Optimization Problems in Communication Networks and Multi-Agent Path Finding

Marika Ivanova

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2019

UNIVERSITY OF BERGEN

Optimization Problems in Communication Networks and Multi-Agent Path Finding

Marika Ivanova

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 13.09.2019

Year: 2019

Title: Optimization Problems in Communication Networks and Multi-Agent Path Finding

Name: Marika Ivanova

Print: Skipnes Kommunikasjon / University of Bergen

Acknowledgements

Firstly, I would like to express my deep and sincere gratitude to my adviser Professor Dag Haugland for his guidance and encouragement during the four years I spent as his PhD student. Without his constant feedback and immense patience, this thesis and the publications written throughout my doctoral education would not have been attainable. I learned a lot from my adviser, and gained valuable experience for which I am deeply indebted. His hardworking, passionate, and precise attitude inspired me greatly in my academic growth.

I am also very grateful to doc. RNDr. Pavel Surynek, PhD, from Czech Technical University, for giving me the precious and unforgettable opportunity to spend three months as an intern in Artificial Intelligence Research Center in Japan. As a co-author of two papers resulting from my internship, he contributed to this thesis as well.

My sincere thanks goes to the entire Department of Informatics. It was a pleasure to work in such an inspiring and accepting environment. In particular, I would like to mention Professor Jan-Joachim Rückmann for his advice and affability, and my fellow doctoral students from the Optimization group for being good friends and making my days enjoyable and exciting. In addition, I greatly appreciate the work of administration staff for helping me with any organisational issue.

I must not forget to thank my husband Jahan Zeb for his endless patience and support during the ups and downs of the entire PhD. Last but not least my parents have always believed in me, encouraged and motivated me in my studies. Anything that I have achieved in my life would be impossible without their unconditional support.

Abstract

This dissertation is a compilation of six research papers that are focused on three different topics summarized in the text.

The first three papers address NP-hard problems arising in ad-hoc wireless communication discussed in Chapter 2. In general, the task is to broadcast a message in a given network of wireless devices while minimizing the power consumption. Problems in this category differ in requirements on the network connectivity, models of power consumption, and the ability of the devices to initiate a signal transmission. Some of the common features of these problems are that a device can simultaneously transmit a signal to all devices within its communication vicinity, and that a signal can travel from its originator to its recipient via multiple intermediate devices. The wireless networks are modeled and studied by means of graph theory. Solution techniques for these problems involve mainly methods of integer linear programming and inexact algorithms with or without performance guarantee.

The next paper is focused on the problem of minimum broadcast time. Unlike the previous topic, the devices are in this problem supposed to send a signal to at most one neighbouring device at a time. The objective is to determine a sequence of signal transmission from a given set of source devices to the remaining ones, while minimizing the time needed for spreading the signal. Chapter 3 describes this problem in detail along with several related problems. The minimum broadcast time problem is also studied from the perspective of integer linear programming as well as the inexact algorithm perspective. Continuous relaxations of the ILP models help to evaluate the quality of the studied inexact methods. The stronger the model is, the more accurate assessment it provides.

The last two papers are dedicated to problems belonging to path planning for multiple robots discussed in Chapter 4. In general, these problems involve a group of agents (robots) initially deployed in an environment. The task is to find a sequence of their moves so that they reach pre-defined destination locations while optimizing a given criterion such as minimum makespan or minimum total arrival time. The agents' movement must obey a set of given rules. An extension of the problem considers agents divided into two (or more) adversarial teams, where the teams have either symmetric or asymmetric objectives. After introducing the adversarial element, the problem of finding a winning strategy for a given team becomes PSPACE-hard, like many other two player games with alternating turns.

Contents

Chapter 1

Preliminaries

1.1 Graph Terminology

A *graph* G is a pair (V,E) of *vertices* [1] and *edges*, where $E \subseteq \binom{V}{2}$. If this inclusion is an equality, G is said to be *complete*. The set A of arcs is derived from E by considering both directions of orientation of the edges. Formally, $A = \{(i,j),(j,i) : \{i,j\} \in E\}$. The notation $|G|$ is sometimes used for denoting the number of nodes in G, thus, $|G| = |V|$. A *path* in a graph is a sequence of edges connecting a sequence of distinct vertices. A path from u to v is denoted by $P_{u,v}$. The *length* of a path is the number of its edges. Consider the set of shortest paths between any two nodes in a graph G. The length of the longest among these shortest paths is called the *graph diameter*, and is usually denoted Δ_G. A graph is said to be *connected* if there exists a path between every two vertices, otherwise it is *disconnected*. A *cycle* in G is a subset of E that forms a path such that the first vertex of the path equals the last one. If G contains a cycle, G is called *cyclic*, otherwise it is called *acyclic*.

Definition 1. *A tree is a graph that is connected and acyclic.*

For a vertex $v \in V$, a *neighbourhood* of v (open neighbourhood), denoted as $N(v)$, is the set of vertices adjacent to v. The size of neighbourhood of v, $\deg(v)$, is called a *degree* of v. A *subgraph* of $G = (V,E)$ is a graph $G' = (V',E')$ such that $V' \subseteq V$ and $E' \subseteq E$. This relation is often written as $G' \subseteq G$.

Definition 2. *A spanning tree of graph $G = (V,E)$ is a tree $T = (V_T,E_T)$ such that $T \subseteq G$ and $V_T = V$.*

A *bipartite graph* is a graph with vertices decomposed into two disjoint sets such that no two vertices within the same set are adjacent. Every acyclic graph is bipartite. A cyclic graph is bipartite if and only if it does not contain a cycle of odd length. If all cycles on four or more vertices in G contain an edge which is not a part of the cycle but connects two vertices of the cycle, then G is called a *chordal graph*. A *clique* in G is a subset $V' \subseteq V$ of vertices such that there is an edge between every two distinct vertices of the clique. Conversely, if no two nodes in V' are adjacent, V' is an *independent set*. Vertices in a *split graph* can be partitioned into a clique and an independent set.

[1] We use the term *vertex* when a graph is considered as an abstract structure. Whenever a graph is discussed in a context in which it represents real objects, we refer to vertices as *nodes*.

For a given subset $V' \subseteq V$ of nodes in G, $G[V']$ denotes the subgraph of G *induced by* V' consisting of nodes in V' and edges in E whose both endpoints are in V'.

In a *weighted graph* G, a *weight* or *cost* $w : E \mapsto \mathbb{R}$ is associated witch each edge $e \in E$. We use the terms *heavier* and *heaviest* when comparing weights of different edges in a graph. The weight of G is defined as $\sum_{e \in E} w(e)$. A spanning tree of G with minimum weight is called a *minimum spanning tree* (MST) of G. Analogous concept is defined for paths in graphs. A *shortest path* from u to v in a weighted graph is a path of minimum weight connecting u and v.

Definition 3. *For a graph $G = (V,E)$ and a subset of vertices $D \subseteq V$, a Steiner tree of G and D is a tree $T = (V',E')$ such that $T \subseteq G$ and $D \subseteq V'$.*

Analogously in weighted graphs, a *minimum Steiner tree* is a Steiner tree of minimum weight.

A *directed graph* is graph, where all the edges are directed from one vertex to another. A directed graph is sometimes called a *digraph*, and its edges are referred to as *arcs*. Let $\overrightarrow{G} = (V,A)$ be a directed graph. The upstream neighbourhood $N^-(v)$ of node v is the set $\{u \in V : (u,v) \in A\}$. Similarly, the downstream neighbourhood $N^+(v)$ is $\{u \in V : (v,u) \in A\}$. We use the standard notation $\deg^-(v) = |N^-(v)|$ and $\deg^+(v) = |N^+(v)|$, called the *in-degree* and the *out-degree* of v, respectively. An *arborescence* rooted at vertex $r \in V$ is a directed tree with arcs directed from r. $\overrightarrow{G}$ is *strongly connected*, if for every pair of vertices $u, v \in V$, there exists a path from u to v and from v to u in $\overrightarrow{G}$.

An *embedding* in plane of a graph G is determined by a function $\Phi : V \mapsto \mathbb{R} \times \mathbb{R}$ that assigns a coordinate to each node in V. The open and closed line segment between $\Phi(u)$ and $\Phi(v)$ is denoted by $\Phi(u,v)$ and $\Phi[u,v]$, respectively. The length of the line segment $\Phi(u,v)$ is denoted by $d(u,v)$.

Definition 4. *Φ is a planar embedding of G, if for all $\{u_1,v_1\},\{u_2,v_2\} \in E$, where $\{u_1,v_1\} \neq \{u_2,v_2\}$, $\Phi(u_1,v_1) \cap \Phi(u_2,v_2) = \emptyset$. If such an embedding exists then G is planar.*

Every tree has a planar embedding and is therefore planar. However, not every embedding of a tree is planar. For an embedding that is not planar, $\Phi(u_1,v_1) \cap \Phi(u_2,v_2) \neq \emptyset$ is called *crossing*. A *Euclidean graph* is a graph together with its embedding Φ, and edges are assigned weights equal to the Euclidean distance between their endpoints given by Φ.

1.2 Combinatorial Optimization

Combinatorial optimization (CO) is a part of applied mathematics that tackles optimization problems over discrete structures. It combines methods from graph theory, linear programming, combinatorics, and the theory of algorithms. In this section, we briefly introduce main concepts in CO used later in the text. For a comprehensive rendition of this topic, interested readers are referred to [77] and [54].

Combinatorial problems arise in many areas of computer science, with a wide range of applications in various industrial disciplines such as production scheduling, logistics,

communication network design, and many more. The core of tackling a problem by methods of CO is the identification of a discrete mathematical structure hidden in the problem, and finding a sufficient abstraction.

CO concerns problems of minimization or maximization of an *objective function* of several variables subject to inequality and equality *constraints* and integrality restrictions on at least some of the variables. In this work, both the objective function and the constraints are assumed to be linear. Combinatorial problems are often formulated as *mixed integer linear programs* (MILP, sometimes abbreviated as ILP or IP) of the standard form

$$\begin{aligned} \max_{x,y}\ & c^\top x + h^\top y \\ & \text{subject to} \\ & Ax + By \leq b, \\ & x \in \mathbb{Z}^n_+, y \in \mathbb{R}^p_+. \end{aligned} \tag{1.1}$$

The problem instance is specified by the input data $c \in \mathbb{R}^n$, $h \in \mathbb{R}^p$, $A \in \mathbb{R}^{m\times n}$ $B \in \mathbb{R}^{m\times p}$ and $b \in \mathbb{R}^m$, $m,n,p \in \mathbb{N}$. A MILP that is not in the standard form, for example if the objective is to minimize or if the constraints contain equalities, can be straightforwardly converted into the standard form. If the integrality constraints are not present, (1.1) is a *linear program* (LP).

The set of points $S = \{(x,y) : x \in \mathbb{Z}^n_+, y \in \mathbb{R}^p_+, Ax + By \leq b\}$ is called the *feasible region*, and a point $(x,y) \in S$ is referred to as a *feasible point* (*feasible solution*) with *objective function value* $c^\top x + h^\top y$. A feasible point (x^*,y^*) is called an *optimal solution* if for every feasible points (x,y) we have that $c^\top x + h^\top y \leq c^\top x^* + h^\top y^*$. Then, $c^\top x^* + h^\top y^*$ is called the *optimal objective function value*.

1.2.1 Relaxation and Bounds

Definition 5. *Let $\mathcal{F}$ be the MILP* $\max\{c^\top x + h^\top y : (x,y) \in S\}$. *The problem* $\mathcal{R}$: $\max\{g(x,y) : (x,y) \in T\}$ *is a relaxation of $\mathcal{F}$ if and only if*

1. *$T \supseteq S$, and*
2. *$g(x,y) \geq c^\top x + h^\top y$ for all $(x,y) \in S$.*

Let z^* and $\underline{z}$ be the optimal objective function value of a MILP and its relaxation, respectively. Further, let $\bar{z}$ be the objective function value of some feasible point. Then, $\underline{z} \leq z^* \leq \bar{z}$. Values $\underline{z}$ and $\bar{z}$ are referred to as a *lower bound* and an *upper bound* on z^*, respectively.

A *combinatorial relaxation* of a MILP is achieved by omitting one or more constraints. By omitting the integrality constraints of a MILP $\mathcal{F}$, we obtain its *continuous relaxation*, also called the *LP relaxation*, denoted as LP($\mathcal{F}$).

1.2.2 Duality

Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Consider an LP (*primal*)

$$\max\left\{c^\top x : Ax \leq b, x \geq 0\right\}. \tag{1.2}$$

We are looking for the best upper bound. If x^* is an optimal solution to (1.2), $y^\top Ax$ with $y \in \mathcal{R}^n_+$ is a general linear combination of equations. If it is possible to select a vector y so that $y^\top Ax^* = c^\top x^*$, we have that $y^\top b \geq c^\top x^*$. The best bound for any x is then the optimal solution to the following LP (*dual*)

$$\min \left\{ b^\top y : A^\top y \geq c, y \geq 0. \right\} \tag{1.3}$$

The relation between primal and dual LP is summarized by

Proposition 1. *If the primal has an optimal solution x^* then the dual has an optimal solution y^* such that $c^\top x^* = b^\top y^*$.*

For LPs, duality provides a standard way to obtain upper bounds. A similar concept is applied to IPs.

Definition 6. *[77] The two problems*

$$z = \max\{c(x) : x \in X\} \tag{1.4}$$

and

$$w = \min\{\omega(u) : u \in U\} \tag{1.5}$$

form a (weak)-dual pair if $c(x) \leq \omega(u)$ for all $x \in X$ and all $u \in U$. When $z = w$, they form a strong-dual pair.

For obtaining an upper bound from the LP relaxation, it is necessary to solve the relaxed program to optimality, whereas any dual feasible solution provides an upper bound on z.

Proposition 2. *[77] The IP $z = \max\{cx : Ax \leq b, x \in Z^n_+\}$ and the LP $w^{LP} = \min\{ub : uA \geq c, u \in R^m_+\}$ form a weak dual pair.*

Proposition 3. *[77] Suppose that problems (1.4) and (1.5) form a weak-dual pair.*

1. *If w is unbounded, (1.4) is infeasible, i.e., $X = \emptyset$.*
2. *If $x^* \in X$ and $u^* \in U$ satisfy $c(x^*) = \omega(u^*)$, then x^* is optimal in (1.4) and u^* is optimal in (1.5).*

1.2.3 Solution Methods

Several effective methods for solving LPs are used in practice. Among these are the *simplex method* and the *interior point method*. The simplex method sequentially tests adjacent vertices of the feasible region (a convex polytope) so that at each new vertex the objective function is either improved or unchanged. The simplex method is very efficient in practice, although its worst-case complexity is exponential.

The interior point method constructs a sequence of feasible points lying inside of the polytope but never on its boundary, that converges to the solution. Its worst-case time complexity is polynomial.

A MILP can be solved by the *branch and bound* (B&B) method, which systematically enumerates candidate solutions by means of state space search. The set of

candidate solutions gradually forms a rooted tree with the full set at the root. The algorithm explores branches of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, a bound on the best possible result of the branch is calculated and compared with estimated upper and lower bounds on the optimal solution. If a solution better than the best one found so far by the algorithm cannot be produced, the entire branch is discarded. Performance of the algorithm depends on efficient estimation of the lower and upper bounds of branches of the search space. If bounds cannot be calculated, the algorithm becomes an exhaustive search.

These and other algorithms are integral parts of most modern solvers such as CPLEX and GUROBI.

1.3 Problems and Complexity

A *computational problem* (problem) is an infinite collection of instances together with a solution for each instance. A problem that can be posed as a yes-no question of the input values is referred to as a *decision problem*. An example of a decision problem is the CLIQUE problem: Given a graph G and an integer k, is there a clique in G of size at least k? In *optimization problems*, the task is to find a "best possible" solution in the set of all feasible solutions to the problem instance. The optimization version of CLIQUE asks for a maximum clique in a given graph G. An optimization problem can be solved by answering a sequence of decision problems: Assume there is an oracle that is able to solve the CLIQUE problem for a given (G,k). The MAXIMUM CLIQUE problem can then be solved by answering its decision version for $k = 1,2,\ldots$ until the answer is "no" for some $k = k'$, and so the maximum clique has the size $k'-1$.

Throughout this thesis, we use several well known concepts from complexity theory, which we state in the following. Detailed explanations of the terminology can be found in any textbook on this topic, such as [67].

An *algorithm* is a procedure that solves a given problem in a finite number of steps. The computational complexity of an algorithm is the amount of time needed for its run, and is measured in terms of the input size. A *polynomial* algorithm runs in time $\mathcal{O}(n^c)$, for some constant c and input w of size $|w| = n$. A *verifier* is an algorithm that determines whether a given certificate is a proof to the fact that w is a yes-instance. An example of a certificate to CLIQUE is some subset of nodes of size k. It can be verified in polynomial time whether there exists an edge between every two nodes.

Definition 7. *P is the class of decision problems for which there exists a polynomial algorithm that solves them.*

Definition 8. *NP is the class of decision problems for which there exists a polynomial verifier.*

Definition 9. *Problem X is polynomial time reducible to problem Y, if a polynomial computable function f exists where for every w, w is a yes-instance of X if and only if $f(w)$ is a yes instance of Y.*

Definition 10. *A decision problem Y is NP-complete if it satisfies:*

1. *Y is in NP.*

2. *Every X in NP is polynomial time reducible to Y.*

Remark: A verifier does not decide whether a certificate is an optimal solution to a given optimization problem instance. When addressing optimization problems, we consider only the second property in Def. 10. If this property is satisfied, we say that the problem is *NP-hard*.

There are many well know examples of NP-hard problems, and they can be formulated as ILP. Therefore, ILP itself is also NP-hard.

The space complexity of an algorithm is the amount of memory space required for its run as a function of the size of the input. An algorithm runs in *polynomial space* if for input of size n requires the space $\mathcal{O}(n^c)$, for some constant c.

