is denoted by $\bar{D}^{T'}_{I(\textsc{um}(S))}$.

The outer and inner T' envelopes for $\textsc{um}(S)$ in some fixed optimal solution are denoted by $opt^{T'}_{\textsc{um}(S)}$ and $opt^{T'}_{I(\textsc{um}(S))}$ respectively.

The outer and inner T' envelopes for $\textsc{lm}(S)$, for a set of disks D, are defined and denoted analogously, namely by considering disks with centre in $\textsc{lm}(S)$ instead of $\textsc{um}(S)$, and substituting $\textsc{um}(S)$ by $\textsc{lm}(S)$ in the notation.

For a given set D of disks, these definitions give us four different envelopes for each square S, and for each set T' of event types. We view the set of disks on each of these envelopes as ordered by non-decreasing x-coordinates of the centres of the disks. Note that we can assume without loss of generality that no two disks on an envelope have the same centre.

Note that each disk from D is on at most one envelope. Furthermore, if we trace an envelope from left to right, the disks of the envelope appear on the envelope in the order of increasing x-coordinates of their centres, and each disk appears on an envelope at most once, following from the fact that all disks have the same radius.

4.5.3 Dynamic Programming Table

We have a table W_{p_i} for every point $p_i \in P_H$. Let S be the square in which p_i lies. Let S^- and S^+ be the adjacent squares to the left and right of S, respectively. Let S^{--} be the square adjacent immediately to the left of S^-, and let S^{++} be the square adjacent to the right of S^+. Let $\mathcal{S}(p_i) = \{S^{--}, S^-, S, S^+, S^{++}\}$. Note that all disks from $D_{\bar{H}}$ that overlap S must be in $\textsc{um}(U)$ or $\textsc{lm}(U)$ for some square U in $\mathcal{S}(p_i)$. This follows as the disks have radius 2, and the squares have side length 1.4.

For every $T' \in \mathcal{P}(T) \backslash \{\emptyset\}$ we have the following indexes for the table W_{p_i}: For the outer T' envelope for each of the ten regions in the set of regions $\{\textsc{um}(U), \textsc{lm}(U) \ : \ U \in \mathcal{S}(p_i)\}$, we have a set of up to three disks that are candidates for the disk d that is on the outer T' envelope for that region at position $x = x_{p_i}$, for the disk just before d on that envelope and for the disk just after d on that envelope. For the inner T' envelope of each of the ten regions, we have one disk that is a candidate for being the disk on that envelope at position $x = x_{p_i}$. Hence, an entry of the table W_{p_i} is indexed by $40 \cdot (2^{|T|} - 1)$ disks (three disks for each of the ten outer envelopes, and one disk for each of the ten inner envelopes, for each choice of T'). For ease of presentation, we write the indexes for the table W_{p_i} as two sets of disks D_U and D_L, where D_U contains all the disks from the inner and outer T' envelopes for any T' and regions $\textsc{um}(U)$ for $U \in \mathcal{S}(p_i)$, and D_L contains all the disks from inner and outer T' envelopes for any T' and regions $\textsc{lm}(U)$ for $U \in \mathcal{S}(p_i)$.

Consider the case that the indexes for the table W_{p_i} for each T' are chosen as the disks that actually form the envelopes under consideration in an optimal solution to the horizontal strip problem, i.e., the indexes contain the corresponding disks on the envelopes $opt^{T'}_{\textsc{um}(S)}$, $opt^{T'}_{I(\textsc{um}(S))}$, $opt^{T'}_{\textsc{um}(S^+)}$ etc. We observe that if the classified coverage requirement of p_i is met by the optimal solution, then it is also met by the disks constituting the indexes for the table. To illustrate this, assume that p_i is covered in the optimal solution by two disks d_1^* and d_2^* from $\textsc{um}(S)$. If $T_{d_1^*} = T_{d_2^*}$, then for $T' = T_{d_1^*}$ we must have that p_i is covered once by the disk d_1 that forms the outer T' envelope for $\textsc{um}(S)$ at $x = x_{p_i}$, and a second time by the disk before or after d_1 on the same envelope, or by the disk on the inner T' envelope for $\textsc{um}(S)$ at $x = x_{p_i}$. In the other cases, the reasoning is similar.

Furthermore, we note that the disks $D_U \cup D_L$ specified as indexes for a table entry $W_{p_i}(D_U,\ D_L)$ completely separate the solution to the right of the line $x = x_{p_i}$ from the solution to the left of that line. In other words, if D' and D'' are different solutions for which the disks $D_U \cup D_L$ are the disks on the 20 envelopes relevant to p_i for all $T' \subseteq T$, then the left part of D' (disks of D' that appear before $D_U \cup D_L$ on their respective envelopes) can be combined with $D_U \cup D_L$ and the right part of D'' (disks of D'' that appear after $D_U \cup D_L$ on their respective envelopes) to form a new feasible solution. Furthermore, a disk d cannot simultaneously be in the left part of D' or D'' and in the right part of D' or D''. This follows as d appears either before or after a disk in $D_U \cup D_L$ on its respective envelope.

The value of an entry of table W_{p_i} is set to infinity if the disks indexing the table entry do not meet the classified coverage requirement for p_i, and otherwise the minimum cost of a set of disks that includes all disks indexing the table entry of W_{p_i} and that also meets the classified coverage requirements of all points preceding p_i in P_H. Once all tables W_{p_i} have been computed, the set of disks corresponding to the minimum value of any entry of table W_p for the last point $p = p_{n_H}$ is output as the solution.

4.5.4 Computing the Table Entries

Let the table entry for the leftmost point $p_1 \in P_H$ be initialised as follows. For every possible choice of disk indexes D_U and D_L, the corresponding table entry $W_{p_1}(D_U, D_L)$ is set to $w(D_U) + w(D_L)$ if $D_U \cup D_L$ meets the coverage requirement of point p_1, and ∞ otherwise. For subsequent points $p_i \in P_H \setminus \{p_1\}$, the value of a table entry $W_{p_i}(D_U, D_L)$ such that $D_U \cup D_L$ meets the coverage requirement of p_i is calculated as the minimum cost that can be obtained via the following method. Take the sum of the cost of the disks covering all points upto p_{i-1} from the table entry $W_{p_{i-1}}(D'_U, D'_L)$ for some D'_U and D'_L, and add the cost of all disks that are in $D_U \cup D_L$, but not in $D'_U \cup D'_L$. For every $p_i \in P_H \setminus \{p_1\}$, we calculate the table entry for each combination of disks on the envelope as follows:

$$W_{p_i}(D_U, D_L) = \begin{cases} \infty, \quad \text{if } D_U \cup D_L \text{ does not meet the coverage} \\ \qquad\quad \text{requirement for } p_i \\ \min_{D'_U, D'_L} \{W_{p_{i-1}}(D'_U, D'_L) + w(D_U - D'_U) \\ \qquad\qquad +w(D_L - D'_L)\}, \quad \text{otherwise} \end{cases} \tag{4.3}$$

Consider the last point $p_{n_H} \in P_H$. The minimum value in the table $W_{p_{n_H}}$ is the cost of the minimum weight solution that covers point p_{n_H} and all other points that preceded it in the ordered set p_H. If we keep as a record for each $W_{p_i}(D_U, D_L)$ the choice of disks D'_U and D'_L that attained the minimum value of equation (4.3), standard traceback techniques for dynamic programming can then be used to obtain a set of disks that forms a feasible solution and has cost equal to that table entry.

Lemma 4.19. *The dynamic programming computes an optimal solution to the horizontal strip problem.*

Proof. Assume that $W_{p_{n_H}} = v$ is the minimum value in table $W_{p_{n_H}}$, and that this value is not ∞. It follows that the set of disks output by the algorithm has weight v, as the weight of every disk added to the solution is accounted for in (4.3). Furthermore, the solution is feasible, as the corresponding entry in each table W_{p_i} is not ∞, and therefore the classified coverage requirement of each point p_i must be satisfied.

It remains to show that, if the optimal solution to the strip problem has weight $v*$, then the algorithm outputs a solution of weight at most $v*$. For a point $p_i \in P_H$, let $\mathcal{S}(p_i)$ be defined as in Section 4.5.3 and let $opt_U(p_i)$ be the disks corresponding to indexes of the table W_{p_i} (i.e., three disks from each outer envelope and one disks from each inner envelope) that are on the outer and inner T' envelopes for $\textsc{um}(U)$ for $U \in \mathcal{S}(p_i)$ for the fixed optimal solution. Let $opt_L(p_i)$ be defined analogously based on the disks on the outer and inner T' envelopes for $\textsc{lm}(U)$, for $U \in \mathcal{S}(p_i)$ for the optimal solution. Let $v^*(p_i)$ be the cost of all disks in the optimal solution that are not to the right of the disks in $opt_U(p_i) \cup opt_L(p_i)$ on their respective envelopes.

We claim that the following inequality holds for all $p_i \in P_H$.

$$W_{p_i}(opt_U(p_i),\ opt_L(p_i)) \leq v^*(p_i) \tag{4.4}$$

We prove our claim by induction on p_i. For p_1 we have that

$$W_{p_i}(opt_U(p_1),\ opt_L(p_1)) = w(opt_U(p_1) \cup opt_L(p_1)) = v^*(p_1) \tag{4.5}$$

This proves the claim for the base case of $i = 1$. For $i > 1$, following from (4.3) that

$$\begin{aligned} W_{p_i}(opt_U(p_i),\ opt_L(p_i)) \leq\ & W_{p_{i-1}}(opt_U(p_{i-1}),\ opt_L(p_{i-1})) \\ & + w(opt_U(p_i) \setminus opt_U(p_{i-1})) \\ & + w(opt_L(p_i) \setminus opt_L(p_{i-1})) \end{aligned}$$

By induction, we know that $W_{p_{i-1}}(opt_U(p_{i-1}),\ opt_L(p_{i-1})) \leq v^*(p_{i-1})$.

Furthermore, the weight $v^*(p_i) - v^*(p_{i-1})$ includes all disks that are in $opt_U(p_i) \cup opt_L(p_i)$ but not in the part of the optimal solution corresponding to $v^*(p_{i-1})$. One can be observe that this includes all of the disks contained in $opt_U(p_i) \setminus opt_U(p_{i-1})$ and in $opt_L(p_i) \setminus opt_L(p_{i-1})$. This follows as the disks in $opt_U(p_i) \setminus opt_U(p_{i-1})$, and the disks in $opt_L(p_i) \setminus opt_L(p_{i-1})$ appear on their respective envelopes to the right of the disks in $opt_U(p_{i-1}) \cup opt_L(p_{i-1})$ and thus are not contained in the part of the optimal solution corresponding to $v^*(p_{i-1})$. Hence it follows that

$$\begin{aligned} v^*(p_i) - v^*(p_{i-1}) \geq\ & w(opt_U(p_i) \setminus opt_U(p_{i-1})) \\ & + w(opt_L(p_i) \setminus opt_L(p_{i-1}) \end{aligned}$$

and the claim is established. It follows that

$$W_{p_{n_H}}(opt_U(p_{n_H}), opt_L(p_{n_H})) \leq v^*$$

and the algorithm outputs a feasible solution of cost at most v^*. We note that the cost must actually be equal to v^* as the algorithm outputs a feasible solution and v^* is the optimal cost. □

As we assume that the size of T is bounded by a constant, it follows that the number of disks required to index a table entry is bounded by a constant as well. The tables W_{p_i} are of polynomial size, and the computation of each table entry can be carried out in polynomial time. Thus, the overall running time is polynomial, and by Lemma 4.19 the algorithm outputs an optimal solution to the horizontal strip problem. This completes the proof of Lemma 4.9.

4.6 Lifetime Maximisation

In this section we describe a known general method [BCSZ04, BCSZ05, GK98a] for approximating the maximum lifetime problem by using an approximation algorithm for the minimum weight sensor cover problem, and its application to our setting. A linear program Π of the form

$$\{\max c^T x \mid Ax \leq b,\ x \geq 0\} \tag{4.6}$$

where A, b and c are non-negative, is known as a *packing problem.* The linear program may be given implicitly and the number of variables x_j may be exponential. For a given vector w, the problem of finding a column j of A such that $\sum_i A_{i,j} w_i / c_j$ is minimised is called the problem of *computing a column of minimum length* with respect to Π. It is known [BCSZ04] that, if a packing problem Π', admits a ρ-approximation algorithm for the problem of computing a column of minimum length with respect to Π' for any given vector w, then the algorithm by Garg-Könemann [GK98a] can be used to compute an $(1+\varepsilon)\rho$-approximate solution to Π'.

The natural linear programming formulation of the problem of maximising the lifetime of a sensor network is as follows:

$$\max \sum_{D' \in \mathcal{D}} x_{D'} \tag{4.7}$$

$$s.t. \sum_{\{D' \in \mathcal{D} \,:\, d \in D'\}} x_{D'} \leq b_d, \qquad \forall d \in D \tag{4.8}$$

$$x_{D'} \geq 0, \qquad \forall D' \in \mathcal{D} \tag{4.9}$$

Here, D is the set of sensor nodes constituting the network, and b_d is the initial battery level of a sensor node $d \in D$, specified in such a way that the battery level is sufficient for d to be active for b_d units of time. The set $\mathcal{D}$ contains all possible sensor covers, i.e., all subsets of D that satisfy the required sensor coverage constraint. The variable $x_{D'}$ represents the length of the part of the schedule during which the nodes of the sensor cover $D' \in \mathcal{D}$ are active. The objective (4.7) is to maximise the total length of the schedule. The constraints (4.8) specify that a node d can take part in sensor covers for a total amount of time that is bounded by b_d.

The linear program described above (4.7) - (4.9) does not have polynomial size, as the number of variables $x_{D'}$ can be exponential. However, it is a packing problem, and the algorithm by Garg and Könemann [GK98a] can be applied. The problem of computing a column of minimum length is the problem of computing a set $D' \in \mathcal{D}$ of minimum cost, where the cost of a node $d \in D$ is given by some weight w_d.