Definition 11. *PSPACE is the class of decision problems for which there exists an algorithm that solves them in a polynomial space.*

Definition 12. *A decision problem Y is PSPACE-complete if it satisfies:*

1. *Y is in PSPACE.*
2. *Every X in PSPACE is polynomial time reducible to Y.*

If Y satisfies condition 2 in Def. 12 we say that X is *PSPACE-hard*.

Approximation algorithms are polynomial algorithms that find approximate solution to NP-hard optimization problems with provable guarantee on the distance of the solution to the optimal one.

Definition 13. *[76] Let $\rho \geq 0$ be a real number. A ρ-approximation algorithm for an optimization problem is a polynomial-time algorithm, that for all instances of the problem produces a solution whose value is within a factor of ρ of the value of an optimal solution.*

The factor ρ is called the *approximation ratio* or *performance guarantee*. For minimization problems, $\rho > 1$, and for maximization problems, $\rho < 1$. An *NP-optimization problem A* is a quadruple $(I, sol, m, goal)$ such that [20]

1. I is the set of the instances of A and it is recognizable in polynomial time.
2. Given an instance x of I, $sol(x)$ denotes the set of feasible solutions of x. These solutions are short, that is, the size of any $y \in sol(x)$ is polynomial with respect to the size of x. Moreover, it can be determined in polynomial time whether, for any x and for any y, $y \in sol(x)$.
3. Given an instance x and a feasible solution y of x, $m(x,y)$ denotes the positive integer measure of y. The function m is computable in polynomial time.
4. $goal \in \{\min, \max\}$.

Definition 14. *The class APX is the set of NP optimization problems for which there exists an approximation algorithm with constant approximation ratio.*

There are problems that are hard to approximate. A problem for which there is a constant ρ such that it is NP-hard to find an approximation algorithm with approximation ratio better than ρ is said to be *APX-hard*.

Definition 15. *[76] A polynomial-time approximation scheme (PTAS) is a family of algorithms in which there is an algorithm $\mathcal{A}_\varepsilon$ for each $\varepsilon > 0$, such that $\mathcal{A}_\varepsilon$ is a $(1+\varepsilon)$-approximation algorithm for minimization problems, and a $(1-\varepsilon)$-approximation algorithm for maximization problems.*

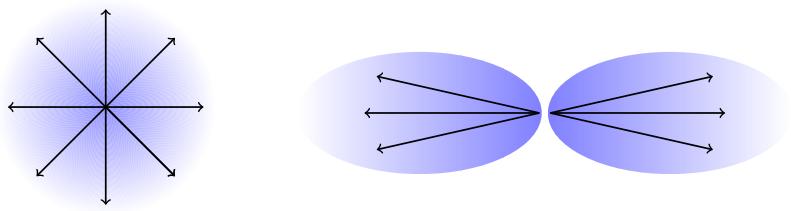
Proposition 4. *APX-hard problems do not admit a PTAS.*

Chapter 2

Power Minimizing Trees in Ad-hoc Wireless Networks

A wireless ad-hoc network (WANET) is a collection of two or more devices with wireless communication and networking capabilities. Broadly speaking, there are two models of wireless networks: single-hop and multi-hop. The single-hop model is based on the cellular network model that provides one-hop wireless connectivity between mobile hosts and static nodes known as base stations [15]. This network type relies on a fixed backbone infrastructure that interconnects all base stations by wired links. Unlike cellular and wired networks, WANETs depend on neither a wired infrastructure nor pre-determined interconnectivity, and are thus a decentralized type of wireless network.

The set of uniform wireless communication devices with fixed locations are connected via wireless links depending on distance between the nodes, their transmission power, error control scheme, background noise and interference. Each device is equipped with an omnidirectional antenna with a communication range schematically illustrated in Fig. 2.1. Hence, a signal reaches all nodes within the communication range of its sender. This range is determined by the power assigned to the sender, and this power can be adjusted over time. The power necessary for relaying a signal to multiple devices is the maximum rather than the sum of the powers necessary to reach all intended receivers. This feature is referred to as the *wireless advantage* [75]. Each device works as a transceiver, which means that it can both transmit and receive a signal.



(a) Top view (b) Side view

Figure 2.1: Omnidirectional antenna

The absence of infrastructure implies their quick deployment and simple configuration. The recent years have witnessed an increasing interest in WANETs, motivated by numerous both civil and military applications and by the continual progression in

wireless technologies. Digital battlefield, disaster management, luggage handling in airports, context aware computing and mobile commerce are examples of the growing list of potential applications of WANETs [78].

Consider a group of devices and a specific sender that initiates a signal transmission. A *unicast* means that there is a single recipient. Whenever the recipients are all the remaining devices within the group, we talk about a *broadcast*. Finally, a *multicast* means that only some of the devices must receive the message. The remaining ones do not have to, but they can serve as intermediate devices forwarding the signal.

The wireless devices should be able to detect the presence of other such devices and to perform the necessary initialization to allow information sharing. Since the devices can take different forms, their computation, storage and communication capabilities and battery capacity vary tremendously [71]. The wireless devices are typically heavily energy constrained due to the use of batteries as their energy source. Energy conservation in WANETs is therefore a plentifully pursued research topic. Most of the energy is spent on signal transmission [34], and so it is desirable to design energy efficient transmission protocols that also satisfy certain requirements imposed on a WANET.

Various levels of connectivity are required on WANETs. However, from a practical point of view, a topology which is "too connected" would often cause communication interference to occur even between nodes that are far apart. Theoretical as well as practical experimental results suggest that the communication graph in WANETs should be as sparse as possible while preserving connectivity [9].

Models considered for WANETs are usually deterministic, that is, they assume that the nodes are fully reliable. In reality, the nodes are devices that may be subject to temporary or permanent failure due to technical damage or battery draining. This fact leads to considering *probabilistic* models whose aim is to capture the real world more plausibly by taking into account the uncertain character of nodes' availability. Besides minimizing the power consumptions of the network, it is also required to guarantee a certain level of reliability over the whole network.

2.1 Wireless Sensor Networks

A related paradigm to WANET are Wireless Sensor Networks (WSN) which attracted a wide range of disciplines where close interactions with the physical environment are essential. A WSN consists of tiny sensor nodes, which act as both data generators and network relays [1]. Each node act as a sensor, a microprocessor, and a transceiver. Sensor nodes have usually a fixed position and are powered by batteries. Data measured by sensors are transmitted via wireless communication links.

There are countless of sensor types, including seismic, electromagnetic, and acoustic, suggesting a broad area of practical application areas such as military, environmental, health, home, and industrial applications.

Multiple sensors are typically integrated into higher-level topologies varying in complexity. These topologies can be divided into flat and hierarchical architecture [49]. In a flat (peer to peer) architecture, every node has the same computational and communication capabilities. In a hierarchical architecture, simple sensor nodes operate in close proximity to their respective *cluster heads*, which possess more processing capacity than ordinary sensor nodes do.

WSNs are often built using one of the following configurations [49]:

- *Bus topology.* Each node is connected to a shared communication bus, in which a signal is transmitted in both directions.
- *Ring topology.* A networks set up in a circular fashion is similar to linear topology, but does not contain the terminating nodes.
- *Star topology.* Consists of a single central node such as hub or a switch to which every node in the network is connected. An intelligent central node is required as all data traffic flows through it. This topology is one of the most common WSN topologies.
- *Tree topology.* A hierarchy of nodes where in the highest level of the hierarchy is a single root node connected to one or many nodes in the level below. The processing and power requirements in nodes increase as the data moves from the branches towards the root node.
- *Mesh topology.* Nodes disseminate their own data and also act as relays to propagate the data from other nodes. The mesh topology can be partially connected or fully connected (where there is a connection between every two nodes).
- *Grid topology.* The network area is partitioned into non-overlapping grid squares of equal size. There should be exactly one node in an active state in each grid square at any time.

There is a need to reduce energy consumption so as to enhance the performance of the network in terms of lifetime. The selection of a suitable topology is therefore crucial. Grid topology has been found energy efficient in theoretical comparison [63], however, this may vary according to specific applications.

2.2 Combinatorial Optimization Problems Motivated by WANETs

As indicated above, energy conservation is a crucial requirement in the design of WANETs. Combinatorial optimization is endowed with methods suitable for this task. It is necessary to introduce a formal network model in order to apply suitable mathematical tools.

2.2.1 Network Model

A WANET is modeled by a complete graph $G = (V,E)$, and a function $p : E \mapsto \mathbb{R}^+$. The wireless devices are represented by nodes in V. As no power limit is imposed on the nodes, a transmission can be established between any two nodes, and thus G is complete. The function p indicates the power requirement for sending a signal along a certain edge. This requirement depends on distance and environmental properties, and is given by $p_{ij} = \kappa d_{ij}^{\alpha}$, where d_{ij} is the Euclidean distance between the devices represented by nodes i and j, α is the constant environmentally dependent factor typically valued between 2 and 4, and $\kappa \geq 0$ is the transmission quality parameter. Since the

value of κ does not affect the optimal solution, it is often assumed to be 1. The transmission range of a node i depends on the power supply P_i assigned to it. The overall power assignment to individual nodes induces a directed transmission graph (see the example in Fig. 2.2). On the other hand, for a given transmission graph, the power assignment to a node corresponds to its longest outgoing arc. It is usually assumed that

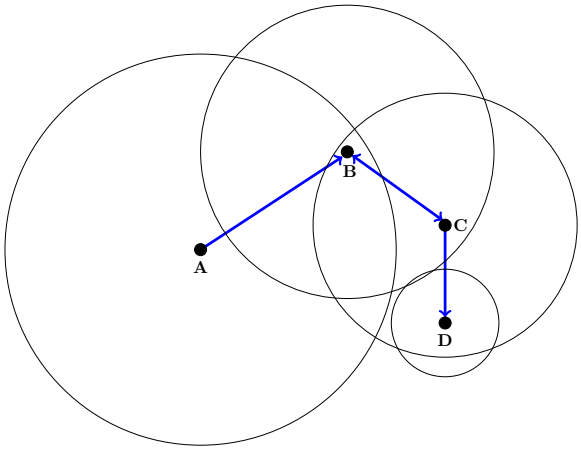


Figure 2.2: Power assignment to nodes (circles) and corresponding induced transmission graph (blue arrows)

nodes are distributed in a Euclidean space, although there are multiple results on variants where general input graphs are considered. In the following combinatorial problems motivated by WANETs, the task is to assign powers to individual nodes while satisfying certain criteria so that the overall power is minimized. The total power is expressed by an objective function varying among the problems.

Multicasting

In certain situations, some of the wireless nodes never initiate any transmission, nor they have to receive any signal. Including them in the network may lead to a more efficient communication. Assumption of the existence of these nodes is often studied as an extension of the problems described in the sections below. In addition to the graph G and the power requirement vector p, the set of nodes $D \subseteq V$, called *destinations*, is also a part of the input. Nodes in $V \setminus D$ are referred to as *non-destinations*. The inclusion of non-destination nodes resembles the generalization of MST leading to the MINIMUM STEINER TREE problem. In WANET scenarios, the presence of such nodes is often referred to as a *multicast* version.

2.2.2 Minimum Energy Broadcast

The MINIMUM ENERGY BROADCAST (MEB) problem consists of a node set V and a specified source $s \in V$, that is supposed to initiate a transmission. All the remaining nodes must receive the signal via communication links induced by power assignment to the nodes.

Problem 1. *Given a directed graph $G = (V,A)$, a source $s \in V$, and power requirements $p : E \mapsto \mathbb{R}^+$, find a power vector $(P_1, P_2, \ldots, P_n) \in \mathbb{R}^n$ of minimum component sum such that there exists a path from s to each $t \in V \setminus \{s\}$ in $G^P = (V, A^P)$, where the arc set $A^P = \{(i,j) \in A : p_{ij} \leq P_i\}$.*

For two distinct nodes u and v in G, the optimal solutions to MEB on the input (G,u,p) and (G,v,p) may differ. An optimal solution differs for different transmission sources. Hence, the nodes must by endowed with a mechanism that allows them to adjust their power levels for a corresponding transmission session. This places demands on the computational capacity and technological solution of the nodes. The advantage of this concept lays in the optimality of each broadcast session.

The directed graph induced by power assignments as defined in Problem 1 is not necessarily an arborescence. However, MEB could be stated slightly differently:

Problem 2. *Given a directed graph $G = (V,A)$, a source $s \in V$, and power requirements $p : E \mapsto \mathbb{R}^+$, find an arborescence T spanning G rooted at s minimizing*

$$\sum_{i \in V} \max_{j \in N_T^-(i)} p_{ij}.$$

Observation 1. *The optimal objective function value of a given instance is equal in Problem 1 and 2.*

MEB has received a significant attention during the last two decades. The problem has been introduced in [75], where the authors suggest three greedy heuristic methods and an additional "sweep" operation aiming to remove unnecessary transmission:

MST algorithm. A MST is constructed, and subsequently the edges are oriented towards the leaves, starting from the source. The last step is to assign power levels to each node v according to the longest arc outgoing from v.

SPT algorithm. The shortest path tree (SPT) algorithm establishes a minimum-cost path from the source node to every other node. The broadcast tree consists of superpositions of these unicast paths.

BIP algorithm. The Broadcast Incremental Power (BIP) algorithm resembles Prim's algorithm for MST. BIP gradually constructs a broadcast tree as follows:

- Initially, the tree T contains only the source s, and all nodes have zero power assignment.
- While there are some nodes outside of T
  - Find a node v not already in T whose addition to T results in a minimum increase of power assignment of some node u already in T, and add v to T.
  - Increase the power assignment of u so that v is reached.

The sweep operation. Sweep can be applied after a solution T is constructed by one of the methods above. It can be called iteratively to its own output to further improve the solution. Non-leaf nodes in T are inspected one by one, and if for some node v there exists $u \in N^+(v)$ such that power assignment of v is sufficient to reach all nodes in $N^+(u)$, then the transmission of u is eliminated.

Problem complexity and approximability

NP-hardness results for both general and geometric MEB are provided in [72] by reduction from SET COVER and PLANAR 3-SAT, respectively. In the former case, the network topology is represented by a generic graph with arbitrary weights, whereas in the latter, a Euclidean distance is considered. The special case where $\alpha = 1$, i.e., where the edge weights correspond to distances between the endpoints belongs to the complexity class P. An optimal solution is determined trivially by assigning minimum power sufficient to reach all nodes in a single hop, regardless of the nodes' arrangement.

The first analytical study of the three methods proposed in [75] is given in [73], where the authors provide quantitative characterization in terms of approximation ratios. It has been showed that the approximation ratio of MST and BIP lies in the interval $[6, 12]$ and $[13/3, 12]$, respectively. The approximation ratio of SPT is proved to be at least $n/2$. These results were gradually improved in several works. The upper bound of the MST algorithm was improved to 6 in [4], which closes the gap, and thus proves that the approximation ratio of the MST algorithm is 6. This upper bound is valid for BIP as well, due to a lemma in [73], which says that for Euclidean instances of MEB, the objective function value of a solution obtained by BIP is at most the weight of a MST for this instance. The lower bound on the approximation ratio of BIP was strengthened to 4.598 in [8]. This work also devises an implementation of BIP with improved time complexity $\mathcal{O}(|V|^2)$.

Problem variants

As an extension of MEB, the introduction of non-destinations leads to the MINIMUM ENERGY MULTICAST (MEM) problem, introduced along with MEB in [75]. It is easy to see that MEM is NP-hard due to the fact that MEB is a special case where $D = V$.

A variant of MEB, assuming that there are k adjustable power levels $w_{i,1}, w_{i,2}, \ldots w_{i,k}$ at each node i is studied in [45]. This problem remains NP-hard which is proved by reducing 3-CNF-SAT to it. In addition, [45] contains an approximation algorithm for this version of MEB with a performance guarantee $\mathcal{O}(n^{\varepsilon})$ and time complexity $\mathcal{O}((k+1)^{\frac{1}{\varepsilon}} n^{\frac{3}{\varepsilon}})$ for $\varepsilon \in (0, 1]$. Another restriction considered in [45] is that each node has the same adjustable power levels, i.e., $w_{i,l} = w_{j,l}$ for $1 \leq l \leq k$ and $1 \leq i, j \leq n$. For this problem, another approximation algorithm, which delivers a solution within $\mathcal{O}(\log^3 n)$ times the optimum is devised.

Probabilistic MEB, where a node reliability is considered, has also been investigated. The probabilistic settings aims to reflect the fact that the wireless devices may be subject to temporary or permanent failure. Each node i is thus assigned a value $q_i \in [0, 1]$ representing the probability that i does not fail during the whole operating time. The value q_i depends on the characteristic of the device represented by i as well

as its surroundings. For example, in military settings, the nodes close to an enemy have smaller q_i than those in a safer distance. A given minimum reliability level $\sigma \in [0,1]$ of each path from the source node s to all other nodes has to be achieved. The probability of a multi-hop transmission along a path from i to j being available is equal to the product of the probabilities q_k associated with the nodes involved in the path. Three ILP formulations of probabilistic MEB together with suitable solving methods are developed in [53]. Another study of ILP formulations and associated valid inequalities is pursued by [7].

A restriction of node locations can be imposed. One example is a random grid network, where nodes are chosen independently and randomly from points of a $\sqrt{n} \times \sqrt{n}$ square grid in the plane. The probability distribution of existence of a node can be non-uniform. A lower bound on the optimal objective function value is proved in [10] together with a near optimal construction method. The 6-approximation of MST turns out to be too pessimistic for instances with restricted node locations which represent the real-life instances more closely. The authors of [31] argue that the approximation ratio of the MST algorithm can be considered close to 4 for practical instances.

ILP models

Integer linear programs for MEB are studied in various works. Three ILP formulations are proposed in [22]. The first formulation assumes exactly one transmission from the source s, and at most one transmission from each node in $V \setminus \{s\}$. This formulation operates with order of transmissions which helps to establish connectivity of the resulting solution. Power assignments are straightforwardly modeled by one continuous variable for each node, leading to an objective function minimizing their sum.

The same objective function is used in the second formulation. In contrast, the second formulation allows at least one transmission from the source, and an arbitrary number of transmissions from the other nodes. The necessary subtour elimination is achieved by additional "sequencing variables", proposed for the TRAVELING SALESMAN PROBLEM.

The last formulation is built upon a single-commodity network flow model, and its interpretation follows from the second formulation.

A similar formulation is proposed in [80], where the continuous power variables for each node are replaced by binary variables y_{ij} for each pair (i,j) of nodes, attaining the value 1 if and only if the power of node i equals p_{ij}. Connectivity is achieved by multi-commodity flow variables and associated flow conservation constraints.

The authors of [37] provide a formal proof that using binary power variables instead of continuous ones leads to a stronger formulation, which can be further strengthened by introducing the multi-commodity flow variables.

Yet another network flow model formulation along with efficient solution techniques utilizes a fast solution of a combinatorial relaxation of the model [51].

Planarity of an optimal solution

Alternatively to Problem 1 and 2, a MEB instance can be specified by $G = (V,E)$, $s \in$V, and coordinates $[x_v, y_v]$ for each node $v \in V$. These coordinates implicitly imply distance d_{ij} between every two nodes i and j.

We address the question whether the embedding of an optimal solution to a Euclidean MEB instances is planar. To the best of our knowledge, the available literature does not provide results on this topic. The unpublished findings presented below are based on the assumption that a feasible solution to MEB is a tree as stated by Prob. 2.

By inspecting a large number of instances and their optimal solution, the planarity of the optimum cannot be guaranteed in general, even though in the majority of cases, the optimum is indeed planar.

Proposition 5. *Consider a Euclidean instance of MEB with a subgraph on four nodes that form a simple (not self-intersecting) quadrilateral ABCD. In the following cases, there exists an optimal solution containing at most one of the arcs (A,C) and (B,D) (or their reverse):*

1. *ABCD is a rectangle.*
2. *Three internal angles in ABCD are obtuse.*
3. *Two adjacent internal angles in ABCD are obtuse, and the other two are acute.*

Proof. Consider an instance of MEB which contains, besides other nodes, four nodes that form a simple quadrilateral *ABCD* with internal angles α, β, γ, and δ, and the source node *S* as depicted in Fig. 2.3. It follows from the law of cosine that the longest

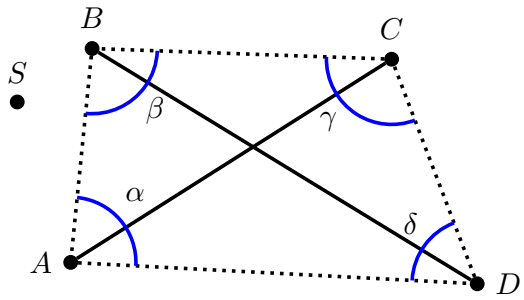


Figure 2.3: Four nodes forming a convex quadrilateral

side of an obtuse triangle is the one opposite to the obtuse angle. The proof analyzes possible solutions in which both diagonals (A,C) and (B,D) (or their reverse) are present. Existence of an alternative solution without crossing and with no greater objective function value is then demonstrated in each case. Let $T = (V, A_T)$ be a solution to the given instance, and let $P_{X,Y}$ be the path from X to Y in T.

1. Assume without loss of generality that $B,C,D \notin P_{S,A}$ and $(A,C),(B,D) \in A_T$. By the law of cosine and the wireless advantage, because $(A,C) \in A_T$, both B and D are covered (receive the signal). The arc (B,D) is thus not needed in T.

2. Assume without loss of generality that δ is the acute angle.

- Let $A,C,D \notin P_{S,B}$ and $(B,D) \in A_T$. Both A and C are covered by (B,D), and so (A,C) as well as (C,A) are thus superfluous in the solution. An analogous argument applies when $A,B,C \notin P_{S,D}$ and $(D,B) \in A_T$.
- Let $B,C,D \notin P_{S,A}$ and $(A,C),(B,D) \in A_T$. B is covered by (A,C). If (A,C) is replaced by (A,B), (B,D) covers C without creating a crossing.
- Let $B,C,D \notin P_{S,A}$ and $(A,C),(D,B) \in A_T$. B is covered by (A,C), (D,B) is thus not needed.

3. Assume without loss of generality that β and γ are obtuse.

- Let $A,C,D \notin P_{S,B}$, $d_{BD} < d_{AC}$, and $(B,D) \in A_T$.
  - If $(A,C) \in A_T$, (B,D) covers C, and so (A,C) can be eliminated.
  - If $(C,A) \in A_T$, D is covered as $d_{CD} < d_{BD} < d_{AC}$. (B,D) can either be removed or replaced with (B,C), if so is needed.
- If $A,C,D \notin P_{S,B}$, $d_{BD} > d_{AC}$ and $(B,D) \in A_T$, it is clear that $d_{BD} > d_{AB}$. Therefore, (B,D) covers both A and C, implying that neither (A,C) nor (C,A) is needed.
- If $B,C,D \notin P_{S,A}$ and $(A,C) \in A_T$.
  - $(D,B) \in A_T$ is not needed, because B is already covered by (A,C).
  - If $(B,D) \in A_T$,(A,C) does not cover D, so (B,D) is needed. By replacing (A,C) with (A,B), the coverage of all nodes is achieved.
- The cases where $A,B,C \notin P_{S,D}$ and $A,B,D \notin P_{S,C}$ are resolved analogously.

□

Remark: If the conditions from Prop. 5 hold for all subsets of four nodes in an instance, then there exists a planar optimal solution. ILP models can be extended by optimality cuts following from the subsets of four nodes satisfying the conditions.

Fig. 2.4 demonstrates an instance in which the conditions form Prop. 5 are not met. In this instance, the quadrilateral $ABCD$ has three acute angles, and coordinates $A = [75,63]$, $B = [50,77]$, $C = [49,63]$, $D = [50,50]$, $X = [45,22]$ and $Y = [47,55]$. The blue edge weights correspond to the power requirements between two nodes. For a convenient display, lengths of edges in the figure are not proportional.

2.2.3 The Range Assignment Problem

In the RANGE ASSIGNMENT PROBLEM (RAP), the task is to minimize the total power consumed under the constraint that adequate power is provided to the nodes to ensure a strong connectivity of the graph. Formally, the problem is stated as follows:

Problem 3. *Given a directed graph $G = (V,A)$ and power requirements $p : E \mapsto \mathbb{R}^+$, find a power vector $(P_1,P_2,\ldots,P_n) \in \mathbb{R}^n$ of minimum sum such that the induced graph $G^P = (V,A^P)$ is strongly connected, where $A^P = \{(i,j) \in A : p_{ij} \leq P_i\}$.*

The requirement on strong connectivity ensures that broadcasting initiated by any node can take place. An obvious advantage of RAP over MEB is that a single transmission graph is constructed in RAP regardless of the source node. However, this can lead

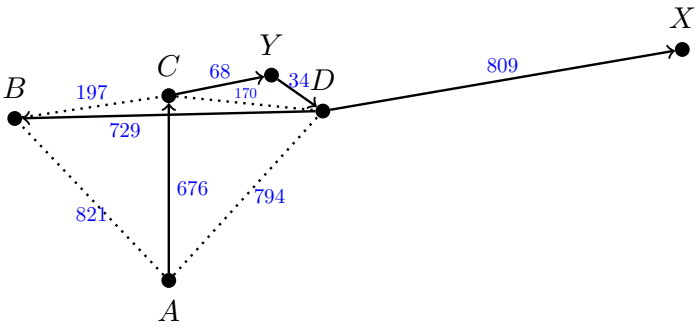


Figure 2.4: A unique optimal solution with crossing to a Euclidean instance of MEB with source in node A.

to solutions that are optimal with respect to RAP, but for certain sources initiating the transmission, the solution may be too expensive. Sect. 2.2.4 describes a problem with a more complicated objective function that combines advantages of both RAP and MEB.

Problem complexity and approximability

In [43], it is shown that for instances consisting of collinear points, there exists an algorithm that solves the problem to optimality in $\mathcal{O}(n^4)$. The authors further prove by reduction from VERTEX COVER that RAP in 3-dimensional Euclidean space is NP-hard for any value of α, and that there exists a $\mathcal{O}(n^2)$ time 2-approximation algorithm based on finding MST. The results are strengthened in [17], where it is proved that RAP is NP-hard for instances in a 2-dimensional Euclidean space for any α, and that for 3-dimensional space, the problem is APX-complete, thus does not admit a PTAS unless P=NP. Let us recall that unlike RAP, for $\alpha = 1$, when the power requirements equal the distances, MEB is solvable in polynomial time. An approximation algorithm improving the performance guarantee approaching 1.69 is presented in [11]. For the case $\alpha = 1$, there exists an approximation algorithm with performance guarantee 1.5 [5]. A recent work [12] achieves an exact $\mathcal{O}(n^2)$ algorithm for instances with nodes arranged in 1-dimensional line.

Problem variants

In an extension of RAP, a further constraint requires that diameter of the transmission graph has to be at most some constant value h. On a family of instances limiting the proximity of two nodes, this variant of RAP is in APX [18]. The 1-dimensional problem with restricted diameter (number of hops) has been studied in [16], where the authors introduce an exact algorithm that runs in $\mathcal{O}(hn^2)$.

Another modification imposes symmetric communication links. This version is often found in the literature under the abbreviation SRAP. In a generalized version, weakly symmetric RAP (WSRAP), the symmetry requirement applies only to a predefined subset of edges. Other edges which are not essential for connectivity are allowed to be unidirectional. The motivation for studying WSRAP stems from the observation that what is really important in the design of WANETs and WSNs is the

existence of a connected backbone of symmetric edges [61]. Imposing symmetry does not change the complexity of the problem, which remains NP-hard in networks with nodes arranged in two and three dimensions [9].

The abbreviation SRAP is sometimes used for the Steiner RAP, where there is a predefined subset $D \subseteq V$ of destination nodes that are required to be included in a solution. The remaining nodes, referred to as *Steiner nodes*, take part in the solution only if their presence reduces the objective function value.

ILP models

An ILP model with an exponential number of constraints, along with a cutting plane method solving it, is presented in [3]. A model based on network flows with a polynomial number of constraints is introduced in [23]. Although this work concerns SRAP with directional antennas, their model can be adapted to the more common version with omnidirectional antennas. For modelling the power assignments, this model uses continuous power variables. By replacing them with binary ones and a corresponding adjustments in the model [37, 52], it is possible to achieve a stronger formulation. An even stronger model uses multi-commodity network flow variables [37]. These strength results are analogous to those regarding MEB. Yet a stronger formulation is obtained by using a multi-tree model [37]. The authors employ binary variables indicating whether or not an arc (i, j) is in the arborescence rooted at $t \in D$. This model is thus applicable to the Steiner RAP.

2.2.4 Shared Broadcast Trees

The idea of broadcasting using a single broadcast tree was first investigated in [57]. The Shared Broadcast Tree (SBT) problem in the form presented here was pursued in [81], where the authors develop an ILP model along with a suitable solving method.

A feasible solution to the SBT problem is any (undirected) spanning tree. This concept is motivated by the intention of incorporating the frequency of use of each node for different broadcast sessions. Every node can transmit a signal, but some are actively transmitting more often than others. We assume that a node acts as a source with a uniform probability. A leaf l transmits a signal only when l is the source, whereas other nodes transmit more often as they also relay signals initiated in other nodes.

Observe that a forwarding node does not have to send a signal back to the node from which it is received. If the signal is received in a node i from its most distant neighbor node i_1, it has to be forwarded to nodes that have not received it yet. That is ensured by relaying the signal along the link to the second most distant node i_2. Due to the wireless advantage, all other neighbor nodes closer than i_2 receive it as well. Conversely, if the signal is received by i from a neighbor node different from its most distant neighbor i_1, node i has to forward it to i_1. It is therefore evident that i does not always have to transmit with the power corresponding to its most distant neighbour. It transmits either with power level p_{ii_2}, if the signal comes from the most distant node i_1, or with power level p_{ii_1}, otherwise. The situation is depicted in Fig. 2.5.

Consider an edge $\{i, j\}$ in a solution T to SBT. Let $T_{i/j}$ denote the subtree of T consisting of nodes from which the path to j contains (i, j). A node i in T contributes

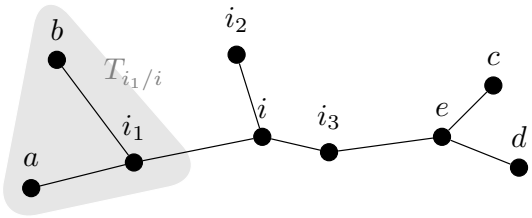


Figure 2.5: Example instance explaining power levels necessary for transmitting a signal from different sources

to the objective function by the expression $c_T(i)$ as follows:

$$c_T(i) = |T_{i_1/i}|p_{ii_2} + |T \setminus T_{i_1/i}|p_{ii_1}. \tag{2.1}$$

We can also regard $c_T(i)$ as a convex combination of i's power levels, with weights corresponding to the number of nodes whose signal is transmitted using the corresponding power level. The objective function is to minimize overall nodes' contributions, that is,

$$c(T) = \sum_{i \in V} c_T(i). \tag{2.2}$$

The SBT problem is thus defined as follows:

Problem 4. *Find a tree $T \subseteq G$ spanning D minimizing $c(T)$.*

Remark: The requirement that a solution must be a tree is necessary due to the nature of the objective function.

Like in MEB, an optimal solution to SBT does not guarantee a planarity of the resulting tree, although a vast majority of its instances have optimal trees with a planar embedding. As an example of the converse, we state the instance on five nodes depicted in Fig 2.6.

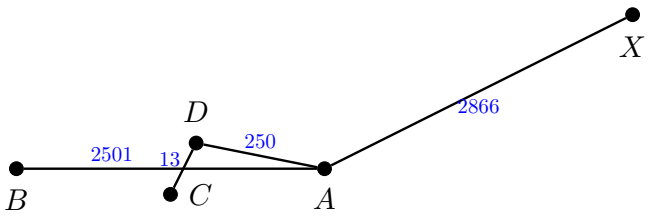


Figure 2.6: A unique optimal solution with crossing to an instance of SBT with objective function value 19904. If the edge $\{DC\}$ is forbidden, the optimal solution becomes a star graph with objective function value 19917. The nodes have coordinates $A = [50,30]$, $B = [49,80]$, $C = [51,48]$, $D = [48,46]$ and $X = [5,1]$.

The introduction of Steiner nodes that are not required to be a part of the resulting tree leads to an extension of SBT referred to as the SHARED MULTICAST TREE (SMT) problem. Some preliminary results are published in Paper I [38], which contains an ILP

model and heuristic methods with local search improvements. Further investigation of the ILP model and a more extensive experimental evaluation are given in Paper III.

In SMT, the objective function has to be adjusted accordingly. The contribution of a single node must reflect the fact that non-destination nodes do not initiate any transmission. Let us define the function $\mu : G \mapsto \mathbb{N}_0^+$ that returns the number of destinations in a given graph G. The contribution of one node in a solution T to the resulting objective function given by Eq. (2.1) is in SMT changed to

$$c_T(i) = \mu(T_{i_1/i})p_{ii_2} + \mu(T \setminus T_{i_1/i})p_{ii_1}. \tag{2.3}$$

This objective function takes into account energy consumption by transmission only. Available literature on multicast versions of MEB and RAP applies this energy model too, however, it is also possible to incorporate other, less significant energy consumption sources as indicated by [34]. As each node that participates in the solution poses an additional energy outlay ε, the objective function (2.2) would then become

$$c(T) = \varepsilon|V_T| + \sum_{i \in V} c_T(i), \tag{2.4}$$

if the energy expenses of each node are homogeneous.

Chapter 3

Minimum Broadcast Time

Broadcasting is the information dissemination process in a communication network. The MINIMUM BROADCAST TIME (MBT) problem is identified by a set of communication devices (nodes), among which some selected ones act as originators of a signal. The number of originators is at least one, and we refer to them as *sources*. The task is to spread a signal from the sources to the remaining nodes along pre-defined communication links in a shortest possible time. In general, communication links are not present between every two nodes. The continuous time is divided into discrete time steps. The communication among the nodes must fulfill the following rules:

1. Each transmission takes place between two adjacent nodes.
2. Each transmission requires one time step.
3. Each node can participate in at most one transmission per time step.

Although this communication protocol is primarily considered as a theoretical model, it appears in various practical applications such as communication among computer processors and telephone networks. Military command, control, communication, computers, intelligence, surveillance and reconnaissance (C4ISR) pose another application for these models [26]. In satellite networks, even though the communication is wireless, signals have to cover large distances, and so the information is sent from one satellite to one of its neighbours at a time. The MBT problem was studied in the context of an existing Chinese BeiDou Global Navigation Satellite System [14].

3.1 Network Model and Definitions

The communication network is determined by a graph $G = (V, E)$ and a subset $S \subseteq V$ of sources.

Definition 16. *The broadcast time $\tau(G,S)$ of $S \subseteq V$ in G is defined as the smallest integer $t \geq 0$ for which there exist a sequence $V_0 \subseteq \cdots \subseteq V_t$ of node sets and a function $\pi : V \setminus S \mapsto V$, satisfying:*

1. *$V_0 = S$ and $V_t = V$,*
2. *for all $v \in V \setminus S, \{v, \pi(v)\} \in E$,*

3. *for all* $k = 1, \ldots, t$ *and all* $v \in V_k$, $\pi(v) \in V_{k-1}$, *and*

4. *for all* $u, v \in V_k \setminus V_{k-1}$, $\pi(u) = \pi(v)$ *only if* $u = v$.