Theorem 4.20. *Let $\mathcal{D}$ be a set of valid sensor covers. If there exists a ρ-approximation algorithm for the problem of computing a set $D' \in \mathcal{D}$ of minimum cost, for any given node weights w_d, then for every fixed $\varepsilon > 0$,* *there exists a $\rho(1+\varepsilon)$-approximation algorithm for the lifetime maximisation problem.*

We remark that Theorem 4.20 applies to a wide variety of lifetime maximisation problems, because the specification of a set $\mathcal{D}$ of valid sensor covers can be arbitrary. For example, if a graph class admits a ρ-approximation algorithm for the weighted connected dominating set problem, then that graph class admits a $(\rho + \varepsilon)$-approximation algorithm for the lifetime maximisation problem, where at any point of time the active nodes must form a connected dominating set.

The combination of Theorem 4.11 and Theorem 4.20 implies the following corollary.

Corollary 4.21. *For every fixed $\varepsilon > 0$, there is a $(6 + \varepsilon)$-approximation algorithm for the problem ML2CUD-T.*

4.7 Connected Sensor Cover

In the previous sections of this chapter, we have discussed only the condition that the set of selected sensors meet the coverage requirement of each point in the set of points P. In many applications, such as settings described in [MYC09, VBL07], it is additionally required that the selected sensors form a connected network. For these settings, it is assumed that each sensor node constituting the network, is equipped with a wireless transmitter, that allows it to send messages to any other node in the network, that is located within a certain communication radius r_c from it. We remark that this corresponds to a communication graph, where the set of sensor nodes are represented by disks each of radius $r_c/2$, and two nodes are adjacent under the condition that their disks intersect. It is a natural expectation that the communication radius $r_c > r$, where r is the sensing radius of each sensor node constituting the sensor network. Under the assumption that $r_c \geq 2r$, i.e., the communication range of a sensor node is at least twice its associated sensing radius, we can extend the algorithms we have presented for *W2CUD-T* and *ML2CUD-T* in this chapter, to their corresponding problem variants with connectivity constraints.

For *W2CUD-T* with a connectivity requirement, we first compute a $(6+\varepsilon)$-approximate solution D' to the problem without the additional connectivity requirement, by using the algorithm from Theorem 4.11. Then, viewing the given disks as disks of radius $r_c/2$, we solve the minimum node-weighted Steiner tree problem for the disks in D' as terminals, using the algorithm with approximation ratio strictly bounded from above by 3.475 for node weighted Steiner trees in unit disk graphs [BGRS10, ES09, ZLGW09].

Let S be the set of Steiner nodes output by the algorithm. The set $D' \cup S$ is then output as a solution to *W2CUD-T* with the additional connectivity requirement. Let opt_c be an optimal solution to *W2CUD-T* with the additional connectivity requirement. Observe that opt_c is a (superset of a) feasible solution to the Steiner tree problem considered above: opt_c is connected and contains disks covering every point in P. Every disk in D' covers a point in P, and hence the centre of any disk in D' is within distance $r + r \leq r_c$ of the centre of some disk in opt_c. Consequently, $opt_c \cup D'$ is connected.

This shows that the Steiner tree approximation algorithm produces a set S of cost less than 3.475 times the cost of opt_c. As the cost of D' is within a factor of $(6 + \varepsilon)$ of the optimal solution to *W2CUD-T* without connectivity requirement, and thus is within the same factor of the cost of opt_c, the overall approximation ratio is bounded by 9.475, for a sufficiently small choice of ε.

Theorem 4.22. *For the variants of W2CUD-T and ML2CUD-T where the active disks in the solution need to be connected, and $r_c \geq 2r$, there exists a 9.475-approximation algorithm.*

4.8 Conclusion

In this chapter we have presented a $(6+\varepsilon)$-approximation algorithm for the target coverage problem with composite events, and observing fault tolerant requirements, both for the lifetime maximisation variant and for the problem of covering all event points by sensors of minimum total cost. Our approach is based on 'guessing properties' of the optimal solution via enumeration techniques, and then using these deduced properties to guide a dynamic programming algorithm.

This is a generalisation of the approach employed by Huang at al. [HGZW09] to obtain a $(6+\varepsilon)$-approximation for the weighted set cover with unit disks. For the latter problem subsequent work has improved the approximation ratio yielding a $(5+\varepsilon)$-approximation algorithm [DY09], and then a further improvement was obtained in the form of a $(4 + \varepsilon)$-approximation algorithm [EM09, ZWX+11]. The main observation resulting in these improvements is that the dynamic programming solution can be applied on several strips simultaneously. One possible direction for future work would be to see whether these improvements can also be adapted to the fault-tolerant target coverage problem with composite events.

Another question of interest is whether our approach can be adapted to arbitrary coverage requirements $k_p \geq 2$ for all $p \in P$. Repeated application of our $(6 + \varepsilon)$-approximation algorithm would incur an extra factor of $k/2$ in the approximation ratio, where $k = \max_p k_p$, which is not desirable. In particular, an algorithm that yields an approximation ratio independent of k would be of interest.

For *W2CUD-T* and also for the special case of weighted geometric set cover with unit disks, it is an interesting open question whether a polynomial-time approximation scheme (PTAS) can be obtained. As far as we can ascertain, no hardness-of-approximation results are known for these problems.

Chapter 5

Online Dominating Set

In this chapter we consider the online problem of constructing a strongly connected in-out dominating set, where an in-out dominating set is a subset of vertices such that every vertex not in the in-out dominating set has an edge to some vertex in the in-out dominating set, and an edge from some vertex in the in-out dominating set.

A *dominating set* in an undirected graph is a subset of vertices, such that every vertex is either in the dominating set or adjacent to some vertex in the dominating set. Finding such a dominating set of minimum size, amongst all possible dominating sets, is known as the minimum dominating set (*MDS*) problem. If one requires that the dominating set is connected, i.e., the subgraph induced by the dominating set is connected, the problem is known as the *minimum connected dominating set* (*MCDS*) problem.

In directed graphs it is not guaranteed that every pair of distinct vertices will be bidirectionally adjacent, i.e., there might not exist a directed edge from the first vertex of the pair to the second, and a directed edge from the second vertex in the pair to the first. Therefore, in directed graphs an analogous problem to the *MCDS* problem is to require that the dominating set be strongly connected, i.e., the subgraph induced by the dominating set is strongly connected. We consider an *in-out dominating set* where for every vertex not in the in-out dominating set, there exists a directed edge from a vertex in the in-out dominating set, and a directed edge to some vertex in the in-out dominating set. To find a minimum strongly connected in-out connected dominating set, amongst all possible strongly connected in-out dominating sets, is the *minimum strongly connected in-out dominating set* (*MSCIODS*) problem.

The study of dominating sets is of interest in practical situations, especially pertaining to routing transmissions in wireless ad-hoc networks. A common issue in wireless ad-hoc networks is to specify a *routing backbone* of the network, that is utilised to facilitate efficient routing of packets through the network. A routing backbone is formed of a subset of nodes in the network, where every node in the network is either part of the routing backbone, or has a neighbour (a node that can be sent a packet with a single-hop transmission) in the routing backbone. Nodes forming the routing backbone are tasked with acting as intermediate nodes in a multi-hop transmission forwarding packets to their destination. It therefore must be that every node in the routing backbone has a path to every other node in the routing backbone, or the network may be disconnected and some packets will have no route to their destination. Additionally, nodes must be able to receive transmissions from the routing backbone. It can be observed that the problem of specifying such a routing backbone of minimum size, is to solve the minimum connected in-out dominating set problem.

In practice, the need to solve this problem online would arise when an ad-hoc network is deployed node by node sequentially. The nodes already deployed would need to form a routing backbone and then as each additional node is deployed, the network must ensure that either the node is adjacent to the routing backbone, possibly by adapting the routing backbone to incorporate additional nodes. Therefore, we consider the online variant of the strongly connected in-out dominating set problem, where the set of vertices is presented one by one and the online algorithm is tasked with maintaining a strongly connected in-out dominating set, with the aim of minimising the size of the in-out dominating set.

The decision problem concerning the existence of a dominating set of cardinality less than ρ in a graph is known to be $\mathcal{NP}$-complete, and indeed it is one of the classical problems shown to be $\mathcal{NP}$-complete by Garey and Johnson [GJ79]. The connected dominating set problem is also $\mathcal{NP}$-complete [GJ79]. As every instance in undirected graphs can be transformed into an instance in directed graphs (for every undirected edge (v, v') between two vertices in the undirected graph, one creates two directed edges in the directed graph $(v, v'), (v', v)$), it follows that *MSCIODS* is also $\mathcal{NP}$-complete.

5.1 Related Work

The problem of constructing a minimum connected dominating set (*MCDS*) in an offline setting has been well studied. A greedy $(\ln \Delta + 3)$-approximation algorithm was presented by Guha and Khuller [GK98b] which was then improved to a $(\ln \Delta + 2)$-approximation by Ruan et al. [RDJ+04], where Δ denotes the maximum degree of any vertex in the graph. Guha and Khuller [GK98b] further showed that there does not exist a polynomial time algorithm with approximation ratio $c \cdot H(\Delta)$ unless $\mathcal{P} \neq \mathcal{NP}$, where H is the harmonic function and $0 < c < 1$.

MSCIODS has been studied before in the offline setting. Thai et al. presented several approximation algorithms for *MSCIODS* [DTL+06, TWL+07, TTD08]. These results were then improved by Li et al. [LDW+09] who presented the first logarithmic approximation, achieving approximation ratio $(3H(n-1)-1)$, where H is the harmonic function. The authors also note that to achieve an approximation ratio strictly less than $2\ln n + \mathcal{O}(1)$ one cannot hope to improve their approach, and new ideas would be needed.

More pertinent work to the online setting also exists. The online dominating set problem was first studied by King and Tzeng [KT97], who have shown that no online algorithm can have competitive ratio less than $n-1$ for the online minimum dominating set (*Online-MDS*) problem. They also present a simple online algorithm that achieves competitive ratio $n-1$.

More relevant to our work is that of Eidenbenz [Eid02] who considers the online variant of the *MCDS* problem (*Online-MCDS*), which in combination with the work of Li et al. [LDW+09] on *MSCIODS* provided the inspiration for our setting. Eidenbenz [Eid02] studies various different graphs and settings. He considers tree graphs, unit disk graphs and graphs of bounded degree (i.e., every vertex has degree at most ρ). Aside from providing algorithms for these problems, complementary lower bound constructions show that the general lower bound of $n-1$ on the competitive ratio for general graphs also holds for bipartite graphs, planar graphs, disk graphs and trees of bounded width. Eidenbenz also shows that no online algorithm can have competitive ratio less than $n-2$ for *Online-MCDS*.

We also remark that Eidenbenz [Eid02] studies a variant that allows the online algorithm to handle vertices being removed form the graph in some round, as determined by the adversary. We also initially considered a variant where the online algorithm can delete vertices from its in-out dominating set for a cost, however for our setting this model is not applicable. It can be shown that an algorithm for the online version of *MSCIODS* (*Online-MSCIODS*) cannot gain anything by removing vertices from its in-out dominating set. This follows as any vertex that can be covered by a strongly connected in-out dominating set can be accepted into the in-out dominating set without needing to revoke any vertices from its in-out dominating set (Lemma 5.3).

To the best of our knowledge, we are the first to consider the problem of specifying a strongly connected in-out dominating set in an online setting.

5.1.1 Contributions

In this work we present a lower bound construction showing that no deterministic online algorithm can achieve competitive ratio $n/4 + 1/4$ for the online variant (*Online-MSCIODS*) of the *MSCIODS* problem. We state and show several observations regarding the problem, and then proceed to complement the lower bound result by showing an online algorithm that achieves competitive ratio $n/4 + 1/4$, which is the best possible competitive ratio of any online algorithm for the problem.

The remainder of the chapter is structured as follows: In Section 5.2 we present a formal definition of the problem, introduce notation, and provide some preliminary observations. In Section 5.3 we present a lower bound construction which holds for all online deterministic algorithms. We then proceed in Section 5.4 to complement this lower bound with the analysis of an algorithm which attains competitive ratio matching the aforementioned lower bound. We then conclude this work in Section 5.5 by summarising the results presented and stating open problems which could be of interest.

5.2 Preliminaries

We first formally define the problem. Consider the following to be the *Minimum Strongly Connected In-Out Dominating Set* (*MSCIODS*) problem. Given as input is a directed graph $G = (V, E)$ containing a set V of $n = |V|$ vertices, and a set of edges E, where an individual edge $(u, v) \in E$ is a directed edge from u to v, where $u, v \in V$. We say that u is *adjacent to* v and v is *adjacent from* u. We also say that two vertices u, v are *bidirectionally adjacent* if there exist edges $(u, v), (v, u) \in E$.

An *in-out dominating set* $D \subseteq V$ satisfies the property that for every $v \in V \setminus D$ there exist edges $(v, d), (d', v) \in E$, where $d, d' \in D$. We do not require that $d \neq d'$. For an in-out dominating set D and a vertex v (or a set of vertices V'), we say that D *covers* v (or V'), or alternately but equivalently v is *covered by* D. Additionally, we say that a vertex $d \in D$ *covers* a vertex $v \in V$ (or equivalently v is *covered* by d) if d and v are bidirectionally adjacent. For technical reasons, every vertex $v \in V$ covers itself if it is a member of the in-out dominating set D.

A solution consisting of a subset of vertices $D \subseteq V$ for *MSCIODS*, is said to be *feasible* if two constraints are satisfied. Firstly that D is an in-out dominating set of V. Secondly, it is required that the in-out dominating set D is strongly connected. The objective function is to minimise $|D|$ amongst all feasible in-out dominating sets.

In this work we consider the online variant of *MSCIODS*, which we refer to as *Online-MSCIODS*. The set of vertices V now forms an ordered set $(v_1, \ldots, v_n)$, and the vertices are *presented* one by one to the online algorithm. After presentation of a vertex $v_i \in V$, the online algorithm must decide whether or not to *accept* vertices from the subset $\{v_1, \ldots, v_i\}$ into its in-out dominating set D. We note that although the algorithm can at any stage accept vertices previously presented, it cannot remove vertices from its in-out dominating set once they have been accepted into it. It must be that the in-out dominating set covers all vertices $\{v_j \ : \ 1 \leq j \leq i\}$. The *final cost* to the algorithm is the cardinality of its in-out dominating set $|D|$ after presentation of the last vertex v_n. The solution quality with comparison to an optimal algorithm (which has knowledge of the entire set of vertices with which to make its decision) is obtained via competitive analysis, namely the ratio of the final cost of the online algorithm to the cost of the optimal offline algorithm. We constrain the optimal offline algorithm such that it must specify an in-out dominating set D, such that for all i where $1 \leq i \leq n$, $D \cap \{v_1, \ldots, v_i\}$ induces a strongly connected subgraph that covers the vertices $v_1, \ldots, v_i$. One can show that without this constraint, any online algorithm will have competitive ratio $n-2$. This follows from a construction presented by Eidenbenz [Eid02] which we sketch here for the reader:

Consider the following construction: The graph consists of a set of vertices $\{v_1, \ldots, v_n\}$, and a set of edges $\{(v_i, v_{i+1}), (v_{i+1}, v_i) : 1 \leq i < n\} \cup \{(v_i, v_n), (v_n, v_i) : 1 \leq i < n\}$. The online algorithm must accept v_1 otherwise v_1 is not covered. As each vertex v_i is presented, where $1 < i < n$, to specify a feasible in-out dominating set the online algorithm must accept v_{i-1} to cover v_i. The optimal offline algorithm will accept only v_n which is strongly connected and covers all vertices. The online algorithm will have final cost $n - 2$ and the optimal offline algorithm will have cost 1.

For the remainder of the chapter we refer to the online algorithm as *alg* and the optimal offline algorithm as *opt*.

5.2.1 Observations

We initiate the study of this problem by first presenting some observations concerning the properties of any arbitrary algorithm for *Online-MSCIODS*, which we will utilise later in the analysis of our algorithm. At this stage we will also note an assumption that we adhere to throughout this work without loss of generality. We will assume that a feasible solution always exists for all of the instances presented to the algorithm, i.e., we do not consider inputs for which it is impossible for any algorithm to specify an in-out dominating set. If such an instance were presented, an algorithm could detect that a solution does not exist, and subsequently output such a finding. As an optimal algorithm would also not be able to present a solution for such an instance, the analysis of such a situation is superfluous to our work.

Lemma 5.1. *Any algorithm for Online-MSCIODS must accept the first presented vertex v_1.*

Proof. For the convenience of the reader we recapitulate one of the properties of an in-out dominating set. It must be that every vertex $v_i \in V$ is either in the in-out dominating set, or has an edge to some vertex in the in-out dominating set, and an edge from some vertex in the in-out dominating set. As the in-out dominating set is initially (before presentation of v_1) empty, it is not possible that v_1 has an edge to or from any vertex in the in-out dominating set. Therefore, the only feasible solution is to accept v_1 into the in-out dominating set in order for v_1 to cover itself. □

We observe that as *opt* must accept the first presented vertex, if *opt* accepts no further vertices it must be that this first presented vertex covers all other presented vertices, and therefore any online algorithm also need only accept this first presented vertex, and can attain competitive ratio 1. This yields the following Corollary:

Corollary 5.2. *If opt has cost one, an online algorithm for Online-MSCIODS need accept only the first presented vertex, and can achieve competitive ratio 1.*

We can also deduce the following useful property for an algorithm solving *Online-MSCIODS.*

Lemma 5.3. *Any vertex v_i that is covered by a strongly connected in-out dominating set $D \subseteq V$, can be accepted into D with cost one.*

Proof. If $v_i \in D$ the claim follows immediately. If $v_i \notin D$ then we show that if v_i is accepted into the in-out dominating set D, D will remain strongly connected. For D to feasibly cover v_i it must be that there exist edges (d, v_i), (v_i, d') for some $d, d' \in D$. Recall that possibly $d = d'$.

Firstly, consider the edge $(d,\ v_i)$. As D is strongly connected, there exists a path from every vertex $d'' \in D \setminus \{d\}$ to d, and as there also exists the edge (d, v_i) it follows that there exists a path from every $d'' \in D \setminus \{d\}$ to v_i.

We have an analogous situation when we consider the edge $(v_i,\ d')$. As D is strongly connected there exists a path from d' to every other vertex in the in-out dominating set $d'' \in D \setminus \{d'\}$, and due to the existence of the edge $(v_i,\ d')$, there exists a path from v_i to every $d'' \in D \setminus \{d'\}$.

As v_i has a path to and from every vertex $d \in D$ in the in-out dominating set, it follows that v_i can be accepted into the in-out dominating set without the acceptance of any other vertices (accruing cost one), and the in-out dominating set will remain strongly connected. $\square$

All vertices must be covered by an in-out dominating set when they are presented, and from Lemma 5.3, an algorithm can accept any vertex covered by a strongly connected in-out dominating set. We therefore obtain the following Corollary:

Corollary 5.4. *An algorithm for Online-MSCIODS can accept all n vertices in V, and provide a solution which is strongly connected, and covers all vertices in V.*

5.3 Lower Bound

In this section we will present a lower bound on the competitive ratio of any deterministic online algorithm for *Online-MSCIODS*.

Lemma 5.5. *For Online-MSCIODS no deterministic online algorithm can achieve competitive ratio less than* $n/4 + 1/4$ *if* n *is odd, or less than* $n/4$ *if* n *is even, where* n *is the number of vertices in the graph.*

Proof. We present two different constructions. Note that the figures presented in this section (Figures 5.1 and 5.2) are for illustrative purposes and for sake of presentation do not include all edges. Firstly, we consider the case where the number of vertices presented is odd.

Case - n is Odd: Consider the following construction: The first vertex v_1 is presented and following from Lemma 5.1 both the deterministic online algorithm *alg*, and the optimal offline algorithm *opt* must accept this. An additional set of vertices are presented namely $\{v_2, \ldots, v_j\}$, where $j = n/2 + 1/2$, and all vertices in this set are bidirectionally adjacent to v_1. Note that all vertices $v_i \in \{v_2, \ldots, v_j\}$ will be covered by v_1, and as such the in-out dominating set $\{v_1\}$ will suffice as a feasible solution.

The algorithm is now presented with a further vertex $\bar{v_2}$ which is bidirectionally adjacent to all vertices $\{v_2, \ldots, v_j\}$. As $\bar{v_2}$ is not covered by v_1, *alg* must now accept some vertex $v_i \in \{v_2, \ldots, v_j\}$ into its in-out dominating set to cover it. Assume without loss of generality that *alg* accepts v_2 to cover $\bar{v_2}$. A further vertex $\bar{v_3}$ is presented which is bidirectionally adjacent to all vertices $v_i \in \{v_3, \ldots, v_j\}$. As $\bar{v_3}$ is not covered by *alg*'s in-out domi-

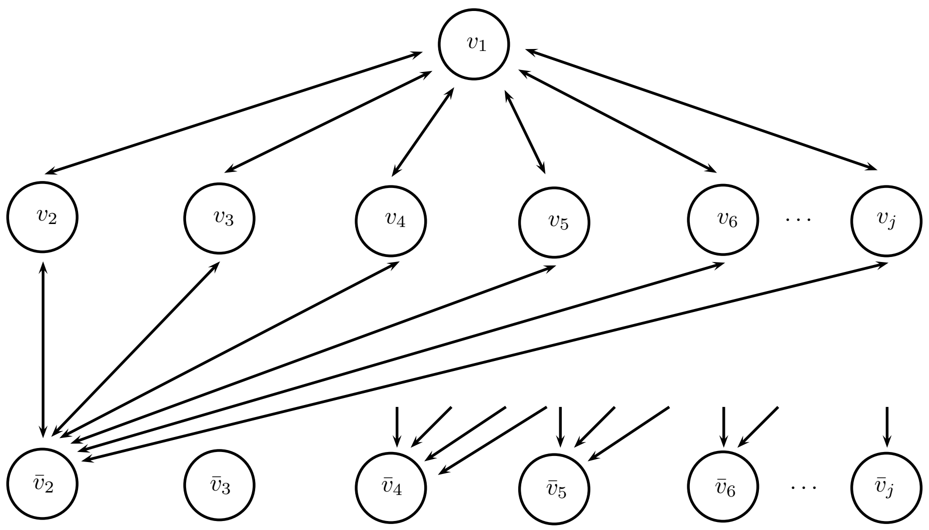

Figure 5.1: n is odd lower bound construction.

nating set, the algorithm must choose a vertex $v_i \in \{v_3, \ldots, v_j\}$ to cover $\bar{v_3}$. Without loss of generality assume this to be v_3.

We continue this trend for all $\bar{v_i} \in \{\bar{v_4}, \ldots, \bar{v_j}\}$ where $\bar{v_i}$ is bidirectionally adjacent to all vertices in $\{v_i, \ldots, v_j\}$, and we assume without loss of generality that *alg* accepts v_i to add to its in-out dominating set, in order to cover $\bar{v_i}$.

After presentation of $\bar{v_j}$, *alg* will have the in-out dominating set $\{v_1, \ldots v_j\}$ and subsequently will have cost $n/2 - 1/2 + 1 = n/2 + 1/2$. The optimal offline algorithm will have the in-out dominating set $\{v_1, v_j\}$ which covers all vertices, and will accrue cost 2. Consequently in the case that the total number of vertices presented is odd, an arbitrary deterministic online algorithm cannot have competitive ratio less than $n/4 + 1/4$.

Case - n is Even: We now consider the alternate case where the total number of vertices presented is even. As with the case where n is odd, the first vertex v_1 is presented and following from Lemma 5.1 both the deterministic online algorithm *alg*, and the optimal offline algorithm *opt* must accept this into their respective in-out dominating sets. An additional set of vertices is presented, namely $\{v_2, \ldots, v_j, v_l\}$, where $j = n/2$, and all vertices in this set are bidirectionally adjacent to v_1.

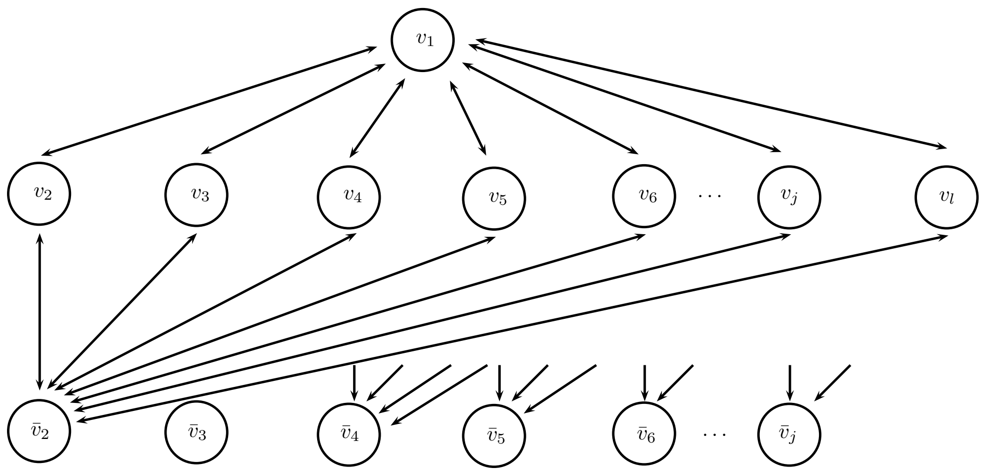

Figure 5.2: n is even lower bound construction.

Similar to the construction for the case of odd n, the remaining vertices $\{\bar{v_2}, \ldots, \bar{v_j}\}$ are presented one by one. The vertex $\bar{v_2}$ is adjacent to all vertices $v_i \in \{v_2, \ldots, v_j, v_l\}$ and we assume without loss of generality that the online algorithm chooses v_2 to accept into its in-out dominating set in order to cover $\bar{v_2}$. The optimal offline algorithm will choose either v_j or v_l to accept into its in-out dominating set, and the correctness of this solution for the optimal algorithm will become apparent in the following discussion. For each remaining vertex $\bar{v_i} \in \{\bar{v_3}, \ldots, \bar{v_j}\}$, $\bar{v_i}$ is bidirectionally adjacent to all vertices $\{v_i, \ldots, v_j, v_l\}$ and we assume without loss of generality that upon presentation of $\bar{v_i}$, the online algorithm accepts v_i into its in-out dominating set to cover $\bar{v_i}$. After the final vertex is presented the online algorithm will have the in-out dominating set $\{v_1, v_2, \ldots, v_j\}$ with cost $n/2$ and the optimal offline algorithm will have in-out dominating set $\{v_1, v_j\}$, or equivalently the in-out dominating set $\{v_1, v_l\}$ with cost 2. As such for this construction an arbitrary deterministic online algorithm will have competitive ratio $n/4$.

After the analysis of both constructions, we can conclude that for the *Online-MSCIODS* problem, no deterministic online algorithm can achieve competitive ratio less than $n/4 + 1/4$ if n is odd, or $n/4$ if n is even.

$\square$

5.4 Online Algorithm

Now that we have concluded that no deterministic online algorithm can achieve competitive ratio less than $n/4 + 1/4$, the aim is to show an algorithm that attains competitive ratio as close to $n/4 + 1/4$ as possible. We state explicitly for the reader that as a consequence of Lemma 5.5, one need only consider the analysis of an algorithm for *Online-MSCIODS* with regard to the cases where the optimal offline algorithm accrues cost two or three. If *opt* has cost greater than, or equal to four then as a consequence following from Corollary 5.4 the online algorithm can accept all n vertices and achieve competitive ratio $n/4$. Alternately, if the optimal offline algorithm has cost one then as a consequence of Corollary 5.2, an online algorithm need also accept only the first presented vertex and attain the same cost as an optimal offline algorithm. We therefore present our algorithm under such assumptions.