A node is said to be *informed* at a given time if it is a source, or it already has received the signal from some other node. Otherwise, the node is said to be *uninformed*. Consequently, the set of informed nodes is initially exactly the set of sources. Each node set V_i, $0 \leq i \leq t$ consists of nodes informed in time step i. The function $\pi(u)$ determines a parent node which forwarded the signal to node u. With this interpretation, the conditions 1.-4. of Def. 16 can be translated into natural language as

1. Initially, the only informed nodes are the sources, and at the end of the transmission, all nodes are informed.

2. Nodes send signal along communication links defined by the set of edges.

3. In each time step, a node is informed by its parent who was informed in the previous time step.

4. A node can inform only one child node at a time.

Any feasible solution determines a *communication forest* $\mathcal{F}$ with the set of nodes V. Arcs in $\mathcal{F}$ are induced by the communication: if a node u sends a message to v, (u,v) is an arc in $\mathcal{F}$. It is easily verified that the $\mathcal{F}$ contains arborescences rooted at distinct sources. Individual arborescences in $\mathcal{F}$ are referred to as *communication trees*. Let $T_s = (V_{T_s}, A_{T_s})$ denote the communication tree rooted at source $s \in S$, and define the function $\alpha : V \to S$ such that $v \in V_{T_{\alpha(v)}}$ for all $v \in V$. That is, α associates a node v with the source at which the communication tree containing v is rooted. For an arc (u,v) in a given communication tree, the integer $t_{u,v}$ denotes the time step in which node u informs v.

Formally, the optimization MBT problem is defined as follows:

Problem 5. *Given* $G = (V,E)$ *and* $S \subseteq V$, *find* $\tau(G,S)$.

In the literature, authors often consider the decision version, because some of the theoretical results depend on a given deadline.

Problem 6. *Given* $G = (V,E)$, $S \subseteq V$, *and a deadline* $t \in \mathbb{Z}_0^+$, *does* $\tau(G,S) \leq t$ *hold?*

Even though the set of sources in a MBT instance is an arbitrary subset of V, the following concept assumes a single source. In a given graph G, any node can be a source. Different sources in G form MBT instances with generally different broadcast times. The *broadcast center* of a graph is the set of all nodes having the smallest broadcast times, i.e., $\arg\min \{\tau(G, \{v\}) : v \in V\}$.

In c-broadcastting, which is a generalization of regular broadcasting, a signal can be sent to up to c nodes adjacent to an informed node in each time step.

3.2 Computational Complexity

MBT has been thoroughly studied from the perspective of computational complexity and approximability, particularly in the early 90'. Its NP-completeness or belonging to P was proved for various graph classes and values of deadline t.

It has been shown that the problem is NP-complete for arbitrary graphs [32] and few years later, this result was obtained for arbitrary graphs with deadline $t = 4$ and a single source by reduction from 3D MATCHING [68]. The same work also contains an $\mathcal{O}(n)$ algorithm for the determining broadcast time of a tree with a single source. As a by-product, this algorithm determines the broadcast center of the input tree.

These results are improved and extended in [42], where the authors exploit the properties of the NP-complete problem PLANAR 3-SAT. First, they prove that another satisfiability problem, referred to as PLANAR 3,4-SAT, is NP-complete, and subsequently use this property to show that MBT is NP-complete for:

- bipartite planar graphs, deadline $t = 2$, and maximum degree at most 3,
- split graphs with deadline $t = 2$,
- chordal graphs with a single source,
- planar graphs with a single source and maximum degree at most 3,
- bipartite planar graphs with maximum degree at most 3 and a single source,
- grid graphs with deadline $t = 2$ and maximum degree at most 3,
- grid graphs with a single source, and
- complete grid graphs with deadline $t = 2$.

The question whether MBT is NP-complete for split graphs with a single source is stated as open. Another complexity result proving that MBT remains NP-complete for 3-regular planar graphs with constant deadline $t \geq 2$ is given in [50] by reduction from EXACTLY-ONE-IN-THREE-3-SAT.

The complexity results above typically exploit sophisticated reductions. We provide a very simple and straightforward proof of NP-completeness of MBT for deadline at most $t = 3$ on bipartite graphs with maximum degree at most 3. There are many restrictions on SAT that preserve NP-completeness. We concentrate on the variant of 3-SAT with the property that each variable is restricted to appear at most three times, and each literal at most twice. This problem is known as 3-3-SAT, and its NP-completeness proof can be found in [56]. Note that in this particular version of 3-SAT, it can no longer be assumed that each clause consists of exactly 3 literals. Further, the assumption about at most two occurrences of a literal is automatic, because a formula φ that contains a variable x that appears only as a positive or only as a negative literal can trivially be transformed into φ' that does not contain x at all, such that φ is satisfiable if and only if φ' is satisfiable.

We now define the association between 3-3-SAT and the decision version of MBT (Prob. 6). For that purpose, consider the following instance

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge \bar{x}_2 \tag{3.1}$$

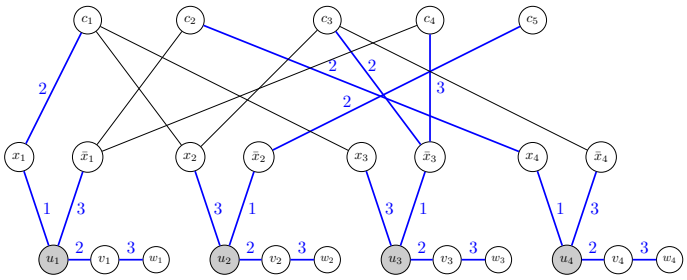


Figure 3.1: Reduction from formula φ (Eq. (3.1)) of 3-3-SAT to an instance of MBT with deadline 3 and maximum node degree 3

of 3-3-SAT with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, reducing to the instance of Prob. 6 in Fig. 3.1. For each variable x_i, we construct a gadget comprising five nodes x_i, $\bar{x}_i$, u_i, v_i, w_i, where u_i is a source, $i = 1, \ldots n$. The first two nodes represent possible literals associated with the variable x_i. The gadget further contains four edges, $\{u_i, x_i\}$, $\{u_i, \bar{x}_i\}$, $\{u_i, v_i\}$, and $\{v_i, w_i\}$. For each clause c_j, there is a node c_j and whenever the clause c_j contains a literal x_i ($\bar{x}_i$), there is an edge $\{c_j, x_i\}$ ($\{c_j, \bar{x}_i\}$).

Proposition 6. *MBT (Prob. 5) is NP-hard.*

Proof. We show that an instance φ of 3-3-SAT is satisfiable if and only if $\tau(G,S) \leq 3$ for the corresponding instance (G,S) of MBT.

Let φ be satisfiable. A source node u_i sends the signal towards the node representing the literal that is evaluated as true in the given truth assignment. In the next two time steps, the signal is further relayed to the nodes representing clause that are satisfied by the literal. As each literal can cause satisfaction of up to two clauses, the corresponding clause nodes receive the signal from the respective literals within the deadline 3. Nodes v_i must receive the signal in the time step 2, in order to reach w_i on time. Thus, the node representing the literal evaluated as false receives the signal in time step 3.

Conversely, let the constructed instance of MBT have broadcast time no more than 3. In a solution T to the instance (highlighted as blue in Fig. 3.1) $\arg\min_{v \in \{x, \bar{x}\}} \{t_{u_x, v}\}$ represents the assignment of truth value to the variable x. The presence of an arc entering a clause node c in T indicates that a truth value of certain variable caused satisfaction of clause c. The auxiliary path (u_x, v_x, w_x) ensures that the clause satisfaction is modeled correctly. As $t_{u_x, v_x} \leq 2$, one of $t_{u_x, x}$ and $t_{u_x, \bar{x}}$ must be 3, and thereby the arc that would incorrectly indicate satisfaction of a clause that is not satisfied by the selected truth assignment would have cost 4, which is not allowed. It is possible to have an arc outgoing from a clause node of cost 3, but the second endpoint is always a node representing some literal y, but the arc (u_y, y) is already a part of T.

The correctness of the truth valuation is ensured by the existence of nodes v_i, so that the node representing literal that is evaluated as false is informed in time step 3, and thereby cannot forward the signal to any clause nodes. □

Proposition 7. *The decision version of MBT (Prob. 6) is NP-complete for bipartite graphs with maximum node degree 3 and deadline 3.*

Proof. In [68], a polynomial algorithm for MBT in trees is devised. As the sequence of vertices $V_0 \subseteq \cdots \subseteq V_t$ and function π in Def. 16 determine a unique forest $\mathcal{F}$ of

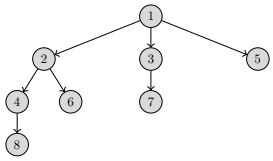


Figure 3.2: A binomial tree with nodes labeled by their positions

arborescences, it can be verified in polynomial time whether the broadcast time of each broadcast tree in $\mathcal{F}$ does not exceed 3, proving that MBT belongs to NP. This concludes together with Prop. 6 that MBT is NP-complete for deadline at most 3.

As each literal appears at most twice in φ, the degree of nodes representing literals is at most 3. Each clause node has degree at most 3 since φ is in 3-CNF. Degrees of nodes u_i, v_i, and w_i do not depend on φ, and have degrees 3, 2, and 1, respectively. Thus, none of the node degrees in the constructed MBT instance exceeds 3.

Nodes of the constructed graph are divided into two disjoint sets $X = \{u_i, w_i, c_j\}$, $i = 1,\ldots,n$, $j = 1,\ldots,m$ and $Y = \{x_i, \bar{x}_i, v_i\}$, $i = 1,\ldots,n$. The graph is bipartite as there are no edges between any two nodes within X or Y. □

3.3 Integer Linear Programming Models

A straightforward ILP formulation of MBT has been proposed in Paper IV together with a suitable solution method. Independently, a similar formulation has recently been published in [24]. In the following, we present an unpublished ILP model for MBT, which exploits the regular structure of binomial trees.

3.3.1 Binomial Tree Model

A *binomial tree* B^k of order k is an ordered tree defined recursively as follows [19]:

- The binomial tree B^0 consists of a single node.
- The binomial tree B^k has a root with k children where the i-th child is the root of a binomial tree of order $k-i$, $i = 1,\ldots,k$.

An example of B^3 is depicted in Fig. 3.2. For a given time step k, the maximum number of informed nodes within k steps is $|S|2^k$. This occurs when the solution of MBT consists of broadcast trees that are binomial.

Observation 2. *If r is the root of B^k, then $\tau(B^k, \{r\}) = k$.*

Let $k \in \mathbb{N}$ and $I = \{1,\ldots,2^k\}$. A directed binomial tree $B^k = (V_{B^k}, A_{B^k})$ with arcs oriented towards the leaves has a regular structure that allows to define a systematic numbering of nodes so that a node number determines unambiguously a *position* in B^k. That is, we need an applicable bijective function $\beta : V_{B^k} \to I$. A suitable bijection β assigns values from I to nodes increasingly with decreasing outgoing degree. In case

of a tie, β takes the smaller value at the node whose parent node is assigned the smaller value. This function is defined recursively as

$$\beta(v) = \begin{cases} 1, \text{ if } v \text{ is the root of } B^k, \\ \beta(u) + 2^{k-deg^+(v)-1}, \text{ if } \pi(v) = u. \end{cases}$$

Nodes in Fig. 3.2 are labeled with their positions according to β.

Definition 17. *For a node $v \in V_{B_k}$ with position $i = \beta(v)$, define the corresponding child position set, $C(i)$, consisting of positions of child nodes of v in B^k. That is,*

$$C(i) = \left\{ \beta(u) : \pi(u) = v, \textit{ i.e., } u \textit{ is a child of } v \textit{ in } B^k \right\}.$$

By the definition of β, it follows that

$$C(i) = \{2^j + i : j = \lceil \log_2 i \rceil, \ldots, k-1\}. \tag{3.2}$$

The idea behind this model is that every broadcasting from a single source follows arcs of a (potentially pruned) binomial tree. The value of $\tau(G,S)$ can be retrieved from the maximum among the positions assigned to nodes. Intuitively, an advantage of this concept is when applied on dense graph instances where feasible solutions should be found quickly. To see this, consider a complete graph K_n. Any assignment of positions $1, 2, \ldots, n$ to nodes is valid and optimal. Moreover, a broadcast time of dense graphs is often equal to the lower logarithmic bound. Near complete graphs have therefore many optimal solutions, which are expected to be discovered easily.

The formulation

Consider a graph $G' = (V', E')$ constructed by adding an auxiliary universal node v_0 to G. The set of nodes and edges is then $V' = V \cup \{v_0\}$ and $E' = E \cup \{\{v_0, v\} : v \in V\}$. So far we have considered binomial trees B^k of an arbitrary order k. For finding a valid assignment of positions to the nodes, we need a binomial tree of order at least $\tau(G,S)$. As $\tau(G,S)$ is unknown, we must determine some suitable upper bound $\bar{t}$ on $\tau(G,S)$.

Observation 3. *A trivial upper bound on $\tau(G,S)$ is $n - |S|$.*

The ILP formulation based on a partition into binomial trees uses variables

$$z_j = \begin{cases} 1, \text{ if } j \leq \tau(G,S), \\ 0, \text{ otherwise,} \end{cases} \qquad y_{is}^v = \begin{cases} 1, \text{ if } \beta(v) = i \text{ and } \alpha(v) = s, \\ 0, \text{ otherwise,} \end{cases}$$

where $v \in V'$, $i \in I = \left\{1, \ldots, 2^{\bar{t}}\right\}$, $s \in S$ and $0 \leq j \leq \bar{t}$. With the definition of G' above, it is straightforward to specify constraints that enforce desired values for y-variables. Whenever $y_{is}^{v_0} = 1$, it indicates that the binomial tree rooted at s is pruned at node with

position i. The formulation based on binomial trees is the following:

$$\min \sum_{j=0}^{\bar{t}} z_j,$$

$$\text{s. t.}$$

$$\sum_{i\in I}\sum_{s\in S} y_{is}^{v} = 1 \qquad v \in V, \tag{3.3a}$$

$$\sum_{v\in V'} y_{is}^{v} = 1 \qquad i \in I, s \in S, \tag{3.3b}$$

$$y_{1s}^{s} = 1 \qquad s \in S, \tag{3.3c}$$

$$y_{is}^{v_0} + y_{is}^{u} + \sum_{v\in V\setminus N(u)} y_{\ell s}^{v} \le 1 \qquad u \in V, i \in I, \ell \in C(i), s \in S, \tag{3.3d}$$

$$y_{is}^{v_0} + y_{\ell s}^{u} + \sum_{v\in V\setminus N(u)} y_{is}^{v} \le 1 \qquad u \in V, i \in I, \ell \in C(i), s \in S, \tag{3.3e}$$

$$\sum_{v\in V} y_{is}^{v} \le z_{\lceil \log i \rceil} \qquad i \in I, s \in S, \tag{3.3f}$$

$$y \in \{0,1\}^{I\times S\times V'}, z \in \{0,1\}^{\bar{t}}. \tag{3.3g}$$

The interpretation of constraints (3.3a) is that every node in the original graph G belongs to exactly one binomial tree. Note that these constraints are quantified only over V and not over V'. In this way it is achieved that v_0 can be regarded as a part of several binomial trees. By (3.3b) is enforced that exactly one node, possibly v_0, is allocated to position i of each binomial tree. By the summation over V' is ensured, that pruned nodes are collectively represented by v_0. Next, (3.3c) enforce that source nodes are always the first nodes in corresponding binomial trees, in accordance with definition (3.3.1) of the function β. Constraints (3.3d) and (3.3e) guarantee that the arcs of binomial trees follow edges in E. In particular, it is enforced by (3.3d) that if u and v are not adjacent in G, then v must not have a position of child of u in any binomial tree. Similarly by (3.3e), if u and v are not adjacent in G, then v must not act as a parent of u in any binomial tree. Without (3.3d) and (3.3e), it could be possible to find a feasible solution, even when no partition of G into pruned binomial trees exists. Finally, the relation (3.3f) between y and z variables follows from Obs. 2. It says that whenever there is a node in a position i, then the delay is at least $\lceil \log i \rceil$.

Consider a subset $U \subseteq V$. Let $N(U) = \{v : \{u,v\} \in E, u \in U\}$, and $\bar{N}(U) = V \setminus N(U)$. Constraints (3.3d) - (3.3e) can then be replaced by stronger

$$\sum_{v\in \bar{N}(u)} y_{\ell s}^{v} \le \sum_{v\in N(\bar{N}(u))} y_{is}^{v} \qquad u \in V, i \in I, \ell \in C(i), s \in S, \tag{3.4a}$$

$$\sum_{v\in \bar{N}(\bar{N}(u))} y_{\ell s}^{v} \le \sum_{v\in N(u)} y_{is}^{v} \qquad u \in V, i \in I, \ell \in C(i), s \in S. \tag{3.4b}$$

It is enforced by (3.4a) that if there is some non-neighbor v of u with a position ℓ in a binomial tree rooted at s, then there must be a neighbor of some non-neighbor of u with position i in the same binomial tree. Similarly, Constraints (3.4b) state that if there is a node v where $\beta(v) = \ell$ in a binomial tree rooted at s in the complement

of neighborhood of all non-neighbors of u, there must also be a neighbor of u with position i in this binomial tree. This reflects the obvious fact that if a tree is pruned at some node, all its descendants must also be excluded from the tree.

Valid inequalities

Let W be a maximal independent set in G. Model (3.3) is strengthened by

$$y_{is}^{v_0} + \sum_{v \in W} (y_{is}^{v} + y_{\ell s}^{v}) \leq 1 \qquad i \in I, \ell \in C(i), s \in S, \tag{3.5}$$

which exploits the fact that no pair of nodes in W is adjacent, and so there must be no two nodes with adjacent β-positions.

We now generalize this idea by using the notion of graph power $G^m = (V, E^m)$ commonly defined as a graph with the same set of nodes as G, and an edge between two nodes in G^m is present if and only if there is a path of length at most m between them in G. For our purposes, we use a slightly modified definition of the edge set

$$E^m = \{\{u, v\} : \text{there exists a path between } u \text{ and } v \text{ in } G \text{ of length } m\}.$$

Definition (3.2) can be generalized to descendants of an arbitrary distance $m = 1, 2, \ldots$ from v in B^k:

$$\begin{aligned} C^1(i) &= C(i), \\ C^{m+1}(i) &= \bigcup_{j \in C^1(i)} C^m(j). \end{aligned} \tag{3.6}$$

Further strengthening of model (3.3) is achieved by introducing valid inequalities

$$y_{is}^{v_0} + \sum_{v \in W_m} (y_{is}^{v} + y_{\ell s}^{v}) \leq 1 \qquad i \in I, \ell \in C^m(i), s \in S, 1 \leq m \leq \Delta_G - 1. \tag{3.7}$$

Clearly, inequality (3.5) is included in (3.7) for $m = 1$. The distance between positions i and $\ell = C^m(i)$ in a binomial tree is m. The maximal independent set W_m contains nodes such that length of any path between any two nodes is different from m, and so there cannot be two nodes in W_m with positions i and ℓ at the same time.