Before presenting observations upon which our algorithm is based, we will discuss a partitioning of the set of vertices V into subsets. With the exception of v_1, all vertices, i.e., all $v_i \in V \setminus \{v_1\}$, upon their presentation will be *classified* into at most one of the following three subsets, under the assumption that the optimal solution has cost less than four. S_1 is the set of all vertices $v_i \in V \setminus \{v_1\}$ that when presented, can be covered by an in-out dominating set $\{v_1\}$. S_2 is the set of all vertices $v_i \in V \setminus (\{v_1\} \cup S_1)$ that upon their presentation can be covered by the set $\{v_1\} \cup S_1$ of vertices that have been previously presented. Finally S_3 is the set of all vertices $v_i \in V \setminus (\{v_1\} \cup S_1 \cup S_2)$, that when presented can be covered by the set of vertices $\{v_1\} \cup S_1 \cup S_2$. Note that with the exception of v_1, every vertex upon its presentation is either classified into S_1, S_2 or S_3. Therefore, $V = \{v_1\} \cup S_1 \cup S_2 \cup S_3$.

We remark explicitly for the reader that vertices are classified upon their presentation, and do not get reclassified after presentation of further vertices.

With regard to the optimal solution, if *opt* has final cost two, we refer to the second vertex that the optimal offline algorithm accepts as o_2. Similarly, in the case where *opt* has final cost three, we refer to the second and third vertex that the optimal offline algorithm accepts, as o_2 and o_3 respectively.

Lemma 5.6. $o_2 \in S_1$

Proof. *opt* has accepted the first presented vertex v_1 following from Lemma 5.1. As o_2 is accepted by *opt*, the only way that the subgraph induced by

$\{v_1, o_2\}$ can be strongly connected is if there exists an edge from o_2 to v_1, and an edge from v_1 to o_2. Consequently, from the definition of S_1 the claim follows. □

From Lemma 5.6 we know that $o_2 \in S_1$. If *opt* has final cost two, then as the optimal offline algorithm can cover all vertices with a strongly connected in-out dominating set formed as a subset of $S_1 \cup \{v_1\}$, we can obtain the following Corollary of Lemma 5.6:

Corollary 5.7. *If opt has final cost two, an algorithm need only accept vertices from* $S_1 \cup \{v_1\}$.

We obtain similar properties if the optimal algorithm has final cost three.

Lemma 5.8. *If opt has final cost three, either* $o_3 \in S_1$, *or* $o_3 \in S_2$.

Proof. Upon presentation of o_3, and its consequent acceptance by *opt*, in order to maintain a strongly connected in-out dominating set the subgraph induced by the set $\{v_1, o_2, o_3\}$ must conform to one of the three following configurations: either o_3 is bidirectionally adjacent to v_1, and consequently $o_3 \in S_1$, o_3 is bidirectionally adjacent to o_2 (recall from Lemma 5.6, $o_2 \in S_1$), and consequently $o_3 \in S_2$, or alternately o_3 has an edge to (respectively from) v_1 (resp. o_2) and an edge from (resp. to) o_2 (resp. v_1), and consequently $o_3 \in S_2$. As these three aforementioned cases are the only possible cases, the claim follows. □

Analogous to the case where *opt* has final cost two, we know from Lemma 5.8 that if o_3 is not in S_1, then it must be the case that $o_3 \in S_2$. As such we can deduce that the optimal solution is contained within the set of vertices $\{v_1\} \cup S_1 \cup S_2$ and consequently any algorithm can form a feasible solution with that set, yielding the following Corollary:

Corollary 5.9. *If opt has final cost three, an algorithm for Online-MSCIODS need only accept vertices from $S_1 \cup S_2 \cup \{v_1\}$.*

For the remainder of this work, let v_k denote the first vertex that cannot be covered feasibly by an optimal in-out dominating set with cardinality less than three. Such a vertex can be determined by an online algorithm as follows. Upon presentation of a vertex v_i, the algorithm will determine if there exists some $v_j \in \{v_1, \ldots, v_i\} \cap S_1$, such that $\{v_1, v_j\}$ is a strongly connected in-out dominating set for all subgraphs induced by the sets of vertices $\{v_1, \ldots, v_l\}$, where $v_1 \leq v_l \leq v_i$. If there exists $\{v_1, v_j\}$ that covers all vertices already presented then $v_k > v_i$, otherwise $v_k = v_i$. Once v_k has been determined the value of k is fixed and for the remaining vertices yet to be presented, the aforementioned test for v_k is not performed. Note that such pairs are the only feasible coverings of a vertex with an in-out dominating set of two vertices, as from Lemma 5.1, v_1 must be accepted and the second vertex in the in-out dominating set must be in S_1, or such a solution would not be strongly connected. If such a pair does not exist then it must be that v_k cannot be covered with less than three vertices. The reader should note that if the optimal solution consists of two vertices then v_k will not exist.

Lemma 5.10. *The third vertex the optimal algorithm accepts (o_3) is presented before v_k.*

Proof. As v_k is defined to be the first vertex that can only be covered by an in-out dominating set with cardinality greater than two, there exists an optimal solution $\{v_1, o_2\}$ that covers v_k with cardinality two. Assume the negation of the claim, specifically that the vertex v_k is presented before the third vertex in the optimal solution o_3. As v_k is defined to be the first vertex which can only be covered with an in-out dominating set of at least size three, and the optimal solution has only cardinality two, a feasible solution would not exist in this scenario.

It remains to show that it cannot be the case that o_3 is v_k. As v_k is defined to be the first vertex that cannot be covered with an in-out dominating set of size less than three, it must be that $\{v_1, o_2\}$ cannot cover o_3. If this was the case then the optimal algorithm would not be able to accept o_3 and provide a strongly connected in-out dominating set, which would make the optimal algorithm infeasible and the claim follows. $\square$

From this result we can derive a key observation that we use in the analysis, namely that the algorithm need only accept vertices presented before v_k, and this will provide a feasible solution.

Lemma 5.11. *Any algorithm for Online-MSCIODS needs only accept vertices from the subset $\{v_i \ : \ i < k\} \subseteq V$ to provide a feasible solution.*

Proof. The claim follows almost directly from Lemma 5.10. As the optimal solution contains at most three vertices, namely $\{v_1, o_2, o_3\}$, and o_3 is presented before v_k by Lemma 5.10, consequently it must be that a feasible solution exists in the subset of vertices $\{v_i \ : \ i < k\} \subseteq V$. $\square$

For ease of presentation we now introduce the following notation, let $X = \{v_i \ : \ v_i \in S_1, \ i < k\}$, $Y = \{v_i \ : \ i \in S_2, \ i < k\}$ and finally let $Z = \{v_i \ : \ v_i \in S_1 \cup S_2 \cup S_3, \ i \geq k\}$. Namely, X is the set of vertices in S_1, presented before v_k, Y is the set of vertices in S_2, presented before v_k, and S_3 is the set of vertices in S_1, S_2 and S_3, that are presented after v_k, together with v_k. We note that the algorithm will know whether a presented vertex v_i should be classified into X, Y or Z, as it can deduce a vertex to be v_k, and if v_k has not been previously presented, and v_i is not v_k, then it knows that k is presented after the current vertex. Additionally, note that (X, Y, Z) is a partition of $V \setminus \{v_1\}$.

Recall that for the following algorithm, we will assume that the optimal solution has cost either two or three. This follows as from Corollary 5.2 if the optimal solution has cost one, the algorithm will be optimal, and if the optimal solution has cost greater than three, the algorithm can accept all n vertices and achieve competitive ratio $n/4$. Consider Algorithm 2 to be the definition of the online algorithm for *Online-MSCIODS*, we also refer to the algorithm as *alg*.

We now show that the algorithm *alg* specified is correct, i.e., upon presentation of a vertex v_i, all vertices $v_j \in (v_1, \ldots, v_i)$ will be covered by the algorithm's in-out dominating set, and the in-out dominating set will be strongly connected.

Lemma 5.12. *The online algorithm alg specifies a correct solution.*

Proof. We first consider the case where the algorithm is presented with an instance where the optimal algorithm has a solution of final cost at most

Algorithm 2: Online Algorithm (*alg*) for *Online-MSCIODS*.

```
Solution set $D = \emptyset$
foreach presented vertex $v_i \in V$ do
    if $v_i$ is covered by $D$ then
        Do not accept any vertices
    else if $v_i = v_1$ then
        Accept $v_1$ into the in-out dominating set $D$
    else
        Classify $v_i$ into $X$, $Y$ or $Z$
        if $v_i \in X$ then
            Do not accept any vertices
        else if $v_i \in Y$ then
            if $D \cup \{v_j\}$ covers $v_i$, for some $v_j \in X \setminus D$ then
                Accept $v_j$ into the in-out dominating set $D$
            else if $D \cup \{v_j, v_l\}$ covers $v_i$, for some $v_j, v_l \in X \setminus D$ then
                Accept $v_j, v_l$ into the in-out dominating set $D$
        else if $v_i \in Z$ then
            if $D \cup \{v_j\}$ covers $v_i$, for some $v_j \in X \setminus D$ then
                Accept $v_j$ into the in-out dominating set $D$
            else if $D \cup \{v_j\}$ covers $v_i$, for some $v_j \in Y \setminus D$ then
                Accept $v_j$ into the in-out dominating set $D$
            else if $D \cup \{v_j, v_l\}$ covers $v_i$, for some $v_j, v_l \in X \setminus D$ then
                Accept $v_j, v_l$ into the in-out dominating set $D$
            else if $D \cup \{v_j, v_l\}$ covers $v_i$, for some $v_j \in X \setminus D$,
            $v_l \in Y \setminus D$ then
                Accept $v_j, v_l$ into the in-out dominating set $D$
            else
                Accept all vertices in $V$
```

three. By the definition of the set S_1, it must be that all vertices in S_1 are covered by v_1, and as such will be covered by the current in-out dominating set, which must contain v_1 as it must be that the algorithm accepts the first vertex following from Lemma 5.1.

From the definition of the set Y, and Lemma 5.11, all vertices in Y can be covered by $\{v_1\} \cup X$, it therefore follows that the algorithm will always be correct to accept only vertices from X to cover it. Additionally, for each vertex to be covered there needs to be at least one vertex in the in-out dominating set with an edge to that vertex, and at least one vertex in the in-out dominating set with an edge from that vertex. Therefore, it suffices to accept at most two vertices from X to cover a presented vertex $v_i \in Y$.

For vertices presented in Z we can deduce the following. In the case that *opt* has final cost three, we note from Lemma 5.11 that it is always feasible to accept vertices $v_j \in \{v_j : j < k\}$. Therefore, any vertex in S_1, S_2 or S_3 can be covered by accepting at most two vertices from $X \cup Y$. In the case where the presented vertex $v_i \in Z \cap S_1$, v_i will be covered by $\{v_1\}$ and the algorithm will accept nothing. In the case that the presented vertex $v_i \in Z \cap S_2$, then it will suffice to accept either at most two vertices from X to cover it, or accept at most one vertex from X and accept at most one vertex from Y. This follows as the optimal solution either has (in addition to v_1) two vertices (o_2, o_3) in X or one vertex (o_2) in X, and one vertex (o_3) in Y. In the case that $v_i \in Z \cap S_3$ it will suffice to accept at most one vertex from X, and at most one vertex from Y to cover it.

Note that as each vertex is accepted into a strongly connected in-out dominating set, the in-out dominating set will remain strongly connected following from Lemma 5.3.

In the case that the optimal algorithm has final cost greater than three, the algorithm will be presented with some vertices that cannot be covered by an in-out dominating set with size at most three. In this case, if such a vertex v_i is presented that cannot be covered by vertices from $X \cup Y \cup \{v_1\}$, then the algorithm will accept all vertices $v_j \in V$. From Corollary 5.4 this will produce a correct solution. $\square$

Lemma 5.13. *If opt has final cost two, alg has competitive ratio at most* $n/4 + 1/4$.

Proof. We first recall from Lemma 5.6 that *opt* has the solution set $\{v_1, o_2\}$, and from Corollary 5.7 it follows that *alg* needs only accept vertices from $\{v_1\} \cup S_1$. Recall that v_1 will cover all vertices $v_i \in S_1$ by definition of S_1. If $|S_2| \leq n/2 - 1/2$ then for each single presented vertex $\bar{v_j} \in S_2$, *alg* needs accept at most one vertex $v_j \in S_1$ to cover $\bar{v_j}$. In the case $|S_2| > n/2 - 1/2$ then *alg* could in the worst case accept all vertices in S_1, and have cost bounded from above by $(n-1)/2$. As it is evident that in both cases *alg* accepts at most $n/2 - 1/2$ vertices from S_1, and additionally accepts v_1, *alg* has a final cost of at most $n/2 + 1/2$. The optimal algorithm has final cost two, and hence *alg* has competitive ratio $n/4 + 1/4$ for *Online-MSCIODS* in the case where the optimal algorithm has a final cost of two. $\square$

Lemma 5.14. *If opt has final cost three, alg has competitive ratio at most* $n/4 + 1/12$.

Proof. We consider two cases to show the claim. Firstly, we assume that

$k \leq 3n/4$. Recall from the statement of the algorithm that at no point does the algorithm accept any vertices $\{v_i \ : \ v_i \geq v_k\}$. We also know as a consequence of Lemma 5.11 that the optimal solution is contained entirely in the set of vertices $\{v_i \ : \ i < k\}$. Therefore, the entire set of vertices that the algorithm will consider accepting is contained in those vertices presented before v_k, and as such the number of vertices accepted by the algorithm, and consequently the final cost of the algorithm is bounded above by $3n/4$. As *opt* has final cost three, the competitive ratio of the online algorithm will be $3n/12 = n/4$.

We now proceed to consider the alternate and more interesting case that $k > 3n/4$. For ease of presentation allow us to recapitulate notation. Let $X = \{v_i \ : \ v_i \in S_1, \ i < k\}$, $Y = \{v_i \ : \ i \in S_2, \ i < k\}$ and $Z = \{v_i \ : \ v_i \in S_1 \cup S_2 \cup S_3, \ i \geq k\}$. Let $x = |X|$ denote the cardinality of the set of vertices in S_1, that are presented before v_k, let $y = |Y|$ denote the cardinality of the set of vertices in S_2, that are presented before v_k and let $z = |Z|$ denote the cardinality of the set of vertices in S_1, S_2 or S_3, presented after v_k.

From Lemma 5.8 either $o_3 \in X$ or $o_3 \in Y$. Consider the case $o_3 \in X$. As both optimal vertices are in X, it must be that any vertex presented can be covered by $\{v_1\} \cup X$ or the optimal algorithm would be incorrect. As such, any vertex presented in $Y \cup Z$ can be covered by accepting at most two vertices in X. Consider the case $y + z \leq n/3 - 1/3$. As for every vertex presented in $Y \cup Z$, the online algorithm will accept at most two vertices from X, the algorithm will accept at most $2n/3 - 2/3$ vertices. Similarly, in the alternate case of $y + z > n/3 - 1/3$ the algorithm will accept less than $2n/3 - 2/3$ vertices from X. Thus, as the algorithm accepts v_1 and at most $2n/3 - 2/3$ vertices from X, in total the algorithm will have final cost bounded by $2n/3 + 1/3$. As *opt* has final cost three, in this case the competitive ratio of the online algorithm would be $2n/9 + 1/9$.

We now consider the alternate case where $o_3 \in Y$. We derive two different bounds on the total number of the vertices the algorithm will accept. Let *acc* denote the number of vertices the algorithm accepts in total, i.e., the final cost of the algorithm. We first can deduce that:

$$acc \leq y + 2z + 1 \tag{5.1}$$

This bound (inequality (5.1)) follows mostly from the correctness of the algorithm (Lemma 5.12). The algorithm will accept v_1 with cost one. For vertices presented in X no vertex need be accepted at it will already be covered by v_1. We can observe that as $o_3 \in Y$, the optimal algorithm has at most one vertex in X. Therefore, there exists at least one vertex in X that in conjunction with v_1, covers all vertices in Y. Therefore, for every vertex presented in Y, the algorithm will accept at most one vertex from X to cover the presented vertex. We also can observe that for every vertex presented in Z, the algorithm will accept at most two vertices into its in-out dominating set to cover it (either one vertex from X or Y, two vertices from X, or one vertex from X and one vertex from Y). In total the algorithm will accept at most $y + 2z + 1$ vertices.

In a similar fashion we now present a second bound on *acc*:

$$acc \leq x + z + 1 \tag{5.2}$$

Analogous to the first bound (inequality (5.1)) on the total cost of the algorithm, this inequality (inequality (5.2)) follows mostly from the proof of feasibility of the specified algorithm (Lemma 5.12). We can observe that in any case, the algorithm could accept all x vertices in X in order to cover all vertices $v_i \in Y$, following from the feasibility of the algorithm which states that any vertex $v_i \in Y$ can be covered by the set of vertices $\{v_1\} \cup X$. Additionally, if all vertices in X have been accepted then it follows from Lemma 5.12 that to cover any vertex presented in $v_i \in Z$, it will suffice to accept at most one vertex in Y to cover it, i.e., for all vertices v_i presented in Z, alg will need to accept at most z vertices from Y to cover v_i.

Utilising the two aforementioned bounds on the total number of vertices accepted by the algorithm (acc), we are now in a position to determine the competitive ratio of the algorithm. Recall that $k > 3n/4$, and consequently $z < n/4$. If we assume $y+z > n/2-1/2$, and consequently that $x < n/2-1/2$ we derive the following:

$$\begin{aligned} acc &\leq x + z + 1 \\ &\leq n/2 - 1/2 + n/4 - 1/4 + 1 \\ &= 3n/4 + 1/4 \end{aligned}$$

In this case, as the optimal algorithm has total cost three, the competitive ratio of the online algorithm will be $3n/12 + 1/12 = n/4 + 1/12$.

In the remaining case, namely that $y + z \leq n/2 - 1/2$, we deduce from the inequality (5.1) the following:

$$\begin{aligned} acc &\leq y + 2z + 1 \\ &= (y + z) + z + 1 \\ &\leq n/2 - 1/2 + n/4 - 1/4 + 1 \\ &= 3n/4 + 1/4 \end{aligned}$$

Similar to the previous case where $y + z > n/2 + 1/2$, as the optimal algorithm has final cost three the competitive ratio of the online algorithm will be $n/4 + 1/12$.

As in all cases, the online algorithm has competitive ratio bounded by $n/4 + 1/12$ the claim follows. $\square$

Following from the previous discussion if the optimal algorithm has final cost one then the online algorithm is optimal, and if the optimal algorithm has cost at least four then the algorithm can accept all vertices and achieve the best possible competitive ratio, within a small additive constant. Therefore, by Lemmas 5.13 and 5.14 we get the following theorem:

Theorem 5.15. *The deterministic online algorithm (alg) has competitive ratio $n/4 + 1/4$ for Online-MSCIODS.*

5.5 Conclusion

In this chapter we have presented and studied the problem of determining a strongly connected in-out dominating set of minimum size, in an online setting. We have presented a lower bound of the problem, showing that no deterministic algorithm can achieve competitive ratio less that $n/4$ in the case where n is even, and no less than $n/4 + 1/4$ in the case where n is odd. We have then complemented this result with an algorithm that achieves competitive ratio $n/4 + 1/4$.

An interesting open problem would be to study a slight variation of *Online-MSCIODS* that we have presented in this chapter. One could consider the online problem of constructing a strongly connected dominating set of minimum size where every vertex not in the dominating set has an edge into the dominating set.

In the offline setting it would also be interesting to try to present an approximation algorithm achieving approximation ratio less than $2 \cdot \ln n$ for the problem of constructing a minimum strongly connected in-out dominating set. This would improve the $2 \cdot \ln n$ approximation algorithm presented by Li et al. [LDW$^+$09]. Another direction for research would be to consider the offline problem of constructing a minimum strongly connected dominating set where every vertex not in the dominating set has an edge into the dominating set.

Chapter 6

Barrier Resilience

In this chapter we consider the problem of determining the barrier resilience of a sensor network consisting of sensors, whose sensor regions are convex shapes of equal width, with respect to paths that do not decrease in the x-coordinate (monotone paths).

In the realm of sensor networks, *barrier coverage* is a fundamental concept. Typically, one will have a setting where there is a starting point and a target point and one wishes to place sensors such that all possible paths from the starting point to the target point can be monitored. In sensor networks there will be a set of sensors between the two points, and a *sensor region* for each sensor that represents the area that the sensor covers. The objective of barrier coverage is to guarantee that when one considers all possible paths between the two points, at least one sensor intersection occurs, i.e., any path from the start point to the target point, will intersect at least one sensor region.

One can quickly realise that barrier coverage has many practical applications. For one example, consider the setting of a room with one door and on the opposite side of the room lies a valuable piece of artwork. One potential security precaution would be to lay a series of sensors between the door and the artwork, such that a potential thief would be detected entering the room and trying to obtain the artwork. Barrier coverage in this setting would dictate that any path from the door to the artwork must incur the thief being detected by at least one sensor.

With a sensor network that provides barrier coverage, no possible undetected path exists between the start point and the target point. However, one can observe that in the case where a single sensor fails, a path could exist such that it is undetected by any sensor in the network. In critical settings, such a scenario would be unacceptable. As such, the notion of robustness in a sensor network becomes of interest. The question changes from the binary notion of 'is a sensor network secure?', i.e., does the network provide barrier coverage, to 'how secure is the sensor network?'. One can attain a greater level of robustness by incorporating redundancy into the sensor network. Specifically, one could require that all possible paths between the start and target points, intersect at least k sensors. One can note that the choice of k would vary on the level of robustness required, offset against an acceptable level of redundancy in the sensor network.

The interested reader can refer to the Ph.D thesis of Kumar [Kum06] for a more general survey of the area of barriers in sensor network. Mindful of the aforementioned notion of robustness in sensor networks, for the remainder of this work we will concern ourselves specifically with two notions of barrier coverage that measure robustness.

We adopt the definitions of *barrier thickness* and *barrier resilience* from Bereg and Kirkpatrick [BK09]. Firstly, one is concerned with the number of sensor regions that are intersected, by any path from the start point to the destination point. More specifically, one is interested in the *barrier thickness* of the sensor network, defined to be the minimum number of such sensor region intersections, over all possible paths from the start region to the destination region. One is also interested in the number of sensors that must be removed such that a path exists between the start point and the destination point, that does not intersect any sensor regions. Specifically *barrier resilience* is defined to be the minimum number of sensors that must be removed, in order that such a path exists between the start region and the destination region, such that no sensors are intersected. The main difference between the two concepts is that the barrier resilience only counts the first time a path intersects a sensor, in contrast to the barrier thickness that counts each time a path enters a sensor region. Note that the notions of barrier thickness and barrier resilience are related, namely that the thickness of the barrier is at least the resilience of the barrier. In particular, if an optimal path that induces the minimum number of sensor region intersections enters each sensor at most once, then the resilience of the barrier is equal to the thickness.

We will consider these concepts of barrier thickness and barrier resilience in a setting where one considers only monotone paths through the network. The study of montone paths is motivated by settings where a path through a sensor network can only travel in one direction along an axis, such as a scenario involving a vertical descent.

6.1 Related Work

There are many different notions of sensor coverage. One can consider the survey papers presented by Meguerdichian et al. [MKPS01] and Cardei and Wu [CW04], or alternatively refer to the Ph.D thesis of Kumar [Kum06] for an overview of this vast area. We focus on settings where a sensor network provides barrier coverage where one must ensure that all paths between the start point and the target point intersect at least one sensor.

Several approaches of ensuring that a sensor network providing barrier coverage is robust have been proposed. Meguerdichian et al. [MKPS01] consider maximal breach paths, where one considers paths that maximise the distance to the closest sensor. A similar approach was proposed by Meguerdichian et al. [MKQP01] who consider the minimum exposure to sensors in the network. They also complement their result with experimental results.

Kumar et al. [KLA05, KLA07] continue this line of research and introduce the measure of *k-barrier coverage* in belt regions. An *open belt region* is defined to be a closed rectangular subset (a strip) of the plane, where a path travels from a point on the lower boundary of the strip, to a point on the upper boundary of the strip. If every path through the sensor network intersects at least k distinct sensors, then the sensor network attains k-barrier coverage. They show that for open belt regions, and sensor networks formed of unit disks, the problem of determining if a sensor network attains k-barrier coverage can be reduced to the problem of determining whether or not a graph has k-node disjoint paths between two vertices.

Bereg and Kirkpatrick [BK09] are the first to extend the theoretical analysis of previous works on barrier resilience from open belts, e.g. [KLA07], to arbitrary (two-dimensional) regions. In particular, Bereg and Kirkpatrick [BK09] provide more specific inspiration for our work. They introduce the concepts of barrier thickness and barrier resilience. We consider variants of barrier thickness and resilience in our work, specifically thickness and resilience with respect to monotone paths. Similar to the work of Bereg and Kirkpatrick [BK09], we also consider the problem of calculating the barrier resilience of a network, however, Bereg and Kirkpatrick consider approximation algorithms for a case where the set of sensor regions are unit disks. In contrast we consider exact polynomial time algorithms, when one considers only monotone paths from the start point to the target point, for sensor regions that are convex shapes of uniform width.

Bereg and Kirkpatrick [BK09] show that one can approximate the resilience of a barrier by relating the thickness of the barrier to the resilience of the barrier formed by a sensor network. Specifically, they show that when one considers sensors whose coverage regions are closed unit disks, an optimal path will enter each sensor at most three times. It follows from this result that the thickness of the barrier is at most three times the resilience of the barrier, when considering unit disks. From this, they outline a method for calculating the thickness of the barrier in polynomial time, and consequently obtain a 3-approximation algorithm for calculating the barrier resilience of a sensor network formed of unit disks. Additionally, when one requires the restriction that the start point and the target point are separated to a certain degree in the plane, they show a 2-approximation algorithm. Furthermore, they sketch an additional result that tightens the approximation ratio from 2 to 5/3.

Tseng and Kirkpatrick [TK11] consider the hardness of calculating the barrier resilience of sensor networks. They show that calculating the barrier resilience of a sensor network formed of unit line segments when considering general paths is $\mathcal{NP}$-hard. They then further extend this result and show that the problem remains $\mathcal{NP}$-hard for sensor networks formed of sensors whose coverage regions form non-symmetrical sensor regions in the plane.

6.2 Contributions

In a similar fashion to Bereg and Kirkpatrick [BK09] we will show that via the use a of dual graph representing the arrangement of the sensor regions forming the network, one can calculate the barrier resilience of a sensor network. We will introduce the concept of *monotone barrier resilience*, that considers the barrier resilience of a sensor network with respect to any monotone path between two points. In contrast to [BK09] who study sensor regions of unit disks, we consider more general sensor regions, namely sensor regions of equal width that are convex shapes, and show that if one restricts the set of potential paths to paths that are monotone, there exists a polynomial time algorithm that can solve the problem of determining the monotone barrier resilience of such a sensor network.

6.3 Preliminaries

Throughout this work we consider the two-dimensional Euclidean plane. For a point p in the plane, let x_p denote the x-coordinate of p and similarly let y_p denote the y-coordinate of p.

A path π is defined as a continuous function $f_\pi : X \to \mathbb{R}^2$, where $X = [0, 1]$. Each path has an initial point $f(0) = x$, and a terminal point $f(1) = y$. The initial and terminal points are known as the *endpoints* of a path. When discussing paths, we equivalently refer to the path from x to y. The *interior* π' of a path π, is defined as $f : X \to \mathbb{R}^2$, where X is the interval $(0, 1)$. Let a *subpath* π' of π be $f : X \to \mathbb{R}^2$ where X is the interval $[a, b]$ and $0 \leq a < b \leq 1$. A path π is *monotone* if for every $j > i$, where $f(i) = p$ and $f(j) = \bar{p}$, $x_p \leq x_{\bar{p}}$.

Consider a set of *sensor regions* D where every sensor region $d \in D$ is a bounded open convex set of the plane, where a set is convex if for any pair of points in the set, all points of the straight line between the pair of points are in the set. A *boundary point* of a set is a point that appears in the closure of the set but not in the interior of the set [Eng89]. Let the set of boundary points of a sensor region form the *boundary* of a sensor region. A set of sensor regions D in the plane is a set of *pseudo-disks* if, for every pair of sensor regions (d, d'), where $d, d' \in D$, there exist at most two points that appear on the boundaries of both d and d'. We are motivated to study pseudo-disks as they are a generalisation of unit disks, that have been studied before for the barrier resilience problem.