Symmetry removal

Another improvement of this model is achieved by a symmetry removal. If a broadcast tree is identical to a binomial tree, we notice that nodes with positions from $C(i)$, i.e., children of some node v with $\beta(v) = i$, are informed in increasing time steps. For example in B^3, $C(2) = \{4, 6\}$ and the corresponding nodes are informed in time step 2 and 3, respectively. If a position $\ell \in C(i)$ corresponds to a node of a binomial tree that is pruned (if $y_{\ell s}^{v_0} = 1$ for some $s \in S$), all positions $j \in C(i)$ such that $j > \ell$ can also be pruned. Thus, adding

$$y_{js}^{v_0} \leq y_{\ell s}^{v_0} \qquad i \in I, j, \ell \in C(i), j < \ell, s \in S \tag{3.8}$$

to the model reduces the set of feasible solutions, while preserving at least one optimal solution.

Decision version

To determine whether a given deadline k is sufficient for broadcasting in an instance (G,S) of MBT, consider the following ILP model:

$$\max \sum_{v\in V}\sum_{i\in I}\sum_{s\in S} y_{is}^{v},$$

$$\text{s. t.}$$

$$\sum_{i\in I}\sum_{s\in S} y_{is}^{v} \leq 1 \qquad v \in V, \tag{3.9a}$$

$$(3.3b)-(3.3e),$$

$$y \in \{0,1\}^{I\times S\times V'}. \tag{3.9b}$$

This model is a modification of formulation (3.3), and uses the same type of variables. The objective function is to maximize the number of nodes that are assigned a position in binomial trees. The constraints (3.9a) state that each node belongs to at most one binomial tree. In contrast to (3.3a), (3.9a) is an inequality, because the binomial trees do not necessarily form a partition of G, and so not all nodes have to be used. The remaining constraints are taken from formulation (3.3).

The parameter k affects I. If the objective function attains the target value $|V|$, it is obvious that positions $\{1,2,\ldots,2^k\}$ can be assigned to all nodes in G under the given constraints, and therefore $\tau(G,S) \leq k$. If the objective function value is less than $|V|$, then clearly $\tau(G,S) > k$.

An optimal solution to Prob. 5 can be determined by a sequential solution to model (3.9) for varying deadlines k. Assume we are given a lower bound $\underline{t}$ and an upper bound $\bar{t}$ on the minimum broadcast time. Initially, we define the set $I = \{1,\ldots,2^{\underline{t}}\}$ and iteratively solve the model while doubling the set I (increasing $\underline{t}$) until the objective function attains the value $|V|$. That indicates that it is the first iteration in which all nodes are assigned a position, and it can be concluded that the minimum broadcast time is $\log_2|I| = \underline{t}$. If the target value $|V|$ is not met for $k = \bar{t}-1$, the last iteration with $k = \bar{t}$ does not have to be conducted, as it is already obvious that $\tau(G,S) = \bar{t}$.

It is suggested that the sequence of models is solved for an increasing k. Different strategies are also available. For example, a binary search may seem a more natural way, however, running the model for larger values of k takes significantly more time than for smaller values of k. Furthermore, large graphs tend to have their broadcast time closer to the lower bounds.

Discussion

Preliminary experimental results indicate that using binomial tree model for obtaining optimal solutions of MBT instances takes longer time compared to the model presented in Paper IV in virtually all tested instances. Tab. 3.1 shows solution times for two instance sets with number of nodes 50 and 100. Each instance set is further divided by the number of sources, in this case 1 and 2. The last parameter that identifies an instance is the number of edges (1st and 6th column). An instance is identified by values $|V|$, $|S|$ and $|E|$. Each instance is solved by an iterative algorithm described in section 3.3.1 on model (3.9) (columns with heading Y), and an analogous algorithm on model introduced in Paper IV (columns with heading X).

Even this small sample of test instances suggests that the model in Paper IV finds an optimal solution substantially faster than (3.9). However, there is a potential in improving the model by means of techniques such as further symmetry removal and additional valid inequalities following from the concept of unique numbering of nodes in a binomial tree. Also, it can be seen that the more sources, the shorter time the computation takes, which is more apparent for model (3.9).

	$\|V\|$=50					$\|V\|$=100				
	$\|S\|$=1		$\|S\|$=2			$\|S\|$=1		$\|S\|$=2	
$\|E\|$	X	Y	X	Y	$\|E\|$	X	Y	X	Y
276	0.22	7.39	0.18	4.37	1041	0.99	261.71	0.7	109.32
262	0.19	4.76	0.2	2.67	1078	0.78	243.66	0.84	84.45
269	0.2	8.43	0.1	2.62	1013	1.26	326.66	0.78	141.28
194	0.3	9.81	0.13	3.08	798	0.74	441.31	0.39	47.28
192	0.14	2.72	0.15	3.16	781	0.91	254.67	0.52	53.74
188	0.16	4.75	0.09	2.26	726	0.87	299.97	0.78	225.19
63	0.19	8.49	0.17	11.82	125	0.57	904.51	0.54	705.06
100	0.16	7.49	0.08	2.59	200	1.23	1126.11	0.76	298.8

Table 3.1: Comparison of solution time of methods based on model (3.9) and an analogous model in Paper IV

3.4 Related Problems

There are many communication protocols for broadcasting in different practical applications, and their description is out of the scope of this thesis. We therefore describe only the problems closely related to MBT.

3.4.1 Broadcast Graphs

A related problem to MBT is the MINIMUM BROADCAST GRAPH problem. In this settings, broadcast protocol according to Def. 16 and $|S| = 1$ are assumed. A *broadcast graph* is a communication network on n nodes with optimal broadcast time $\lceil \log n \rceil$ regardless of the source. Every complete graph K_n satisfies this property, yet K_n is not minimal. The minimum number of edges in any broadcast graph on n nodes is denoted as $B(n)$. A *Minimum broadcast graph* is a broadcast graph with $B(n)$ edges.

NP-completeness follows from the same result for MBT. A long list of papers deals with determining minimum broadcast graph for different number of nodes. The initial work [30] shows minimum broadcast graphs for $n = 2^k$ and $n \leq 15$. Further investigation [36] leads to a technique for constructing broadcast graphs for many values of n.

In the generalized c-broadcasting, a *c-broadcast graph* has a broadcast time $\lceil \log_{c+1} n \rceil$, and analogously, $B_c(n)$ denotes the minimum number of edges in any c-broadcast graph. A c-broadcast graph with $B_c(n)$ edges is said to be a *minimum c-broadcast graph*. A time-relaxed c-broadcasting [48] allows additional t time steps. Thus, a time-relaxed c-broadcast graph has broadcast time $\lceil \log_{c+1} n \rceil + t$.

Another generalization is the k-fault tolerant broadcast problem [46]. The task is to identify sparse graphs with reliable transmission schemes. The protocols of a k-fault tolerant broadcast graph are predefined in such a way that if any k edges in the protocol fail, the signal still reaches all nodes in the graph.

ILP formulations to construct c-broadcast graphs, k-fault tolerant c-broadcast graphs and minimum c-broadcast graphs are presented in [48].

3.4.2 The Gossiping Problem

The last problem mentioned in this overview is THE GOSSIPING PROBLEM, also known as the TOTAL INFORMATION EXCHANGE. In this problem, each node is initially given a different message that needs to be distributed to all other nodes [13]. For this, in each time step the nodes send each other messages consisting of an arbitrary number of pieces of information. The standard restriction is that in one time step, a node can communicate with only one of its neighbors. One distinguishes 1-way (or half-duplex) mode, where the information can be sent through a link in only one direction in a single time step, and 2-way (or full-duplex) mode, where two nodes may exchange all their information through a link that connects them within a single time step. The most intensively studied efficiency criterion in this theory is the number of time steps needed for disseminating all pieces of information to every node [27].

Chapter 4

Path Finding for Multiple Robots

Before focusing on path finding for multiple robots, let us summarize techniques for simple path finding in graphs, which act as building blocks in problems that consider multiple robots.

4.1 Basic Path Finding

Basic path finding is a well known task in computer science. Given a graph $G = (V, E)$, the objective is to find a path between two selected nodes s and t referred to as the *source* and the *target*, respectively. Graph searching methods such as breadth-first search and depth-first search find a path if given a sufficient amount of time.

The aim is often to find a shortest path between s and t in G with given edge lengths. Bellman-Ford algorithm yields a shortest path from s to all of the other nodes in $\mathcal{O}(|V||E|)$ time.

A common approach used for finding a shortest path is *Dijkstra's algorithm* [28], which guarantees to yield an optimal path in $\mathcal{O}(|E|\log|V|)$ time. Dijkstra's Algorithm visits vertices in the graph starting with s. It then repeatedly inspects the vertex which was not yet visited that lies closest to s. It expands outwards from s until it reaches t. A drawback of this approach is that it may expand too many vertices that later turn out to lie very far from t.

Another possibility is the greedy *best first search* algorithm which can find a path to t faster, but the selected path is not guaranteed to be optimal. Instead of expanding nodes close to s, it selects those close to t. Obviously, which node is exactly the closest is not known, and therefore it uses a heuristic estimate, which guides the way towards t.

*A** algorithm [35] combines the advantages of Dijkstra's algorithm and the best first search algorithm. It guarantees to find an optimal path and also exploits a heuristic estimation of the distance to the target, which helps to avoid expanding unsuitable vertices. The heuristic estimation is particularly useful when there is incomplete information about the graph. For that reason, A* is popular in video game development, where the exact distances cannot be predicted due to a frequent change of the environment, or due to a lack of knowledge of the current situation in the environment.

A* searches for a shortest path from s to t by maintaining a tree of paths originating at s, and extending those paths until its termination criterion is satisfied. Alg. 1 shows a pseudocode of A*. At each iteration, the algorithm determines which of its paths should be extended. The selection is based on the cost of the path so far and a heuristic

estimate of the cost required to extend the path all the way to t. Specifically, A* chooses the path that minimizes

$$f(v) = g(v) + h(v),$$

where v is the node by which the path is extended, $g(v)$ is the cost of the path from s to v, and $h(v)$ is the heuristic estimate of the cost of the cheapest path from v to the target t. A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended.

The performance of A* strongly depends on the selected heuristic function h. The heuristic is *admissible* if $h(v)$ is no larger than the length of the shortest path from v to t. An admissible heuristic is an optimistic guess as it underestimates the length of the shortest path. Thus, $f(v)$ is a lower bound on the path cost via v. The heuristic is said to be *monotonous* or *consistent* if for all $v' \in N(v)$ we have that $h(v) \leq w_{vv'} + h(v')$.

Observation 4. *Every monotonous heuristic is admissible.*

Proof. Let h be monotonous, and let $s = v_1, v_2, \ldots, v_k = t$ be an optimal path. As h is monotonous, we have that $h(v_i) - h(v_{i+1}) \leq w_{v_i v_{i+1}}$. This implies that $h(v_1) \leq \sum_{i=1}^{k-1} w_{v_i v_{i+1}}$. □

Observation 5. *For a monotonous heuristic, f is non-decreasing along any path.*

Proof. Let $v' \in N(v)$, i.e., $g(v') = g(v) + w_{vv'}$. Then, $f(v') = g(v') + h(v') = g(v) + w_{vv'} + h(v') \geq g(v) + h(v) = f(v)$. □

Proposition 8. *If h is a monotonous heuristic, A* finds an optimal path.*

Proof. By Obs. 5, if h is monotonous, the values of $f(v)$ are not decreasing along any path. A* selects for expansion the node v with minimal value of f, that is, among all paths to v there cannot be a shorter path than the one just selected. Thus, the path selected from s to v is optimal, which is inductively extended up to t. □

If the trivial monotonous heuristic $h(v) = 0$ for all $v \in V$ is used, the algorithm becomes Dijkstra's algorithm.

There are two different perspectives of the running time. The first way counts the running time as a function of $|V|$ and $|E|$, which is more common in the context of graph theory. In this case, if the monotonous heuristic is determined in constant time, the complexity of A* equals the complexity of Dijkstra's algorithm. Another way, popular in AI community, is measuring the running time in terms of the depth of the solution and the branching factor of the search space. In this context, graphs are typically very large, and avoiding examination of the entire graph is desirable, in fact, it is one of the major goals of the algorithm. If we examine every node at depth up to d before the target is found, we end up visiting $\mathcal{O}(b^d)$ nodes before termination.

4.2 Path Finding for Multiple Robots

The requirement of finding multiple non-colliding paths significantly increases the complexity of the problem.

Algorithm 1: A* algorithm

Input : $G = (V,E)$, $s,t \in V$, $w : E \mapsto \mathbb{R}^+$
Output: A shortest path from s to t or failure if no such path exists

$C \leftarrow \emptyset$ `// Closed set - expanded nodes, will not be entered again`
$O \leftarrow \{s\}$ `// Open set - nodes to be expanded`
for $v \in V \setminus \{s\}$ **do**
  $g(v) \leftarrow \infty$
  $f(v) \leftarrow \infty$
  $\pi(v) \leftarrow$ null `// Predecessor of node` v
end
$g(s) \leftarrow 0$
$f(s) \leftarrow h(s)$
while $O \neq \emptyset$ **do**
  $v \leftarrow$ select a node from $\arg\min\{f(u) : u \in O\}$
  if $v = t$ **then return** path from s to t reconstructed using π
  $O \leftarrow O \setminus \{v\}$
  $C \leftarrow C \cup \{v\}$
  for $u \in N(v) \setminus C$ **do**
    if $u \notin O$ ***or*** $g(v) + w_{vu} < g(u)$ **then**
      $O \leftarrow O \cup \{u\}$
      $\pi(u) \leftarrow v$
      $g(u) \leftarrow g(v) + w_{vu}$
      $f(u) \leftarrow g(u) + h(u)$
    end
  end
end
return failure

We consider an environment with several identical moving entities referred to as *agents*. Source and target locations are uniquely determined for each agent. The objective is to find a route for every agent from its source to its target while avoiding obstacles in the environment. The environment is modeled as an undirected graph, where the agents are placed in the nodes, and move along the edges from one node to another. In this context, the paths/routes are allowed to contain a node multiple times. Movement of the agents is carried out in discrete time steps, where the relocation of an agent from one node to its neighbor takes exactly 1 time step. Agents must not collide with obstacles and other agents that are also moving along planned routes towards their own targets.

Methods for solving path finding for multiple robots are divided into two main approaches: *centralized (coupled)* and *decentralized (decoupled)*. A centralized approach incorporates a global decision maker that regards all the agents as a single entity, and plans paths for them simultaneously. On the other hand, a decentralized approach can considerably reduce computations by decomposing the problem into several smaller subtasks. Paths are typically computed for each agent individually, ignoring all other agents. The interactions are handled along the way to avoid collisions. This is usually much faster but yields suboptimal solutions and loses completeness [60].

Local repair A* (LRA*) [65] is a decentralized algorithm readily applicable to path finding for multiple robots. Each agent searches for a route to the destination using the A* algorithm, ignoring all other agents except for its current neighbours. The agents then start to follow their routes, until a collision is impending. Whenever an agent is about to move into a position occupied by another agent, it instead replans the route from its current position.

Problems dealing with navigating a group of mobile robots can be formulated as multi-agent path planning. However, the primary motivations for the problem are tasks of moving certain entities within an environment with obstacles. The constraint of limited free space represents a key aspect that makes the problem non-trivial. Examples of applications include, but are not limited to: parcel delivery in office building by multiple mobile robots [33], navigation of robots within an industrial warehouse environment [29], and control and management of a fleet of autonomous mobile robots for transshipment tasks in harbors, airports and marshalling yards [2]. Shipping container rearranging can also be formulated as path-planning for multiple agents where agents are represented by containers [70].

4.2.1 Mathematical Model and Variants

The problem of path finding for multiple robots is defined by a quadruple $(G, R, \lambda_0, \lambda_+)$, where

1. $G = (V, E)$ is the undirected graph representing the environment,
2. $R = \{r_1, \ldots, r_{|R|}\}$ is the set of robots or agents, $|R| \leq |V|$,
3. $\lambda_0 : R \mapsto V$ is the injective function that assigns each agent its initial node, and
4. $\lambda_+ : R \mapsto V$ denotes the injective function that assigns a target node to each agent.

In general, there is no restriction on the graph, but majority of works in this area considers 4 or 8-connected grid graphs with some missing nodes representing obstacles. There are the following two different models of agents' movement dynamics: *Multi-Agent Path Finding* (MPF) and *Pebble Motion on Graph* (PMG).

Multi-Agent Path Finding An agent can shift from one node to its neighbor on condition that the neighbor is either unoccupied or is being left by some other agent at the same time step. At most one agent is allowed to pass an edge within one time step. That is, agents are not allowed to exchange their positions within one time step. It is however possible that there is not a single unoccupied node, and agents still perform cyclic moves, although most of the studied instances contain at least one unoccupied node. Finally, no two agents can enter the same node at the same time, as there cannot be more than one agent simultaneously at a node. Under this settings, the optimization variant of MPF is shown to be NP-hard [69].

In general, agents are considered as independent isolated units, and one agent is not aware of other agents' path plans. There are variants of MPF where this is not the case (see Sect. 4.2.2).

Pebble Motion on Graphs *Pebble motion on graphs* [44] can be regarded as a restricted variant of MPF. The difference lies in the rules for movement. While MPF enables entering a vertex that is simultaneously being left by another agent, such transfer is not permissible in pebble motion. As an illustration we mention the 15-puzzle, also known as Lloyd's 15 [6]. Its generalized $N \times N$ version has been proved to be NP-hard in [21]. The situation becomes much easier when the optimality of the number of moves is not required, i.e., when the problem is to find any sequence of movements so that the target locations are reached. In this case the problem belongs to P [44], and so does MPF when the optimality is abandoned, as every solution to PMG is a solution to MPF. As all results relevant to this chapter consider the movement rules of MPF, PMG is not discussed any further.