Two sensor regions $d, d' \in D$ *intersect* if there exists some point p that appears on the boundary of both d and d'. Similarly, a path π intersects a sensor region $d \in D$ if there exists some point p that appears in d and is a point of π.

Let l_d denote the *left-most* point constituting the boundary of a sensor region d, in terms of x-coordinate. Note that l_d may not be uniquely defined in which case l_d refers to an arbitrary left most point, unless we specify otherwise. Similarly, let r_d denote the right-most point constituting the boundary of a sensor region d. The *width* of a sensor region d is the x-distance between the left-most and right-most points, i.e., $x_{r_d} - x_{l_d}$.

Lemma 6.1. *Each sensor region has a left-most point.*

Proof. Firstly, we state the following three well known properties of topological spaces:

- For some $S \subseteq \mathbb{R}^n$, S is a compact set iff S is bounded and closed. [Sut75]
- The continuous image of a compact set is compact. [Sut75]
- The continuous image of a connected set is connected. [Sut75]

The interval $[0, 1]$ is bounded and closed, and consequently is compact. Consider a function $f_\pi : [0, 1] \to \mathbb{R}^2$ representing a path in the Euclidean plane. Let the image of f be denoted by P, i.e., let P be the set of points on the path π. As $[0, 1]$ is compact, the image of f is also compact.

Now consider a function $g : \mathbb{R}^2 \to \mathbb{R}$ which maps points in the plane to their x-coordinates. Let $g(P)$ be denoted by P', i.e., for every point $p \in P$,

$x_p \in P'$ is the x-coordinate of p. As P is compact, $g(P)$ is also compact, and thus P' is bounded and closed. Furthermore, as the single interval $[0, 1]$ is connected, it must be from the definition of the function f that P is also connected and thus, as the continuous image of a connected set is also connected, P' is also connected. As P' is connected, bounded, and closed it follows that the P' contains an x-coordinate of minimal value amongst all x-coordinates in P', and this represents the left-most point of the path π.

Consider a path π' to be the closed path that traces the boundary of a sensor region d. It will follow that the left-most point of d will exist as the left-most point of π' exists. $\square$

Note that following from Lemma 6.1 the right-most point of a sensor region d can be determined by symmetrical arguments.

In this work it is useful to refer to certain parts of the boundary of a sensor region. We define a *boundary segment* of some sensor region $d \in D$ to be the following: Consider a closed path π_b that traces the boundary of a sensor region. Let a boundary segment between two points p and p' on the boundary of d, be the set of points that appear on the subpath between p, p' of the path π_b. Throughout this work we will refer numerous times to a specific classification of the boundary into two particular boundary segments, namely the *upper-boundary segment* and the *lower-boundary segment*, as the *extremal boundary segments.* Consider a sensor region d, and specifically its left most and right most points, l_d and r_d respectively.

The upper-boundary segment of a sensor region is defined as the set of points formed by the union of the left-most and right-most points, together with the points on the boundary segment contained clockwise from the left-most to the right-most points of the sensor region. Similarly, the lower-boundary segment of a sensor region is defined as the set of points formed by the union of the left-most and right-most points, together with the points on the boundary segment contained anti-clockwise from the left-most to the right-most points of the sensor region. We illustrate the two extremal boundary segments of a square in Figure 6.1.

Figure 6.1: The upper-boundary segment (left), and lower-boundary segment (right) of a square.

Observe that the left-most points (and right-most points) of d will appear on both the upper-boundary segment and the lower-boundary segment. We say that a point p on the boundary of a sensor region is *strictly on* the upper-boundary segment if p is disjoint from the lower-boundary segment. The definition of a point being strictly on the lower-boundary segment is analogous.

Let a point p be *above a sensor region* d if p has greater y-coordinate than some point p' strictly on the upper-boundary segment of d, such that p' has the same x-coordinate as p, i.e, $x_p = x_{p'}$. Analogously, we say a *point*

p is below a sensor region d if p has a smaller y-coordinate than some point p' strictly on the lower-boundary segment of d, such that $x_p = x_{p'}$.

6.3.1 Problem Definitions

A natural modelling of a sensor network forming a barrier is to consider each sensor and the region within which it can detect events, to be represented by a corresponding sensor region in the plane. Similarly, one considers all possible paths an intruder could take as the set of paths that can exist in the plane from a start point s to a target point t. A *sensor region intersection* is a point p, that appears on both the boundary of a sensor region d, and appears in the path π. Note that a path entering a sensor region at a point of the boundary and leaving the sensor region at another point of intersection will incur two intersection points. A path π *intersects* a sensor region if there exists at least one sensor region intersection. Throughout this work, we will assume that the start point s and the target point t are disjoint from all sensor regions.

In the literature [BK09] two related problems are studied that we take interest in. Firstly the *thickness of the barrier* which is defined as follows:

Definition 6.2. (Barrier Thickness) The minimum number of sensor region intersections over all possible paths from the start point s, to the target point t.

We mainly consider the following problem that although related to barrier thickness, is more pertinent to our setting.

Definition 6.3. (Barrier Resilience) The minimum number of sensors regions that once removed allows the existence of a path from the start point s, to the target point t, without intersecting any sensor region.

We consider throughout our work, only monotone paths. As such, we define analogous concepts to those described above, that apply to the setting where the set of paths considered, is restricted to monotone paths.

Definition 6.4. (Monotone Barrier Thickness) The minimum number of sensor region intersections over all possible monotone paths from the start point s, to the target point t.

Definition 6.5. (Monotone Barrier Resilience) The minimum number of sensor regions that once removed allows the existence of a monotone path from the start point s, to the target point t, without intersecting any sensor region.

An instance of the *monotone barrier resilience* problem contains as input a set of sensor regions D, and two points s, t where s is the start point and t is the target point, placed in the two-dimensional Euclidean plane. The aim of the problem is to determine the monotone barrier resilience of the sensor network. The main result of this work is to show the following theorem:

Theorem 6.6. *There exists an algorithm that, given an instance of the monotone barrier resilience problem for an arrangement of sensor regions that are of equal width and form a set of pseudo disks, will determine the monotone barrier resilience of the sensor network in polynomial time.*

6.4 Monotone Barrier Resilience Algorithm

In order to show the existence of the algorithm in Theorem 6.6, we will follow the broad approach of Bereg and Kirkpatrick [BK09] and take inspiration from their 3-approximation algorithm for a similar problem, where they consider paths of arbitrary direction and a set of sensor regions restricted to unit disks.

One can construct a dual graph representing the sensor network in the following manner. Consider an arrangement of sensor regions D in the Euclidean plane. For each point of intersection between the boundaries of the sensor regions introduce a vertical line in the arrangement at the x-coordinate of each intersection point. Similarly, for every sensor region $d \in D$ introduce a vertical line in the arrangement at the x-coordinate of the left-most point of d and a vertical line at the x-coordinate of the right-most point of d. The vertical lines are required to segment faces of the arrangement such that in the resulting dual graph one can adhere to the constraint that paths are monotone.

For each face of the arrangement of the sensor network, one creates a corresponding vertex in the dual-graph. Additionally, for every pair of adjacent faces there is a weighted edge in the dual graph between the vertices, corresponding to the pair of adjacent faces in the graph. For the following set of edges, consider only edges that correspond to a transition from a face f to a face f' where the faces f and f' have either equal x-range, or the x-range of the face f' is greater than the x-range of the face f. We illustrate an example of a basic arrangement with the corresponding dual graph overlaid in Figure 6.2. For the sake of presentation we only include a small subset of the edges in the dual graph.

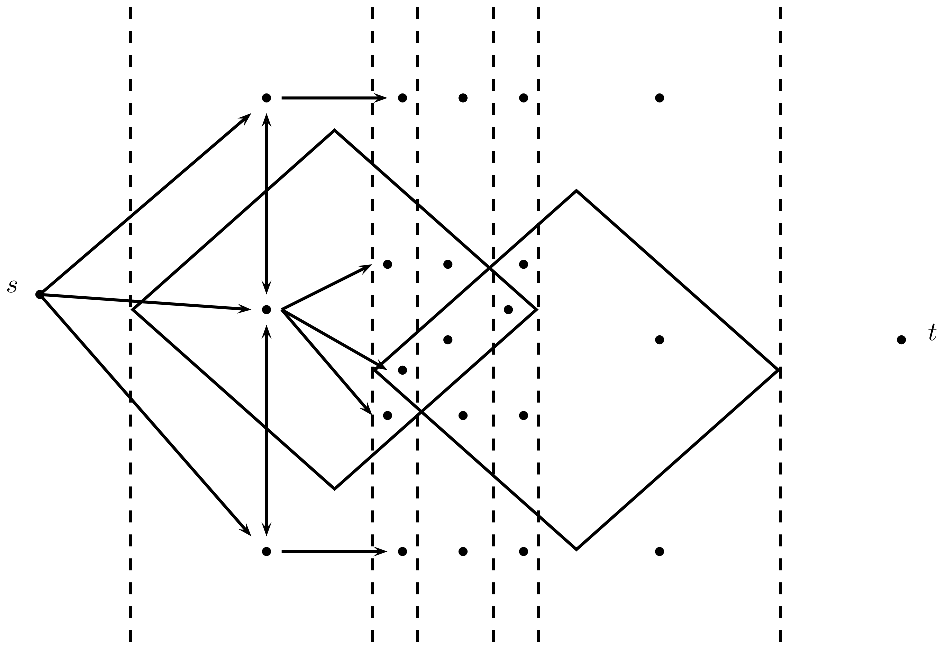

Figure 6.2: An example of a basic arrangement with s and t and the corresponding dual graph overlaid.

Edges that correspond to entering a sensor region are assigned weight one, and edges that leave a sensor region are assigned weight zero. We add additional vertices for the faces containing the start point s and the terminal point t. All edges to the face containing t from adjacent faces have weight zero, and all edges from the face containing s to adjacent faces have either weight one if they enter a sensor region, or weight zero otherwise. A path from s to t of weight k represents k sensor regions being entered by that path. A path of minimum weight through the arrangement corresponds to the minimum number of sensor regions that any monotone path through the network must intersect.

One can find a shortest path from s to t in this dual graph in polynomial time, using standard shortest path algorithms. Specifically, this will indicate the monotone barrier thickness of the sensor network.

Theorem 6.7. *There exists an algorithm that, given an instance of the monotone barrier thickness problem for an arrangement of sensor regions that are of equal width and form a set of pseudo disks, will determine the monotone barrier thickness of the sensor network in polynomial time.*

Recall our earlier remark, that should it be the case that an optimal path (that induces the minimum number of possible sensor region intersections) intersects each sensor at most once, then it follows that the monotone barrier resilience is equal to the monotone barrier thickness, and as such one would be able to calculate the monotone barrier resilience of a sensor network in polynomial time using the above method. Note that the complexity of constructing this dual graph is the complexity of constructing the arrangement, which can be computed in $\mathcal{O}(n^2)$ [AGR00], where n is the number of curve segments in the arrangement, plus the complexity of computing the shortest path, both of which are polynomial in the input size of the problem. We will adopt this approach in our work.

For the remainder of this section we will consider a fixed optimal solution for an instance of the monotone barrier resilience problem, and let $D^* \subseteq D$ denote the set of sensor regions that are not removed in the set of objects that corresponds to an optimal solution. We refer to such sensor regions D^* that are not removed by an optimal solution as *obstacles*.

We will show the following theorem regarding the maximum number of times a monotone path intersects a sensor region:

Theorem 6.8. *When considering an arrangement of sensor regions D of uniform width that are pseudo disks, a shortest monotone path π that does not intersect any obstacles in D^*, will intersect the boundary of any sensor region $d \in D \setminus D^*$ at most twice.*

Once this claim is established, we can observe that a shortest monotone path between s and t that avoids the obstacles D^*, will enter each sensor region in the network at most once, and as such the monotone thickness of the network will be equal to the monotone resilience of the barrier. From the discussion in the previous section Theorem 6.6 will follow. To show the result, we first require some preliminary observations before continuing and discussing the aforementioned Theorem 6.8.

Lemma 6.9. *Consider two points p and p' on the boundary of a sensor region d. If p and p' appear on distinct extremal boundary segments of d then there does not exist a monotone path from p to p', whose interior does not intersect d.*

Proof. We first note that as p and p' appear on distinct extremal boundary segments, it must be that neither p nor p' are a left-most point, or a right-most point of the sensor region d. This follows from the observation that the left-most point and the right-most point of a sensor region appear on both extremal boundary segments (i.e., the upper-boundary segment and the lower-boundary segment). Therefore, without loss of generality, assume that p appears strictly on the upper-boundary segment, and that p' appears strictly on the lower-boundary segment.

Assume the negation of the statement of the lemma. Specifically, assume that there does exist some monotone path π between p and p', where no point constituting the interior of π intersects d. We consider the two such paths that could exist between p and p', as illustrated in Figure 6.3.

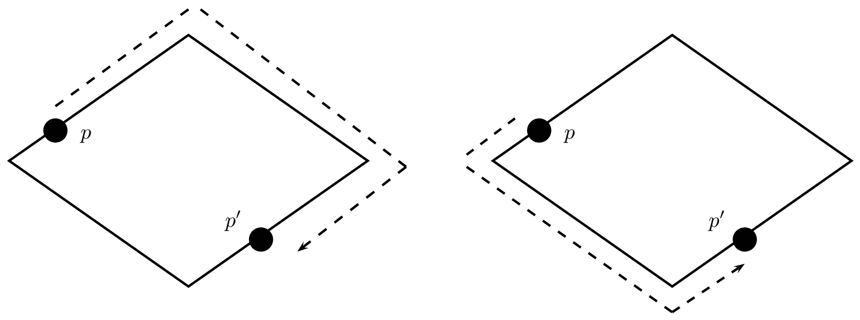

Figure 6.3: Two possible paths between p and p' that do not intersect the sensor region d.

The path π could have a monotone subpath over the upper-boundary segment of d originating at p. When π reaches the right most point of d, it must be that π has a subpath decreasing in the x-axis from some point to the right of the right most point of d to p'. Such a subpath would violate the monotonicity constraint.

In the alternate case, we know that p is not a left most point of d and therefore if some subpath of π is under the lower-boundary segment to reach p', it must be the case that some point forming the path π is to the left of the left-most point of d and consequently to the left of p. Such a path π would violate the constraint that π is monotone. As in both cases, there cannot exist a monotone path from p to p' that does not intersect d, the claim follows. $\square$

As we know two points must be on the same extremal boundary segment of a sensor region d for there to exist a monotone path between them (that does not intersect d), we proceed to make observations regarding instances where this extremal boundary segment of d is intersected by another sensor region d'.

Lemma 6.10. *Consider the boundary segment of a sensor region d contained between the intersection points induced by the intersection of another sensor region d'. That boundary segment is contained entirely within the sensor region d', i.e., there exists no point on that boundary segment of d, that is outside d'.*

Proof. Denote the two intersection points as i_1 and i_2 (assume i_1 appears clockwise on the boundary from i_2) and trace the boundary segment from i_1 to i_2. If any part of the boundary segment was outside of d', it would follow that along this trace, there exists some intersection point i_3 where the boundary segment of d would intersect the boundary of d'. From the definition of the set of sensor regions being a set of pseudo-disks, a third intersection point cannot exist and the claim follows. $\square$

Following from Lemma 6.10 we can obtain the following Corollary:

Corollary 6.11. *Consider a sensor region d with two points p and p' on the lower-boundary (upper-boundary) segment. Assume that a sensor region d' intersects the lower-boundary (upper-boundary) segment of d, with intersection points i_1, i_2. If either p or p' appear on the boundary segment contained*

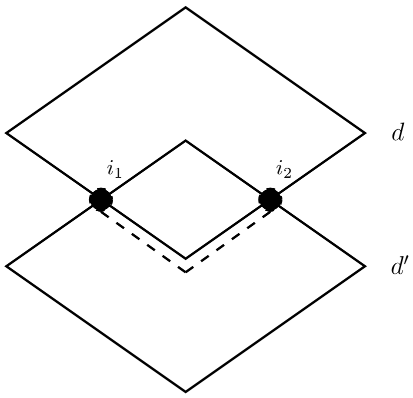

Figure 6.4: An illustration of Lemma 6.10.

anti-clockwise (clockwise) between i_1 and i_2 then they appear inside the sensor region d' and there cannot exist a monotone path from p to p', whose interior does not intersect d or d'.

In the following we will show results for a restricted input, namely a constraint on the arrangement of sensor regions D given as input. We will require that all sensor regions have width r, i.e., for all $d \in D$, $x_{r_d} - x_{l_d} = r$ and also focus on only shortest monotone paths that do not intersect any obstacles in D^*. We now proceed to restate and show the proof of Theorem 6.8.

Theorem 6.8. *When considering an arrangement of sensor regions D of uniform width that are pseudo disks, a shortest monotone path π that does not intersect any obstacles in D^*, will intersect the boundary of any sensor region $d \in D \setminus D^*$ at most twice.*

Proof. Consider any sensor region $d \in D \setminus D^*$. We define p_1 as the intersection point where π first intersects the boundary of d, let p_2 be the point where π intersects the boundary of d for a second time and let p_3 denote the third point where the path π intersects the boundary of d. Intuitively, p_1 represents the path entering the sensor region d, p_2 represents the path leaving the sensor region d and p_3 would represent the path entering the sensor region d for a second time. We will show such a point p_3 cannot exist, and consequently a shortest monotone path can only enter a sensor region once.

It follows from Lemma 6.9 that p_2 and p_3 must both appear on the lower-boundary segment (or both appear on the upper-boundary segment), or it will be the case that a monotone path that does not intersect d cannot exist from p_2 to p_3. We will assume without loss of generality throughout this proof that p_2 and p_3 both appear on the lower-boundary segment. In the case that p_2 and p_3 both lie on the upper-boundary segment analogous arguments apply.

Consider a fourth point p_4 on the boundary of d where the path π intersects the boundary of the sensor region d a fourth time. Intuitively, this would represent the path π leaving the sensor region d a second time. Note that as we consider only the shortest monotone path that does not intersect any obstacles in D^*, it must be that there does not exist a shorter monotone path from p_2 (the point where π leaves d for the first time) to p_3 (the point where π enters d a second time), otherwise it would contradict the fact that π is a shortest monotone path (see Figure 6.5 for an illustration). We note for the reader that all figures in this proof of this theorem are for illustrative purposes. We assume that additional obstacles in D^* exist that prevent a path inside the sensor region d from p_1 to p_4 that doesn't intersect any sensor region, but such obstacles are excluded from the figures for sake of presentation.

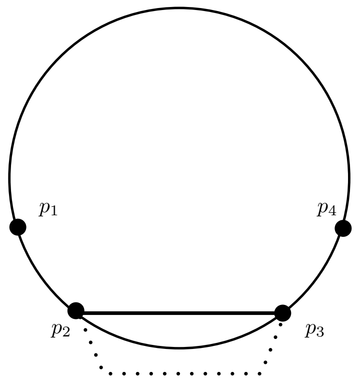

Figure 6.5: One path (dotted) from p_2 to p_3 outside of d, and a shorter path from p_2 to p_3 inside d.

Furthermore, we can deduce that there must be some sensor region $d' \in D^*$ intersecting the lower boundary of d between p_2 and p_3. Assume the negation that there does not exist such a sensor region d'. Consider a monotone path from a point on the boundary of d just to the left of p_2 to a point on the boundary just to the right of p_3 contained entirely inside the sensor region d, that traces the boundary segment contained anti-clockwise from p_2 to p_3 (such a trace is arbitrarily close to the boundary but not intersecting it). Note that all points on the path would be inside d. This path from p_2 to p_3 inside d would be shorter than a path that leaves the sensor region at p_2, and re-enters the sensor region at p_3, and consequently would contradict the definition of π being the shortest monotone path.

It must also be the case that the two intersection points i_1, i_2 between d and d' are on the boundary segment contained anticlockwise from p_2 to p_3. If only one of the two intersection points is on the boundary segment contained anticlockwise between p_2 and p_3, then either p_2 or p_3 must be on the boundary segment contained anticlockwise between i_1 and i_2 (assuming without loss of generality that i_2 appears anticlockwise on the boundary of d from i_1), following from Corollary 6.11.

We now proceed by considering two different cases. We first consider the case (illustrated in Figure 6.6) where the left-most point of the sensor region d' has x-coordinate greater than the left-most point of d, i.e., $x_{l_{d'}} > x_{l_d}$.

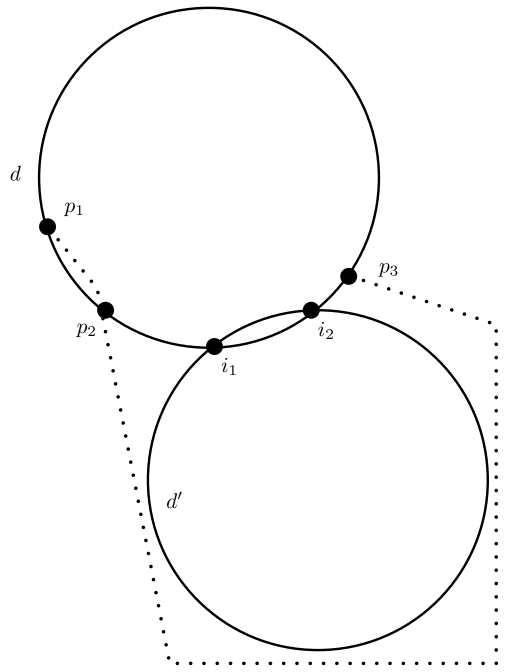

Figure 6.6: A path from p_1 to p_3 around d'.

It must be the case that the right most point of d' is disjoint from d, as the left-most point of d' is to the right of the left-most point of d, and all sensor regions have uniform width. It must also be that p_3 is above d'. Suppose otherwise, specifically that p_3 is under d'. There would exist some segment of the lower-boundary of d' that is above p_3. It follows from Lemma 6.10 that the boundary segment of d' contained clockwise between i_1 to i_2 must be entirely inside d or it would induce more than two intersection points, violating that the set of sensor regions D is a set of pseudo-disks. This segment of d' cannot contain the right-most point of d, as we know the right-most point of d' is disjoint from d.

We note that the monotone path between p_2 and p_3 cannot solely be over d' as such a path would intersect d to the left of i_1. As p_3 is defined as the third point where the path π intersects the boundary of d , and we know i_1 appears clockwise from p_3 on the boundary, this cannot be the case. We therefore know that the path π must contain some subpath under d' (as π does not intersect the obstacles $d' \in D^*$). The subpath under d' must have a point p', to the right of the right-most point of d'. There must also exist another subpath from this point p' to the right of the right-most point, to p_3. As it is the case that the right-most point of d', must be to the right of the right-most point of d, and as all sensor regions have the same width, this subpath from a point p' to the right of the right-most point of d' to p_3 would not be monotone, and as such it would violate the constraint that the path π is monotone.

We now discuss the alternate case (illustrated in Figure 6.7) where the left-most point of d' has x-coordinate equal to or less than the left most point of d, i.e., $l_{x_{d'}} \leq l_{x_d}$.

It must be that the left-most point of d' is disjoint from d in the case that the left-most point of d' is to the left of d, as all sensor regions have equal width. In the case that the left-most point of d has the same x-coordinate as

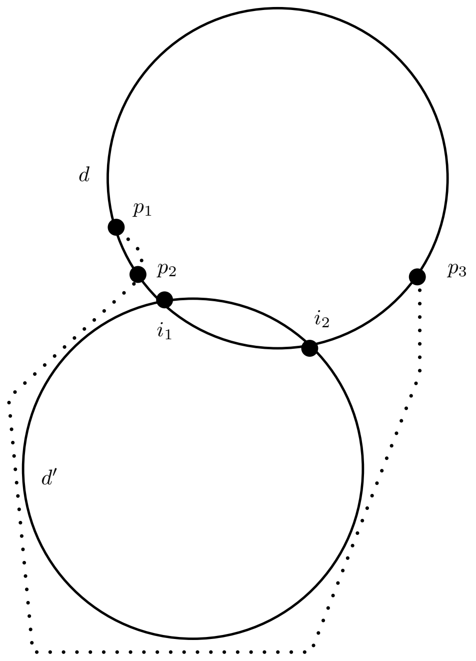

Figure 6.7: A path from p_1 to p_3 around d', where $l_{x_{d'}} \leq l_{x_d}$.

the left-most point of d', then either the left-most and right-most points of d' are disjoint from d, or more than two intersection points are introduced and the set of sensor regions no longer forms a set of pseudo-disks. For analogous reasons to the previous case, it must also be that p_2 is above d'. If this were not the case there would be some boundary segment of d' above p_2, contained clockwise between the intersection points i_1 and i_2, and from Lemma 6.10 is entirely within d. The left-most point of d' cannot be on this boundary segment as we know that the left most point of d' is disjoint from d.

For analogous reasons to the previous case, a monotone path between p_2 and p_3 cannot only consist of subpaths over d'. Therefore there must be a subpath of π that is under d'. To have a subpath under d', it must be that the path π, has a point p' to the left of the left-most point of d'. As p_2 is to the right of the left most point of d', such a subpath from p_2 to the point p' that is to the left of the left-most point of d' would not be monotone, and consequently would violate the constraint that π is monotone.

As in both cases the existence of a monotone path between p_2 and p_3 is not possible, the claim follows.

$\square$

Following immediately from Theorem 6.8, we will obtain the following Corollary helping to relate the thickness and resilience of a barrier in our setting.

Corollary 6.12. *Given a set D of convex pseudo-disks all of equal width, a shortest monotone path from the starting point s to the target point t, that does not intersect any obstacles D^*, will enter each sensor region $d \in D$ at most once.*

From Corollary 6.12 and the discussion at the beginning of this section, we can apply a shortest path algorithm to the dual-graph to calculate the thickness of the barrier, and as we know the resilience of the network is equal to the thickness of the network, Theorem 6.6 follows, i.e, there exists a polynomial-time algorithm for computing the monotone barrier resilience of a sensor network consisting of sensor regions of equal width, that are a set of pseudo disks.

6.5 Conclusion

We have shown the existence of a polynomial time algorithm that can determine the monotone barrier resilience of a sensor network, whose sensor regions are of a convex shape, equal width and form a set of pseudo disks. This is in contrast to the problem of determining the barrier resilience which is known to be $\mathcal{NP}$-hard for general paths, when considering non-symmetrical sensor regions [TK11]. With respect to the problem of determining the monotone barrier resilience of a sensor network, a problem of interest would be to ascertain the complexity of sensor regions of equal width that are concave. Additionally, if the restriction of requiring that the arrangement of sensor regions forms a set of pseudo disks was relaxed, the complexity of the monotone barrier resilience problem would be open. Furthermore, our proof holds only for the case that all the sensor regions are of equal width. As such, a natural progression of this work would be to determine the complexity of the problem were one to consider sensor regions of arbitrary width.

Regarding the problem of determining the barrier resilience of a sensor network for general paths, as far as we are aware, the complexity of computing barrier resilience is still open for sensor regions of unit disks. As unit disks are a natural representation of sensors with omnidirectional antennae, it would be of interest to show either the problem to be $\mathcal{NP}$-hard, or to show an exact polynomial time algorithm that determines the barrier resilience of such a network.

Chapter 7

Conclusion

We have studied four algorithmic problems, that are motivated by real life issues arising from wireless communication and sensor networks. We have provided and analysed approximation algorithms, online algorithms and exact algorithms.

In Chapter 3 we considered the problem of scheduling multicast links, such that all transmissions are correctly received with respect to interference in the network. We also considered the difference between atomic multicast transmissions, and splittable mutlicast transmissions where the atomic condition is relaxed and it is only required that every receiver in the set of receivers, correctly receives the transmission in some round. We have shown that existing work for the unicast case can be generalised to the multicast case, and yields an $\mathcal{O}(\log \Gamma)$-approximation algorithm for the scheduling problem. In addition, we have shown that a schedule containing atomic multicast transmissions is at most $\mathcal{O}(\log \Gamma)$ longer than a schedule that allows splittable multicast requests. It would be interesting to know whether this factor can be reduced. The overriding open question with regard to assigning powers and scheduling transmission links is to provide a constant-factor approximation algorithm for the scheduling problem of unicast links. It would also be interesting to know if the approach used to show a constant factor approximation algorithm for the throughput problem in the unicast-case [Kes11], can be generalised to the case of multicast transmission requests.