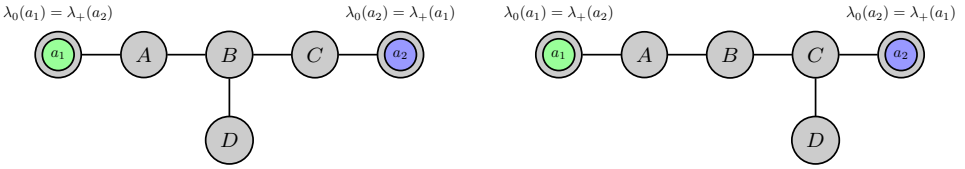
There are three most common optimization criteria [79] in problems belonging to path finding for multiple robots to be pursued:

- Minimum total arrival time - total number of time steps that the agents need before arriving in their targets.
- Minimum makespan - the number of time steps needed by the latest arriving agent.
- Minimum total distance - total number of moves performed by the agents.

In a corresponding decision problem, we are given a parameter k and ask whether it is possible that agents reach their targets so that the objective function does not exceed k.

4.2.2 Cooperative Path-Finding

Cooperative Path-finding (CPF) [65] is a special case of MPF where each agent is assumed to have a full knowledge of all other agents and their planned routes. Precisely speaking, a solution algorithm can take into account paths planned for agents that were



(a) An instance for which CA* fails to find a solution

(b) An instance for which CA* finds an optimal solution

Figure 4.1: Demonstration of incompleteness of CA*

processed earlier, and adjust paths searched later according to them. The concept of CPF is most relevant to the contribution of this thesis, as all the studied variants assume knowledge to at least some agents' path planning.

CA*

A decoupled algorithm for solving CPF, known as cooperative A* (CA*), has been introduced in [65] together with several improvements reducing its complexity. Its basic version works with a *spatial graph* G' constructed from the initial graph $G = (V,E)$. The spatial graph can be regarded as k copies (layers) $G_0, G_1, \ldots, G_k$ of G, which are arranged parallel with each other. These layers represent the time dimension of the agents' movement. All edges within one layer are removed, because a movement along them would represent an agent's change of position in zero time, which is prohibited by the rules for movement. Consider nodes v_i and v_{i+1} in two consecutive layers of G', both corresponding to node v in the original graph. Further, let $v_{i+1}^1, \ldots, v_{i+1}^{|N(v)|}$ be the nodes in the $(i+1)$-th layer corresponding to neighbors of v in the original graph. The spatial graph contains edge $\{v_i, v_{i+1}\}$, which represents an agent staying at its node from time step i to $i+1$, and edges $\left\{v_i, v_{i+1}^1\right\}, \ldots, \left\{v_i, v_{i+1}^{|N(v)|}\right\}$ representing an agent's movement to the adjacent nodes. Edges in G' are always directed from the lower layer to the higher layer, as it is forbidden to move "back in time".

Agents are initially placed in the first layer of G', and if a node t is a target of some agent a in the original graph, nodes corresponding to t in each layer become targets of a in G', and a must reach one of them. This models the fact that an agent can reach its target at any time step. The algorithm processes agents one by one, and always searches a path in G' for a currently considered agent using A*. Once a path is found in G', its nodes in G' are *reserved*, and agents processed afterwards must avoid the reserved nodes. In this way it is ensured that no collisions occur during the agents' advancement.

CA* is neither complete, nor optimal. It is however popular because it does not pose any specific requirement on the instance and also because of its simplicity. An example of an instance for which CA* is unable to find a solution is depicted in Fig. 4.1a. The instance consists of a path on five nodes $\lambda_0(a_1), A, B, C, \lambda_0(a_2)$, and a node D adjacent to the node B in the middle of the path. Agents a_1 and a_2 are placed at the two endpoints of the path, and have targets $\lambda_+(a_1)$ and $\lambda_+(a_2)$ at the opposite endpoints, respectively.

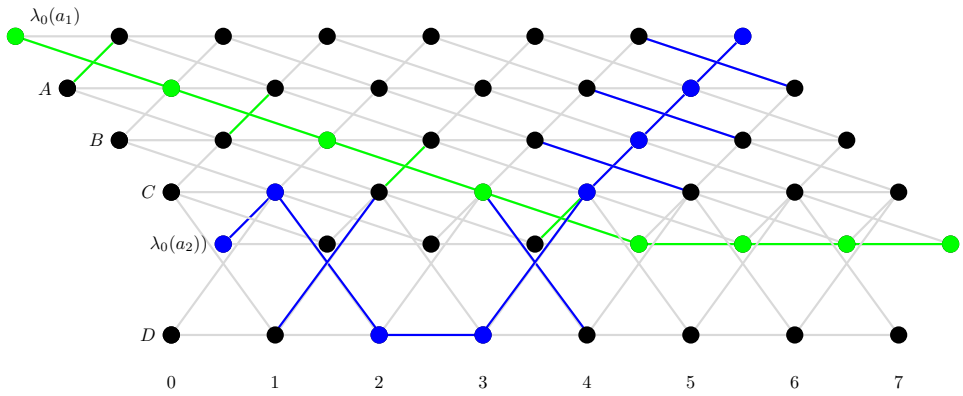


Figure 4.2: Spatial graph corresponding to the instance in Fig. 4.1b

This instance has a solution in which one agent uses the node D outside of the path in order to allow the other agent to pass towards its target. CA* finds a path in the spatial graph that corresponds to the shortest path from a_1's initial position $\lambda_0(a_1)$ to $\lambda_+(a_1)$. When a_2 is subsequently processed, no path to $\lambda_+(a_2)$ in G' that avoids reserved nodes and edges exists. Fig. 4.1b depicts a slightly modified situation in which CA* is able to find the optimal solution. The plan of a_1 is again to go through the shortest path towards $\lambda_+(a_1)$, i.e., $\lambda_0(a_1), A, B, C, \lambda_+(a_1)$. When the path for a_2 is calculated in G', the only option to avoid reserved nodes is to make a step aside in the second step, let a_1 pass, and then continue towards $\lambda_+(a_2)$. Thus, a_2 passes the sequence of nodes $\lambda_0(a_2), C, D, D, C, B, A, \lambda_+(a_2)$. The spatial graph corresponding to the instance shown in Fig. 4.1b, in which agent a_1 is processed first, is displayed in Fig. 4.2. Green and blue nodes represent the positions occupied by agent a_1 and agent a_2, respectively, in some time step. Coloured edges mark those edges along which an agent moves, and cannot be passed by another agent. Therefore, both directions are always marked, as no two agents are allowed to exchange their position within a single time step. The quality of a solution often depends on the order in which agents are processed. Note that in some orderings of the agents, a solution may not exist, while there can be a solution for another ordering. This is also the case of the instance in Fig 4.1b, because if a path for a_2 was planned first, a solution would again not be found.

OD+ID

An example of a coupled algorithm is to apply A* on the following state space: Each state is encoded as an $|R|$-tuple representing all agents' positions. There is an edge between two states if and only if the transition between the two states represents a valid move. With this representation, A* finds an optimal solution, but the obvious drawback of this approach is its time and space complexity. The size of the state space is $\mathcal{O}(|V|^{|R|})$ with the branching factor $\mathcal{O}(d_{max}^{|R|})$ where d_{max} is the maximum degree in G.

An improvement of this naive approach is achieved by *operator decomposition* (OD)[62]. In this approach, a fixed ordering of agents is assumed, and a move is assigned to each agent in succession. Thus, a transition from one state to another cor-

responds to moving only one agent while other agents' positions remain unchanged. Under this representation, every $|R|$-th state corresponds to a full move of the group of agents, and is known as *standard state*. The other states are referred to as *intermediate states*. Standard and intermediate states are conceptually different, they are nevertheless treated equivalently by A*.

Although operator decomposition allows the algorithm to avoid considering a substantial portion of the search space, the resulting algorithm is still exponential in the number of agents [62]. Independence detection (ID) aims to plan paths for agents as independently as possible. The idea is to plan paths for agents separately whenever a cooperation is not needed as they would not collide if they moved along their shortest paths. When OD is combined with ID, agents are divided into groups which are then planned separately using OD. Once a plan for all groups is known, the movement is simulated, and if the plans for two or more groups contain conflicts, the affected groups are merged together, and a new plan is searched for these larger groups.

Other notable algorithms

Besides the methods described above, there are many algorithms of various properties designed for CPF in recent decades, and the research in this area still continues.

An alternative method developed in [59] achieves an implicit cooperation among agents by using so called direction maps.

In [74], a decoupled algorithm *flow annotated replanning* (FAR) for grid graphs is presented. FAR trades the completeness for an improved efficiency, and uses an abstraction of the grid graph into a flow-annotated grid graph. A* is then applied separately for each agent in this abstracted environment.

Another CPF algorithm that claims completeness is *Push and Swap* [47]. However, in [25] were identified instances for which this algorithm fails to obtain a solution. Consequently, algorithm *Push and Rotate* is then presented as an adaptation of the Push and Swap technique. By fixing the Push and Swap's shortcomings, the authors developed an algorithm that is complete for the class of instances that contain at least two unoccupied nodes.

An alternative approach for arbitrary graphs introduces a two-level Increasing Cost Tree Search (ICTS) algorithm [64], consequently compared with A* based techniques. Another algorithm capable of solving a wider range of instances than ICST is Conflict-based Search (CBS) [64]. Low-level searches in CBS are performed as single-agent path finding.

4.3 Scenarios with Adversaries

Consider a problem of MPF where agents are divided into two (or more) adversarial teams competing for achieving some goal. The teams move in turns so that the i-th team moves in every i-th time step, while agents of other teams act as obstacles.

The division into adversarial team is a generalization of MPF that increases the problem's complexity. The problem becomes a multi-player game with full information, and so various search space algorithms developed for these types of games are applicable for adversarial versions of MPF. Examples of these algorithms include alpha-

beta algorithm widely applied on computer chess, and Monte Carlo tree search which was recently successfully employed in the game of Go [66]. These algorithms are often combined with methods of machine learning, such as neural networks and genetic algorithms.

In problems with adversarial elements, we consider one specific team of agents, and try to solve the decision problem whether there exists a winning strategy for the selected team.

4.3.1 Adversarial Cooperative Path Finding

In ADVERSARIAL COOPERATIVE PATH FINDING (ACPF), all the agents are defined by their initial and target nodes in a graph, and the task of one team is to reach all the targets by corresponding agents before the opponent's agents reach their positions. The teams are typically assumed to be of the same number of agents, although it is not required in general. Similarly, the placement of targets of the teams does not have to be symmetric, and so the starting conditions are not necessarily "fair" for the teams.

Besides the points 1.-4. in Sect 4.2.1, the formal definition of ACPF consists of the following elements:

5. $\mathcal{T} = \{T_1, T_2, \ldots, T_{|\mathcal{T}|}\}$, $|\mathcal{T}| \leq |R|$ contains the set of disjoint teams. Each agent belongs to exactly one team. Two teams are considered in a typical instance.
6. $t^* \in \{1, 2, \ldots, |\mathcal{T}|\}$ denotes the index of a team for which the strategy is searched, and for which the answer whether there exists a winning strategy is of interest.

It is also assumed that there is a mechanism that determines the next move of a team different from T_{t^*} when given the sequence of all agents' previous placements. This mechanism is unknown, and is expected to act as an oracle that is able to determine the best possible move for an adversarial team in each time step. The decision version of ACPF is formulated as

Problem 7. *Given an instance $(G, R, \lambda_0, \lambda_+, \mathcal{T}, t^*)$ of ACPF, does there exist a movement of team T_{t^*} at every time step in which the team is to move, so that every agent $a \in T_{t^*}$ reaches its target location $\lambda_+(a)$, and so that no other team reaches its desired locations earlier?*

The problem was proved to be PSPACE-hard for two teams in [39] by a polynomial reduction from TRUE QUANTIFIED BOOLEAN FORMULA (TQBF). It is also known that the problem is EXPTIME, but whether or not it belongs to PSPACE remains an open question.

In experimental settings, a time limit during which the agents move is imposed. When this limit is up, the winner is the team whose agents captured the highest number of target nodes.

4.3.2 Area Protection Problem

Another concept of adversarial MPF is the AREA PROTECTION PROBLEM (APP). Unlike ACPF, where the goals of all teams of agents is to reach their targets, the adversarial teams in APP have different objectives. The first team of *attackers* consists of agents

whose goal is to capture pre-defined targets in the area protected by the second team of *defenders*. There is a one-to-one mapping between attackers and target nodes. The opponent team of defenders aims to prevent the attackers from reaching their targets by occupying selected locations.

The common feature in all MPF problems is that once a location is occupied by an agent, it cannot be entered by another agent unless it is first vacated by the agent which occupies it (agents cannot push away each other). This property is a key tool for the team of defenders. Formally, the problem is defined by points 1.-3. in Sect. 4.2.1, and

5. $D \subseteq R$ and $A \subseteq R$ with $D \cap A = \emptyset$ and $D \cup A = R$, that is, an agent is either an attacker or a defender,
6. $\lambda_+^A : A \mapsto V$, an injective function that assigns a target node to each attacker.

Defenders do not have any pre-defined targets, as their objective is to prevent attackers from reaching their targets. The decision version of APP is formulated as

Problem 8. *Given an instance* $(G, R, \lambda_0, A, D, \lambda_+^A)$*, is it possible to navigate the defenders in a way that no attacker reaches its target?*

Paper V [40] contains a PSPACE-hardness proof of APP.

Like in problems without the adversarial element, there are several possible objective functions of APP:

- Maximize the number of targets not captured by attackers.
- Maximize the sum of distances between attackers and their targets.
- Minimize the time spent by attackers at captured targets.

The available literature focuses on the first objective.

Target allocation

Finding a solution to APP can be viewed as two separate subproblems. As there are no predefined targets for defenders, the initial step is to identify nodes which are important for attackers on paths towards their targets. This subproblem is called *target allocation*. The nodes that are determined as important then become targets of defenders. The next subproblem is then to navigate defenders towards the targets allocated to them in the previous step. A typical candidates for defenders' targets are obviously attackers' targets, but it is not necessarily the smartest option. Particularly in instances where defenders are highly outnumbered by attackers, it is more reasonable to capture nodes lying in bottlenecks through which attackers must pass. In this way, even a small group of defenders may be able to protect a large number of attackers' targets.

4.3.3 Area Protection Problem with Communication Maintenance

An extension of APP by a requirement of communication maintenance (APPC) has been introduced in Paper VI [41]. The environment is assumed to be a 4-connected grid graph with possible missing nodes (obstacles), which allows to define the connectivity constraints.

A Graph G representing the environment is embedded in a plane so that all edges have length 1 unit and each node v has coordinates $C_v = (x_v, y_v)$. The physical location l_v represented by v is the unit square centered at C_v. Further, the set B consists of square areas representing obstacles. The visibility range r is the maximum distance between two locations, such that if there are agents placed at them, they can communicate with each other. A communication is impossible between locations l_u and l_v if the line segment $[C_u, C_v]$ intersects any obstacle.

Definition 18. *Given a visibility range r, $G_r = (V, E_r)$, where $E_r = \{\{u,v\} : [C_u, C_v] \cap B = \emptyset$ and $d(C_u, C_v) \leq r\}$ is the visibility graph of G.*

Let $S_t \subseteq V$ denote the set of nodes occupied by defenders at time step t. The decision problem is extended from Prob. 8 with the additional requirement of connectivity formulated as

Problem 9. *Given an instance $(G, R, \lambda_0, A, D, \lambda_+^A)$, is it possible to navigate the defenders in a way that no attacker reaches its target, and the induced subgraph $G_r[S_t]$ of the connectivity graph remains connected at every time step t?*

PSPACE-hardness of APP implies also the same result for APPC.

Chapter 5

Contribution of the Thesis

This thesis is a compilation of six papers in which three independent topics are looked into. The first three papers are focused on ad-hoc wireless networks discussed in Chapter 2. The fourth paper deals with the broadcast time problem addressed in Chapter 3. These four papers share the common characteristic that integer programming is an essential solution method. In this sense, the last two papers are distinguished from the former. They are devoted to adversarial variants of path finding for multiple robots summarized in Chapter 4, which are commonly approached by methods prevalent in the field of artificial intelligence.

5.1 Paper I: Shared Multicast Trees in Ad-hoc Wireless Networks

The contribution of Paper I lies in introducing the Shared Multicast Tree (SMT) problem, a generalization of the Shared Broadcast Tree (SBT) problem. An ILP model for SMT is proposed by modification of a model for SBT presented in [81]. Additional constraints related to non-destination nodes are included in the model, and constraints in the existing model for SBT are quantified with respect to the non-destinations. The presented model is subsequently proved to be a correct formulation of SMT. Appendix A of Paper I contains details of the proof.

Further, several inexact construction methods are proposed. The first two methods are based on construction of a solution to different problems, specifically MST and MEB, respectively. The construction is followed by local search, which identifies non-destinations whose presence in the solution is disadvantageous. These non-destinations are then eliminated from the solution. The third method is an algorithm devised specifically for SBT/SMT. Its main idea is to gradually expand a solution by adding new edges, anticipate subtree sizes in the resulting solution, and thereby give an estimate of the final objective function value each time a new edge is appended.

The experimental part is divided into two sections. The first section focuses on investigation of the instance sizes that are solvable by the proposed ILP model using the CPLEX solver. It is also studied how the increasing number of destinations and non-destinations affects the solution time and the objective function value. In the second section, the comparison of the inexact algorithms indicates that the method based on subtree size anticipation outperforms the other two algorithms.