In Chapter 4 we studied a generalisation of the problem of covering targets with unit disks, where each target must be covered once and there is only one event type to be monitored. Specifically, we considered the problem of covering targets with at least two disks, and with t-event types. We presented a $(6 + \varepsilon)$-approximation algorithm for the problem that matches the approximation ratio of a previously presented algorithm for the problem with one event type and covering each point only once. Additional algorithms have been presented for the problem that introduce techniques that improve the approximation ratio (to $(5+\varepsilon)$ [DY09] and $(4+\varepsilon)$ [EM09]). It would be of interest to know if these improved results for the single cover, single event type problem extend to the case of covering each point twice in a setting with multiple event types.

In Chapter 5 we considered the strongly connected online in out dominating problem. We have provided a lower bound that shows that no deterministic algorithm can achieve competitive ratio less than $n/4+1/4$. We then stated an online algorithm that attains competitive ratio $n/4 + 1/4$, which complements the aforementioned lower bound. A natural continuation of this work would be to study randomised online algorithms for the same problem. A further potential direction for research would be to study the offline problem of determining a strongly connected dominating set of minimum size, where every vertex not in the dominating set has a directed edge from a vertex in the dominating set (or a directed edge to a vertex in the dominating set). One could investigate if techniques presented by Li et al. [LDW$^+$09] can be adapted to this setting, where the problem is to specify a strongly connected in-out dominating set of minimum size.

For the final problem presented in Chapter 6 of this thesis we studied the problem of determining the monotone barrier resilience of a sensor network. We have shown that one can determine the monotone barrier resilience of a sensor network in polynomial time, for instances formed by a set of convex pseudo disks, all of equal width. It would be of interest to know the complexity of monotone barrier resilience in different settings, possibly with concave objects, or convex objects of arbitrary width. With regard to the problem of determining the barrier resilience of sensor networks for general paths, the complexity of sensors networks consisting of unit disks would be of interest.

Bibliography

[AD09] M. Andrews and M. Dinitz. Maximizing capacity in arbitrary wireless networks in the SINR model: Complexity and game theory. In *Proceedings of the 28th Conference of the IEEE Communications Society (INFOCOM 2009)*, pages 1332–1340, April 2009.

[AEMN06] C. Ambühl, T. Erlebach, M. Mihalák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2006)*, LNCS 4110, pages 3–14. Springer, 2006.

[AGR00] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 705–706. Society for Industrial and Applied Mathematics, 2000.

[AKK04] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, 2004.

[ALP09] C. Avin, Z. Lotker, and Y.-A. Pignolet. On the power of uniform power: Capacity of wireless networks with bounded resources. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA 2009)*, LNCS 5757, pages 373–384. Springer, 2009.

[AMS06] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG 2006)*, 2(2):153–177, 2006.

[AY05] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[BCSZ04] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky. Power efficient monitoring schedules in sensor networks. In *IEEE Wireless Communication and Networking Conference (WCNC 2004)*, pages 2329–2334, 2004.

[BCSZ05] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky. Efficient energy management in sensor networks. In Y. Xiao and Y. Pan, editors, *Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing*, volume 2. Nova Science Publishers, 2005.

[BEY98] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*, volume 53. Cambridge University Press New York, 1998.

[BGRS10] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 583–592. ACM, 2010.

[BK09] S. Bereg and D. Kirkpatrick. Approximating barrier resilience in wireless sensor networks. In *Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2009)*, LNCS 5084, pages 29–40. Springer, 2009.

[CCHP09] C. Chekuri, K.L. Clarkson, and S. Har-Peled. On the set multicover problem in geometric settings. In *Proceedings of the 25th Annual Symposium on Computational Geometry (SCG'09)*, pages 341–350. ACM, 2009.

[CCJ90] B.N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graphs. *Discrete mathematics*, 86(1):165–177, 1990.

[CD05] M. Cardei and D.Z. Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, May 2005.

[Chu36] A. Church. An unsolvable problem of elementary number theory. *American journal of mathematics*, 58(2):345–363, 1936.

[CJ03] E.H. Callaway Jr. *Wireless sensor networks: architectures and protocols*, volume 3. Auerbach publications, 2003.

[CKM$^+$07] D. Chafekar, VS Kumar, M.V. Marathe, S. Parthasarathy, and A. Srinivasan. Cross-layer latency minimization in wireless networks with SINR constraints. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 2007)*, pages 110–119. ACM, 2007.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158. ACM, 1971.

[CTLW05] M. Cardei, M. T. Thai, Y. Li, and W. Wu. Energy-efficient target coverage in wireless sensor networks. In *Proceedings of INFOCOM 2005*, volume 3, pages 1976–1984. IEEE, March 2005.

[CW04] M. Cardei and J. Wu. Coverage in wireless sensor networks. *Handbook of Sensor Networks*, pages 422–433, 2004.

[Die05] R. Diestel. Graph theory, vol. 173 of. *Graduate Texts in Mathematics*, 2005.

[DTL$^+$06] D.Z. Du, M. Thai, Y. Li, D. Liu, and S. Zhu. Strongly connected dominating sets in wireless sensor networks with unidirectional links. In *Frontiers of WWW Research and Development (APWeb 2006)*, LNCS 3841, pages 13–24. Springer, 2006.

[DVZ$^+$06] A. Dhawan, C. T. Vu, A. Zelikovsky, Y. Li, and S. K. Prasad. Maximum lifetime of sensor networks with adjustable sensing range. In *SNPD-SAWN 2006*, 2006.

[DY09] D. Dai and C. Yu. A $(5+\epsilon)$-approximation algorithm for minimum weighted dominating set in unit disk graph. *Theoretical Computer Science*, 410(8-10):756–765, 2009.

[EG10] T. Erlebach and T. Grant. Scheduling multicast transmissions under SINR constraints. In *Proceedings of the 6th International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks, and Autonomous Mobile Entities (ALGOSENSORS 2010)*, LNCS 6451, pages 47–61. Springer, 2010.

[EGK11a] T. Erlebach, T. Grant, and F. Kammer. Maximising lifetime for fault-tolerant target coverage in sensor networks. In *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, pages 187–196. ACM, 2011.

[EGK11b] T. Erlebach, T. Grant, and F. Kammer. Maximising lifetime for fault-tolerant target coverage in sensor networks. *Sustainable Computing: Informatics and Systems*, 1(3):213–225, 2011.

[Eid02] S. Eidenbenz. Online dominating set and variations on restricted graph classes. *Technical Report 280, Department of Computer Science, ETH Zurich*, 2002.

[EM09] T. Erlebach and M. Mihalák. A $(4+\epsilon)$-approximation for the minimum-weight dominating set problem in unit disk graphs.

In *Proceedings of the 7th International Workshop on Approximation and Online Algorithms (WAOA 2009)*, LNCS 5893, pages 135–146. Springer, 2009.

[Eng89] Ryszard Engelking. *General topology.* Heldermann, 1989.

[ES09] T. Erlebach and A. Shahnaz. Approximating Node-Weighted Multicast Trees in Wireless Ad-Hoc Networks. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly (IWCMC 2009)*, pages 639–643. ACM, 2009.

[Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[FGHV10] A. Fanghänel, S. Geulen, M. Hoefer, and B. Vöcking. Online capacity maximization in wireless networks. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures (SPAA 2010)*, pages 92–99. ACM, 2010.

[FKV09] A. Fanghänel, T. Kesselheim, and B. Vöcking. Improved algorithms for latency minimization in wireless networks. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009)*, LNCS 5556, pages 447–458. Springer, 2009.

[GHN08] Y. Gao, J.C. Hou, and H. Nguyen. Topology control for maintaining network connectivity and maximizing network capacity under the physical model. In *the 27th Conference on Computer Communications (INFOCOM 2008)*, pages 1013–1021. IEEE, 2008.

[GHWW09] O. Goussevskaia, M. Halldórsson, R. Wattenhofer, and E. Welzl. Capacity of arbitrary wireless networks. In *the 28th Conference on Computer Communications (INFOCOM 2009)*, pages 1872–1880, April 2009.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, CA, 1979.

[GK98a] N. Garg and J. Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS 1998)*, pages 300–309. IEEE, 1998.

[GK98b] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[GK00] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[GMW08] O. Goussevskaia, T. Moscibroda, and R. Wattenhofer. Local Broadcasting in the Physical Interference Model. In *ACM SIGACT-SIGOPT International Workshop on Foundations of Mobile Computing (DialM-POMC), Toronto, Canada*, August 2008.

[GOW07] O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer. Complexity in geometric SINR. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 100–109. ACM, 2007.

[Hal09] M. Halldórsson. Wireless Scheduling with Power Control. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA 2009)*, LNCS 5757, pages 368–380. Springer, 2009.

[HGZW09] Y. Huang, X. Gao, Z. Zhang, and W. Wu. A better constant-factor approximation for weighted dominating set in unit disk graph. *Journal of Combinatorial Optimization*, 18(2):179–194, 2009.

[HM85] D.S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.

[HW09] M. Halldórsson and R. Wattenhofer. Wireless Communication is in APX. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009)*, LNCS 5555, pages 525–536. Springer, 2009.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 40(4):85–103, 1972.

[Kes11] 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*, pages 1549–1559. SIAM, 2011.

[KLA05] S. Kumar, T.H. Lai, and A. Arora. Barrier coverage with wireless sensors. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 284–298. ACM, 2005.

[KLA07] S. Kumar, T.H. Lai, and A. Arora. Barrier coverage with wireless sensors. *Wireless Networks*, 13(6):817–834, 2007.

[KT97] G.H. King and W.G. Tzeng. On-line algorithms for the dominating set problem. *Information Processing Letters*, 61(1):11–14, 1997.

[Kum06] S. Kumar. *Foundations of coverage in wireless sensor networks.* PhD thesis, The Ohio State University, 2006.

[LDW$^+$09] D. Li, H. Du, P.J. Wan, X. Gao, Z. Zhang, and W. Wu. Construction of strongly connected dominating sets in asymmetric multihop wireless networks. *Theoretical Computer Science*, 410(8-10):661–669, 2009.

[LGW04] A. Leon-Garcia and I. Widjaja. *Communication Networks.* McGraw-Hill, Inc., New York, NY, USA, 2 edition, 2004.

[MBHI$^+$95] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, and D.J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.

[MKPS01] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and MB Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *the 12th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 3, pages 1380–1387, 2001.

[MKQP01] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom 2001)*, pages 139–150. ACM, 2001.

[Mos07] T. Moscibroda. The worst-case capacity of wireless sensor networks. In *6th International Symposium on Information Processing in Sensor Networks (IPSN 2007)*, pages 1–10. IEEE, 2007.

[MW06] T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–13. IEEE, 2006.

[MWZ06] T. Moscibroda, R. Wattenhofer, and A. Zollinger. Topology control meets SINR: the scheduling complexity of arbitrary topologies. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 2006)*, pages 310–321. ACM, 2006.

[MYC09] M. Marta, Y. Yang, and M. Cardei. Energy-efficient composite event detection in wireless sensor networks. *Wireless Algorithms, Systems, and Applications*, pages 94–103, 2009.

[RDJ$^+$04] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K.I. Ko. A greedy approximation for minimum connected dominating sets. *Theoretical Computer Science*, 329(1):325–330, 2004.

[RS09] G. Resta and P. Santi. Latency and capacity optimal broadcasting in wireless multihop networks. In *Proceedings of IEEE International Conference on Communications (ICC 2009)*, pages 1–6. IEEE, 2009.

[Sch98] A. Schrijver. *Theory of linear and integer programming.* Wiley, 1998.

[SS10] P. Sanders and D. Schieferdecker. Lifetime maximization of monitoring sensor networks. In *Proceedings of the 6th International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks, and Autonomous Mobile Entities (ALGOSENSORS 2010)*, LNCS 6541, pages 134–147. Springer, 2010.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[Sut75] W.A. Sutherland. *Introduction to metric and topological spaces.* Oxford University Press, USA, 1975.

[TK11] K.C.R. Tseng and D. Kirkpatrick. On barrier resilience of sensor networks. In *Proceedings of the 7th international conference on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities (ALGOSENSORS 2011)*, pages 130–144. Springer, 2011.

[TTD08] M.T. Thai, R. Tiwari, and D.Z. Du. On construction of virtual backbone in wireless ad hoc networks with unidirectional links. *IEEE Transactions on Mobile Computing*, 7(9):1098–1109, 2008.

[Tur36] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[TWDJ08] M.T. Thai, F. Wang, D.H. Du, and X. Jia. Coverage problems in wireless sensor networks: designs and analysis. *International Journal of Sensor Networks*, 3(3):191–200, May 2008.

[TWL$^+$07] M.T. Thai, F. Wang, D. Liu, S. Zhu, and D.Z. Du. Connected dominating sets in wireless networks with different transmission ranges. *IEEE Transactions on Mobile Computing*, 6(7):721–730, 2007.

[Vaz04] V.V. Vazirani. *Approximation algorithms.* Springer, 2004.

[VBL07] C.T. Vu, R.A. Beyah, and Y. Li. Composite event detection in wireless sensor networks. In *IEEE International Performance,* *Computing, and Communications Conference (IPCCC 2007)*, pages 264–271, 2007.

[Wes01] D.B. West. *Introduction to Graph Theory*, volume 2. Prentice-Hall, Upper Saddle River, NJ, 2001.

[ZG08] Q. Zhao and M. Gurusamy. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(6):1378–1391, 2008.

[ZLGW09] F. Zou, X. Li, S. Gao, and W. Wu. Node-weighted Steiner tree approximation in unit disk graphs. *Journal of Combinatorial Optimization*, 18(4):342–349, 2009.

[ZWX$^+$11] F. Zou, Y. Wang, X.H. Xu, X. Li, H. Du, P. Wan, and W. Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoretical Computer Science*, 412(3):198–208, January 2011.