5.2 Paper II: The Shared Broadcast Tree Problem and MST

Because of the limited size of instances practically solvable to optimality, following from the computational complexity of SBT, approximability and approximation algorithms are often researched. Paper II presents an instance of SBT for which the algorithm that construct a MST yields a solution to SBT with objective function value six times the optimum, proving that the approximation ratio of the algorithm that constructs a MST as a solution to SBT is at least 6. In addition, experimental results then reveal that the ratio between the optimal objective function value, and the objective function value of MST solutions in randomly generated instances is much more favourable than the lower bound on the approximation ratio.

5.3 Paper III: Integer Programming Formulations for the Shared Multicast Tree Problem

Two ILP formulations for SMT are developed in Paper III. The first model is based on broadcast trees, whereas the second one adapts several network flow techniques presented in [58]. Both models are subsequently extended by redundant variables and corresponding constraints which strengthen the formulations. The models are further strengthened by introducing valid inequalities. A theoretical study of the models proves that network flow based models are at least as strong as corresponding models built on broadcast trees. The experimental evaluation discovers instances showing that flow based models are in fact stronger.

Experimental evaluation further exhibits a profound trade-off between the time necessary for solving LP relaxation of the models, and the strength of the lower bound obtained. The LP relaxation of the strongest model yields an integral solution in most of the instances. However, its running time rules out its practical usability. For addressing this issue, a constraint generation (CG) technique is developed, which increases the size of practically solvable instances. Lower bounds are also yielded during the course of standard branch and bound algorithm. A comparison of lower bound produced from CG and branch and bound shows that CG is able to obtain stronger lower bounds within the selected time limit of 20 minutes.

Paper III also contains an adaptation of a metaheuristic algorithm from [55], originally developed for the minimum Steiner tree problem, combined with local search methods developed in Paper I. The algorithm is able to solve a vast majority of tested instances to optimality. However, the optimality is proved only in instances for which it is possible apply branch and bound and solve them to optimality within a practical time. In larger instances, the computed solutions are not proved to be optimal, but the results indicate a very good potential of the metaheuristic algorithm.

5.4 Paper IV: Computing the Broadcast Time of a Graph

We develop a straightforward ILP model for the Minimum Broadcast Time (MBT) problem, as the existing literature focusing on mathematical models for MBT is rather scarce. Recently published [14] contains a non-linear mathematical formulation of

MBT, and authors in [24] present an ILP model. The models in these works serve for the purpose of formal description of MBT, and are not investigated further as potential solution methods. On the contrary, an exact method that exploits the problem characteristics and iteratively solves a decision version of the presented ILP model is devised in Paper IV. This method is also applied to the LP relaxation of the model, and the outcome indicates that it yields fairly strong lower bounds, often coinciding with the optimum.

Besides the continuous relaxation, analytical and combinatorial lower bounding techniques are studied. According to the experimental evaluation, these methods provide weaker lower bounds than the LP relaxation. Upper bounds are obtained by a greedy algorithm of which the main feature is an iterative construction of broadcast trees of restricted size. This algorithm is parametrized by the size limitation of the broadcast trees. The larger trees are searched, the tighter upper bound can be achieved.

The numerical experiments also provide an insight into the relation between some of the graph properties and its broadcast time. It has been observed that graphs with more nodes as well as denser graphs tend to have its broadcast time closer to its trivial lower bound $\log(n)$ (see Sect. 3.4.1). Also, increasing size and density of the graph instances narrows the gap between upper and lower bounds.

5.5 Paper V: Area Protection in Adversarial Path-finding Scenarios with Multiple Mobile Agents on Graphs

Paper V introduces the Area Protection Problem (APP) as a modification of the previously studied Adversarial Cooperative Path Finding (ACPF) [39] problem. Its main contribution lies in the proof that APP is PSPACE-hard. This result is achieved by demonstrating a polynomial-time reduction from TRUE QUANTIFIED BOOLEAN FORMULA (TQBF).

Several strategies are investigated for the team of defenders. A building block of all the considered strategies is a so called single stage *destination allocation*, in which a node is assigned to each defender at the beginning of the agents' movement. The defenders then try to reach these destinations by any cooperative path-finding algorithm, in this case we selected local repair A* (see Sect. 4.2). A destination can be regarded as a target node for defenders, the difference is that while a target is part of the input, the destination is determined by a solution method.

The strategies differ in the approach of target allocation. The two simplest methods, random and greedy allocation, select attackers' targets as destinations for defenders. A more sophisticated method, *bottleneck simulation*, runs a simulation of attackers' movement and tries to predict locations frequently passed by attackers. These locations are assumed to be bottlenecks in the environment, and blocking them may prevent a larger number of attackers from reaching their targets.

The experiments are carried out on different environment types and different positions of teams. This method is particularly successful in environments rich on bottlenecks, as it correctly identifies them.

5.6 Paper VI: Maintaining Ad-hoc Communication Network in Area Protection Scenarios with Adversarial Agents

An extension of APP, in which defenders are required to maintain the possibility of communication between each other, is presented in Paper VI. The possibility of communication is modeled by connectivity of the communication graph, whose nodes are the defenders' locations, and whose edges connect node pairs between which agents can communicate.

We approach this problem by dividing the defenders into *communicators* and *occupiers* with different purposes. Occupiers have the same task as regular defenders, i.e., they intend to prevent attackers from reaching their targets, while communicators are supposed to ensure the connectivity maintenance by moving to suitable nodes.

From a theoretical point of view, we show that the problem whether it is possible for communicators to maintain communication when all occupiers reach their destinations is NP-complete, which was proved by reducing VERTEX COVER to it.

Bibliography

[1] AKYILDIZ, I., AND VURAN, M. C. *Wireless Sensor Networks*. John Wiley & Sons, Inc., New York, NY, USA, 2010. 2.1

[2] ALAMI, R., FLEURY, S., HERRB, M., INGRAND, F., AND ROBERT, F. Multi-robot cooperation in the martha project. *IEEE Robotics Automation Magazine 5*, 1 (March 1998), 36–47. 4.2

[3] ALTHAUS, E., CALINESCU, G., MANDOIU, I. I., PRASAD, S., TCHERVENSKI, N., AND ZELIKOVSKY, A. Power efficient range assignment in ad-hoc wireless networks. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.* (March 2003), vol. 3, pp. 1889–1894 vol.3. 2.2.3

[4] AMBÜHL, C. An optimal bound for the mst algorithm to compute energy efficient broadcast trees in wireless networks. In *Automata, Languages and Programming* (Berlin, Heidelberg, 2005), L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., Springer Berlin Heidelberg, pp. 1139–1150. 2.2.2

[5] AMBÜHL, C., CLEMENTI, A. E. F., PENNA, P., ROSSI, G., AND SILVESTRI, R. Energy consumption in radio networks: Selfish agents and rewarding mechanisms. In *WAOA* (2003). 2.2.3

[6] ARCHER, A. F. A modern treatment of the 15 puzzle. *American Mathematical Monthly 106* (1999), 793–799. 4.2.1

[7] BARTA, J., LEGGIERI, V., MONTEMANNI, R., NOBILI, P., AND TRIKI, C. Some valid inequalities for the probabilistic minimum power multicasting problem. *Electronic Notes in Discrete Mathematics 36* (2010), 463 – 470. ISCO 2010 - International Symposium on Combinatorial Optimization. 2.2.2

[8] BAUER, J., HAUGLAND, D., AND YUAN, D. New results on the time complexity and approximation ratio of the broadcast incremental power algorithm. *Information Processing Letters 109*, 12 (2009), 615 – 619. 2.2.2

[9] BLOUGH, D. M., LEONCINI, M., RESTA, G., AND SANTI, P. *On the Symmetric Range Assignment Problem in Wireless Ad Hoc Networks*. Springer US, Boston, MA, 2002, pp. 71–82. 2, 2.2.3

[10] CALAMONERI, T., CLEMENTI, A. E., MONTI, A., ROSSI, G., AND SILVESTRI, R. Minimum-energy broadcast in random-grid ad-hoc networks: Approximation and distributed algorithms. In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (New York, NY, USA, 2008), MSWiM '08, ACM, pp. 354–361. 2.2.2

[11] CALINESCU, G., MANDOIU, I., AND ZELIKOVSKY, A. Symmetric connectivity with minimum power consumption in radio networks. vol. 223, pp. 119–130. 2.2.3

[12] CARMI, P., AND CHAITMAN-YERUSHALMI, L. On the minimum cost range assignment problem. In *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings* (2015), pp. 95–105. 2.2.3

[13] CHROBAK, M., GASIENIEC, L., AND RYTTER, W. Fast broadcasting and gossiping in radio networks. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 2000), FOCS '00, IEEE Computer Society, pp. 575–. 3.4.2

[14] CHU, X., AND CHEN, Y. Time division inter-satellite link topology generation problem: Modeling and solution. *International Journal of Satellite Communications and Networking 36*, 2 (2017), 194–206. 3, 5.4

[15] CLEMENTI, A. E. F., CRESCENZI, P., PENNA, P., ROSSI, G., AND VOCCA, P. On the complexity of computing minimum energy consumption broadcast subgraphs. In *STACS 2001* (Berlin, Heidelberg, 2001), A. Ferreira and H. Reichel, Eds., Springer Berlin Heidelberg. 2

[16] CLEMENTI, A. E. F., IANNI, M. D., AND SILVESTRI, R. The minimum broadcast range assignment problem on linear multi-hop wireless networks. *Theor. Comput. Sci. 299* (2003), 751–761. 2.2.3

[17] CLEMENTI, A. E. F., PENNA, P., AND SILVESTRI, R. Hardness results for the power range assignment problem in packet radio networks. In *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques* (1999), D. S. Hochbaum, K. Jansen, J. D. P. Rolim, and A. Sinclair, Eds., Springer Berlin Heidelberg, pp. 197–208. 2.2.3

[18] CLEMENTI, A. E. F., PENNA, P., AND SILVESTRI, R. The power range assignment problem in radio networks on the plane. In *STACS 2000* (Berlin, Heidelberg, 2000), H. Reichel and S. Tison, Eds., Springer Berlin Heidelberg, pp. 651–660. 2.2.3

[19] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001. 3.3.1

[20] CRESCENZI, P., AND KANN, V. Approximation on the web: A compendium of NP optimization problems. In *Randomization and Approximation Techniques in Computer Science, International Workshop, RANDOM'97, Bolognna, Italy, July 11-12. 1997, Proceedings* (1997), pp. 111–118. 1.3

[21] DANIEL, R., AND K., W. M. Finding a shortest solution for the n x n extension of the 15-puzzle is intractable. In *AAAI* (1986), T. Kehler, Ed., Morgan Kaufmann, pp. 168–172. 4.2.1

[22] DAS, A. K. Minimum power broadcast trees for wireless networks: Integer programming formulations. In *INFOCOM* (2003). 2.2.2

[23] DAS, A. K., MARKS, R. J., EL-SHARKAWI, M., ARABSHAHI, P., AND GRAY, A. Optimization methods for minimum power bidirectional topology construction in wireless networks with sectored antennas. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.* (Nov 2004), vol. 6, pp. 3962–3968 Vol.6. 2.2.3

[24] DE SOUSA, A., GALLO, G., GUTIERREZ, S., ROBLEDO, F., RODRÍGUEZ-BOCCA, P., AND ROMERO, P. Heuristics for the minimum broadcast time. *Electronic Notes in Discrete Mathematics 69* (2018), 165 – 172. Joint EURO/ALIO International Conference 2018 on Applied Combinatorial Optimization (EURO/ALIO 2018). 3.3, 5.4

[25] DE WILDE, B., TER MORS, A., AND WITTEVEEN, C. Push and rotate: cooperative multi-agent path planning. In *AAMAS* (2013). 4.2.2

[26] DEKKER, A. Applying social network analysis concepts to military . . . *Connections 24* (2002), 93–103. 3

[27] DIETZFELBINGER, M. Gossiping and broadcasting versus computing functions in networks. *Discrete Applied Mathematics 137*, 2 (2004), 127 – 153. 3.4.2

[28] DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numer. Math. 1*, 1 (1959), 269–271. 4.1

[29] EVERETT, H. R., GAGE, D. W., GILBREATH, G. A., LAIRD, R. T., AND SMURLO, R. P. Real-world issues in warehouse navigation. vol. 2352, pp. 2352 – 2352 – 11. 4.2

[30] FARLEY, A., HEDETNIEMI, S., MITCHELL, S., AND PROSKUROWSKI, A. Minimum broadcast graphs. *Discrete Mathematics 25*, 2 (1979), 189 – 193. 3.4.1

[31] FLAMMINI, M., NAVARRA, A., AND PERENNES, S. The “real” approximation factor of the mst heuristic for the minimum energy broadcasting. *J. Exp. Algorithmics 11* (Feb. 2007). 2.2.2

[32] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. 3.2

[33] HADA, Y., AND TAKASE, K. Multiple mobile robot navigation using the indoor global positioning system (igps). In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)* (Oct 2001), vol. 2, pp. 1005–1010 vol.2. 4.2

[34] HALGAMUGE, M. N., ZUKERMAN, M., RAMAMOHANARAO, K., AND VU, H. L. An estimation of sensor energy consumption. *Electromagnetics Research B*, 12 (2009), 259–295. 2, 2.2.4

[35] HART, P. E., NILSSON, N. J., AND RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC-4(2)* (1968), 100–107. 4.1

[36] HARUTYUNYAN, H. A., AND LIESTMAN, A. L. More broadcast graphs. *Discrete Applied Mathematics 98*, 1-2 (1999), 81–102. 3.4.1

[37] HAUGLAND, D., AND YUAN, D. *Compact Integer Programming Models for Power-optimal Trees in Ad Hoc Wireless Networks*, vol. 158. 01 2011, pp. 219–246. 2.2.2, 2.2.3

[38] IVANOVÁ, M. Shared multicast trees in ad hoc wireless networks. In *Combinatorial Optimization - 4th International Symposium, ISCO 2016, Vietri sul Mare, Italy, May 16-18, 2016, Revised Selected Papers* (2016), pp. 273–284. 2.2.4

[39] IVANOVÁ, M., AND SURYNEK, P. Adversarial cooperative path-finding: Complexity and algorithms. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014* (2014), pp. 75–82. 4.3.1, 5.5

[40] IVANOVÁ, M., SURYNEK, P., AND HIRAYAMA, K. Area protection in adversarial path-finding scenarios with multiple mobile agents on graphs - A theoretical and experimental study of strategies for defense coordination. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence, ICAART 2018, Volume 1, Funchal, Madeira, Portugal, January 16-18, 2018.* (2018), pp. 184–191. 4.3.2

[41] IVANOVÁ, M., SURYNEK, P., AND NGUYEN, D. T. N. Maintaining ad-hoc communication network in area protection scenarios with adversarial agents. In *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018.* (2018), pp. 348–353. 4.3.3

[42] JANSEN, K., AND MÜLLER, H. The minimum broadcast time problem for several processor networks. *Theoretical Computer Science 147*, 1 (1995), 69 – 85. 3.2, A.1.1

[43] KIROUSIS, L. M., KRANAKIS, E., KRIZANC, D., AND PELC, A. Power consumption in packet radio networks. *Theor. Comput. Sci. 243*, 1-2 (July 2000), 289–305. 2.2.3

[44] KORNHAUSER, D., MILLER, G., AND SPIRAKIS, P. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *25th Annual Symposium onFoundations of Computer Science, 1984.* (Oct 1984), pp. 241–250. 4.2.1

[45] LIANG, W. Constructing minimum-energy broadcast trees in wireless ad hoc networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing* (New York, NY, USA, 2002), MobiHoc '02, ACM, pp. 112–122. 2.2.2

[46] LIESTMAN, A. L. Fault-tolerant broadcast graphs. *Networks 15*, 2 (1985), 159–171. 3.4.1

[47] LUNA, R., AND BEKRIS, K. E. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One* (2011), IJCAI'11, AAAI Press, pp. 294–300. 4.2.2

[48] MCGARVEY, R. G., RIEKSTS, B. Q., VENTURA, J. A., AND AHN, N. Binary linear programming models for robust broadcasting in communication networks. *Discrete Appl. Math. 204*, C (2016), 173–184. 3.4.1

[49] MCGRATH, M. J., AND SCANAILL, C. N. *Sensor Network Topologies and Design Considerations*. Apress, Berkeley, CA, 2013, pp. 79–95. 2.1

[50] MIDDENDORF, M. Minimum broadcast time is np-complete for 3-regular planar graphs and deadline 2. *Information Processing Letters 46*, 6 (1993), 281 – 287. 3.2

[51] MIN, M., PROKOPYEV, O., AND PARDALOS, P. M. Optimal solutions to minimum total energy broadcasting problem in wireless ad hoc networks. *Journal of Combinatorial Optimization 11*, 1 (Feb 2006), 59–69. 2.2.2

[52] MONTEMANNI, R., AND GAMBARDELLA, L. Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. *Computers & Operations Research 32*, 11 (2005), 2891 – 2904. 2.2.3

[53] MONTEMANNI, R., LEGGIERI, V., AND TRIKI, C. Mixed integer formulations for the probabilistic minimum energy broadcast problem in wireless networks. *European Journal of Operational Research 190*, 2 (10 2008), 578–585. 2.2.2

[54] NEMHAUSER, G. L., AND WOLSEY, L. A. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988. 1.2

[55] PAJOR, T., UCHOA, E., AND WERNECK, R. F. A robust and scalable algorithm for the steiner problem in graphs. *Mathematical Programming Computation 10*, 1 (Mar 2018), 69–118. 5.3

[56] PAPADIMITRIOU, C. H. *Computational complexity.* Addison-Wesley, 1994. 3.2

[57] PAPADIMITRIOU, I., AND GEORGIADIS, L. Minimum-energy broadcasting in multi-hop wireless networks using a single broadcast tree. *Mobile Networks and Applications 11*, 3 (Jun 2006), 361–375. 2.2.4

[58] POLZIN, T., AND DANESHMAND, S. V. A comparison of steiner tree relaxations. *Discrete Applied Mathematics 112*, 1 (2001), 241 – 261. Combinatorial Optimization Symposium, Selected Papers. 5.3

[59] RENEE JANSEN, M., AND R. STURTEVANT, N. Direction maps for cooperative pathfinding. 4.2.2

[60] ROSS KINSELLA RYAN, M. Exploiting subgraph structure in multi-robot path planning. *Journal of Artificial Intelligence Research 31* (01 2008), 497–542. 4.2

[61] SANTI, P. Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv. 37*, 2 (June 2005), 164–194. 2.2.3

[62] SCOTT STANDLEY, T. Finding optimal solutions to cooperative pathfinding problems. vol. 1. 4.2.2

[63] SHARMA, D. A., VERMA, S., AND SHARMA, K. Network topologies in wireless sensor networks : A review. 2.1

[64] SHARON, G., STERN, R., GOLDENBERG, M., AND FELNER, A. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence 195* (2013), 470 – 495. 4.2.2

[65] SILVER, D. Cooperative pathfinding. In *Proc. of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, 2005* (2005), pp. 117–122. 4.2, 4.2.2, 4.2.2

[66] SILVER, D., HUANG, A., MADDISON, C. J., GUEZ, A., SIFRE, L., VAN DEN DRIESSCHE, G., SCHRITTWIESER, J., ANTONOGLOU, I., PANNEERSHELVAM, V., LANCTOT, M., DIELEMAN, S., GREWE, D., NHAM, J., KALCHBRENNER, N., SUTSKEVER, I., LILLICRAP, T., LEACH, M., KAVUKCUOGLU, K., GRAEPEL, T., AND HASSABIS, D. Mastering the game of Go with deep neural networks and tree search. *Nature 529*, 7587 (jan 2016), 484–489. 4.3

[67] SIPSER, M. *Introduction to the Theory of Computation*, second ed. Course Technology, 2006. 1.3

[68] SLATER, P., COCKAYNE, E., AND HEDETNIEMI, S. Information dissemination in trees. *SIAM Journal on Computing 10*, 4 (1981), 692–701. 3.2, 3.2

[69] SURYNEK, P. An optimization variant of multi-robot path planning is intractable. In *AAAI* (2010). 4.2.1

[70] SURYNEK, P. Multi-robot path planning. In *Multi-Robot Systems*, T. Yasuda, Ed. IntechOpen, Rijeka, 2011, ch. 14. 4.2

[71] TOH, C. K. *Ad Hoc Wireless Networks: Protocols and Systems*, 1st ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. 2

[72] ČAGALJ, M., HUBAUX, J.-P., AND ENZ, C. Minimum-energy broadcast in all-wireless networks: Np-completeness and distribution issues. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking* (New York, NY, USA, 2002), MobiCom '02, ACM, pp. 172–182. 2.2.2

[73] WAN, P.-J., CĂLINESCU, G., LI, X.-Y., AND FRIEDER, O. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless Networks 8*, 6 (Nov 2002), 607–617. 2.2.2

[74] WANG, K.-H. C., AND BOTEA, A. Fast and memory-efficient multi-agent pathfinding. In *Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling* (2008), ICAPS'08, AAAI Press, pp. 380–387. 4.2.2

[75] WIESELTHIER, J. E., NGUYEN, G. D., AND EPHREMIDES, A. On the construction of energy-efficient broadcast and multicast trees in wireless networks. pp. 585–594. 2, 2.2.2, 2.2.2, 2.2.2

[76] WILLIAMSON, D. P., AND SHMOYS, D. B. *The Design of Approximation Algorithms*, 1st ed. Cambridge University Press, New York, NY, USA, 2011. 13, 15

[77] WOLSEY, L. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. 1.2, 6, 2, 3

[78] YOUNIS, M., AND OZER, S. Z. Wireless ad hoc networks: technologies and challenges. *Wireless Communications and Mobile Computing 6*, 7 (2006), 889–892, doi: 10.1002/wcm.449. 2

[79] YU, J., AND LAVALLE, S. M. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.* (2013). 4.2.1

[80] YUAN, D. Computing optimal or near-optimal trees for minimum-energy in wireless networks. In *Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05)* (April 2005), pp. 323–331. 2.2.2

[81] YUAN, D., AND HAUGLAND, D. Dual decomposition for computational optimization of minimum-power shared broadcast tree in wireless networks. *IEEE Transactions on Mobile Computing 11* (2012), 2008–2019. 2.2.4, 5.1

Chapter 6

Attached Papers

Paper I

6.1 Shared Multicast Trees in Ad Hoc Wireless Networks

Marika Ivanova

In: Cerulli R., Fujishige S., Mahjoub A. (eds) *Combinatorial Optimization.* ISCO 2016. Lecture Notes in Computer Science, vol 9849. Springer, Cham

Shared Multicast Trees in Ad Hoc Wireless Networks

Marika Ivanova

University of Bergen, Norway
Marika.Ivanova@uib.no
http://www.uib.no/en/persons/Marika.Ivanova

Abstract. This paper addresses a problem of shared multicast trees (SMT), which extends a recently studied problem of shared broadcast trees (SBT). In SBT, a common optimal tree for a given set of nodes allowing broadcasting from any node to the rest of the group is searched. In SMT, also nodes that neither initiate any transmission, nor act as destinations are considered. Their purpose is exclusively to relay messages between nodes. The optimization criterion is to minimize the energy consumption. The present work introduces this generalization and devises solution methods. We model the problem as an integer linear program (ILP), in order to compute the exact solution. However, the size of instances solvable by ILP is significantly limited. Therefore, we also focus on inexact methods allowing us to process larger instances. We design a fast greedy method and compare its performance with adaptations of algorithms solving related problems. Numerical experiments reveal that the presented greedy method produces trees of lower energy than alternative approaches, and the solutions are close to the optimum.

Keywords: ad hoc wireless network, Steiner tree, multicast communication, ILP model, heuristic algorithm

1 Introduction

The purpose of a multicast communication in a wireless ad-hoc network is to route information from a source to a set of destinations. Given a set of devices and distances between them, the task is to assign power to each device (node), so that the demands of the network are met and the energy consumption is as low as possible, assuming their locations are fixed. Power efficiency is an important aspect in constructing ad-hoc wireless networks since the devices are typically heavily energy-constrained. Individual devices work as transceivers, which means that they are able to both transmit and receive a signal. Moreover, the power level of a device can be dynamically adjusted during a multicast session.

Unlike wired networks, nodes in ad-hoc wireless networks use omnidirectional antennas, and hence a message reaches all nodes within the communication range of the sender. This range is determined by the power assigned to the sender, which is the maximum rather than the sum of the powers necessary to reach

all intended receivers. This feature is often referred to as the wireless multicast advantage [19].

The problems of finding power-minimizing trees in wired networks are generalizations of the minimum Steiner tree problem (e.g. [15]). Many wireless network design tasks are NP-hard [8,13]. The following problems are relevant to our work.

Minimum Energy Broadcast (MEB) is the problem of constructing an optimal arborescence for broadcasting from a given source to all remaining nodes. In order to be able to perform a broadcast session from different sources, a separate tree has to be stored for each source. A generalized multicast version assumes that the message is intended for a predefined subset of vertices. Remaining vertices can be used as intermediate nodes forwarding the message to other nodes, and possibly reduce the total cost. Such nodes are referred to as *Steiner nodes*.

Range Assignment Problem (RAP) concerns the problem of assigning transmission powers of minimum sum to the wireless nodes while ensuring network connectivity [1, 7]. Unlike MEB, the resulting links formed by the energy assignment are undirected. A generalization of the problem considers the strong connectivity within a nonempty subset of nodes.

Shared Broadcast Tree (SBT). A crucial drawback of MEB is the necessity of storing one tree for each source. The basic idea of SBT [17, 20] is to construct a common tree that is source independent and hence simplifies routing, as the relaying node does not need to know the original source in order to adjust its corresponding power level. Instead, the power level depends merely on the immediate neighbour from which the message was received. This idea is based on the observation that a signal that is being forwarded by a node does not have to reach the neighbour from which it originally came. So, if a signal comes from the most distant neighbour, the relaying power must correspond to the second most distant neighbour. When, on the other hand, a message comes from one of the closer neighbours, it has to be forwarded with the power necessary to reach the most distant one. With this conception, we get two power levels, and their selection involves only a single binary decision making.

This work introduces the shared multicast tree (SMT) problem, a generalization of SBT. Analogously to the multicast versions of MEB and RAP, in SMT we assume that there are two types of nodes, called destinations and non-destinations, respectively. Destinations can initiate a transmission, and must receive every transmission initiated by other destinations. Non-destinations can relay a message, but do not initiate any transmission. Neither do they have to receive any transmission. Passing messages via non-destinations is thus optional, and is chosen only if it saves energy, which is the main motivation for SMT. The goal is to find a common source-independent tree that connects the destinations while minimizing the power.

The decentralized nature of wireless ad-hoc networks implies its suitability for applications, where it is not possible to rely on central nodes, or where network infrastructure does not exist. This is typical for various short-term events like conferences or fixtures. Simple maintenance makes them useful in applications

such as emergency situations, disaster relief, military command and control, and most recently, in the mobile commerce sector.

2 Related Work

MEB was introduced in [19], where the authors considered three heuristic algorithms of which most cited is Broadcast Incremental Power (BIP), a greedy $\mathcal{O}(N^2 \log N)$ approximation algorithm. To our best knowledge, the most recent results for lower and upper bounds on the approximation ratio are 4.6 [3] and 6 [2], respectively. Much is written about refinements of fast sub-optimal methods, for instance [11,12,14,18]. In [7], the authors study RAP and compare cases when the resulting graph is required to be strongly and weakly connected. Several heuristic approaches are proposed (e. g. in [5,6]). Many works are also dedicated to mathematical programming techniques. Various ILP models for both MET and RAP are presented in [9, 10, 13, 16]. A special case where the transmission ranges of the stations ensure the communication between any pair of stations in at most h hops is investigated in [8].

The first work concerning SBT is [17], where the idea of a single source-independent tree embedding N broadcast trees for different sources is introduced. The authors show that using the same broadcast tree does not result in widely varying total powers for different sources. Another contribution of [17] is a polynomial-time approximation algorithm to construct a single broadcast tree, including an analysis of its performance. In [20], the authors present an ILP formulation and apply a dual decomposition method. This approach enables solving larger instances than an explicit formulation can solve, and with less than 3% performance gap to global optimality.

3 Notation and Assumptions

Let $H = (V_H, E_H)$ be an undirected graph and $u \in V_H$. The degree of u in H is denoted by $\deg_H(u)$. The input and output degree of $v \in V_K$ in a directed graph $K = (V_K, A_K)$ is denoted by $\deg_K^-(v)$ and $\deg_K^+(v)$, respectively. Let $H' = (V_{H'}, E_{H'})$ be a subgraph of H. Then, $H \setminus H'$ denotes the graph induced by the node set $V_H \setminus V_{H'}$.

A wireless network is modelled as a complete graph $G = (V, E)$, where V corresponds to the network nodes, and the edges E correspond to potential direct links between them, i.e. $\forall i, j \in V, i \neq j : \{i, j\} \in E$. The set $A = \{(i, j) : i, j \in V, i \neq j\}$ contains all arcs derived from E. Next, $D \subseteq V$ is a nonempty set of destinations with $N = |V|$ and $M = |D|$.

Let $\mathbf{z} \in \{0, 1\}^E$ be a vector with components corresponding to edges in E. The undirected graph induced by $\mathbf{z}$ is defined as $G_{\mathbf{z}} = (V, E_{\mathbf{z}})$, where $\{i, j\} \in E_{\mathbf{z}} \Leftrightarrow z_{ij} = 1$. The directed graph induced by $\mathbf{x} \in \{0, 1\}^A$ is defined analogously.

For $i, j \in V$, the power requirement for transmission from i to j is denoted by p_{ij}, and depends on the distance d_{ij} between i and j and environmental properties. More precisely, $p_{ij} = \kappa d_{ij}^{\alpha}$, where α is an environment-dependent

parameter (typically valued between 2 and 4) and κ is a constant. In this work, the power requirements p_{ij} are referred to as the *arc costs*. It follows from $d_{ij} = d_{ji}$ that the power requirements are symmetric.

If $\{i, j\}$ is an edge in a tree $T = (V_T, E_T)$, where $V_T \subseteq V$, $E_T \subseteq E$, we use $T_{i/j}$ to denote the subtree of T consisting of all vertices k such that the path from k to j visits i, as introduced in [20]. Additionally, we define a function $\mathrm{nod}(T_{i/j})$ that returns the number of destinations in $T_{i/j}$.

Neighbours of i in T are denoted i_1^T, i_2^T, $i_3^T, \ldots$ in non-increasing order of distance from i. If there is no risk of confusion, we simply omit the superscript T. The highest and second highest power levels of i are defined by its neighbours i_1 and i_2, respectively. If i is a leaf, we set $p_{ii_2} = 0$. The contribution $c_T(i)$ of i in T to the total cost depends on i's power levels:

$$c_T(i) = \mathrm{nod}(T_{i_1/i})p_{ii_2} + \mathrm{nod}(T \setminus T_{i_1/i})p_{ii_1}. \tag{1}$$

The total cost $P(T)$ of the tree T is then determined as

$$P(T) = \sum_{i \in V} c_T(i). \tag{2}$$

Problem 1. (SMT): Find a tree T in G minimizing $P(T)$ such that T spans D.

The most costly two incident edges of each node contribute to the objective function value. This reflects the nature of SBT/SMT, when the power level of a node is determined by the most costly link along which a message has to be forwarded. The power requirement of the link is multiplied by the number of senders whose transmissions are relayed through this link, which captures how often the link is used.

4 Discrete Optimization Model

We present an integer programming model for SMT, extending the model in [20] by non-destinations. In this setting we consider a set of destinations $D \subseteq V$ where a broadcast session takes place. Variables are defined as follows:

$$z_{ij} = \begin{cases} 1 & \text{if edge } \{i, j\} \in E \text{ is in } T, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{ij}^s = \begin{cases} 1 & \text{if arc } (i, j) \in A \text{ is used to transmit messages from } s \in D, \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{ij}^s = \begin{cases} 1 & \text{if node } i \in V \text{ uses power } p_{ij} \text{ to transmit messages from } s \in D, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{x^s} \in \{0, 1\}^A$ denote the vector consisting of variables x_{ij}^s for all $(i, j) \in A$. The ILP model is presented below. The x-variables induce $|D|$ directed trees,

that are encapsulated into a single spanning tree induced by the z-variables. The power levels are determined by the y-variables.

$$\min \sum_{(i,j)\in A} \sum_{s\in D} p_{ij} y_{ij}^{s} \tag{3a}$$

s.t.

$$\sum_{\{i,j\}\in E} z_{ij} \leq N-1, \tag{3b}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij}^{s} = 1 \qquad j,s\in D, j\neq s, \tag{3c}$$

$$x_{jk}^{s} \leq \sum_{i\in V\setminus\{j\}} x_{ij}^{s} \leq 1 \qquad j\in V\setminus D, s\in D, k\in V\setminus\{j\}, \tag{3d}$$

$$\sum_{i\in V\setminus\{j\}} x_{ij}^{s} \leq \sum_{k\in V\setminus\{j\}} x_{jk}^{s} \qquad s\in D, j\in V\setminus D, \tag{3e}$$

$$x_{ij}^{s} + x_{ji}^{s} = z_{ij} \qquad \{i,j\}\in E, s\in D, \tag{3f}$$

$$x_{ij}^{j} = 0 \qquad j\in D, i\in V\setminus\{j\}, \tag{3g}$$

$$x_{ij}^{s} \leq \sum_{k\in V: p_{ik}\geq p_{ij}} y_{ik}^{s} \qquad s\in D, (i,j)\in A, \tag{3h}$$

$$\mathbf{z}\in\{0,1\}^{E}, \mathbf{x},\mathbf{y}\in\{0,1\}^{A\times D}. \tag{3i}$$

By constraint (3b), we express the upper bound on the number of edges in the Steiner tree. There is also a lower bound $M-1$ on the size of the spanning tree, but addition of this constraint would neither reduce the space of feasible solutions nor increase the strength of the model. If the tree does not contain any Steiner nodes, its size is the lower bound, while if all nodes are used (either as Steiner nodes, or $D=V$), its size equals the upper bound. By (3c), we ensure that a message from source s reaches a destination j from exactly one neighbour $i\in V$. Next, constraint (3d) covers the case when $j\in V\setminus D$: If a non-destination j forwards a message from s towards k, the message must come from exactly one neighbour i. Note that assuming there is no outgoing arc from a non-destination j, constraint (3d) does not prevent j from being a leaf in $G_{\mathbf{x}^s}$. We make such undesired solutions impossible by adding constraint (3e), which reduces the set of feasible solutions. However, (3e) is not necessary, because a solution, where a non-destination that does not relay any message is assigned a non-zero power, would be filtered out by the minimization procedure. The expression (3f) says that for any edge $\{i,j\}$ in $G_{\mathbf{z}}$ a message from s is transferred via either arc (i,j) or arc (j,i). The next constraint (3g) expresses that a transmission initiated by $s\in D$ cannot reach s again, which implies non-existence of a directed cycle containing s. Finally, by (3h), we define a relation between x-variables and y-variables. When arc (i,j) is used for transmission of a message from $s\in D$, vertex i relaying the message must be assigned power at least p_{ij}. The remainder of

this section justifies that the model is a correct formulation of SMT. Proofs of all claims can be found in Appendix A.

Lemma 1. *Let* $(\mathbf{x}, \mathbf{z})$ *satisfy (3b) - (3i). A replacement of all directed arcs in* $G_{\mathbf{x}^s}$ *by undirected ones yields graph* $G_{\mathbf{z}}$*, for all* $s \in D$*.*

Lemma 2. *Let* $(\mathbf{x}, \mathbf{z})$ *satisfy (3b) - (3i) and* $s \in D$*. All arcs in a path* $(s = u_1, u_2, \ldots, u_k)$ *in digraph* $G_{\mathbf{x}^s}$ *are directed from* s *towards* u_k*.*

Proposition 1. *Let* $(\mathbf{x}, \mathbf{z})$ *satisfy constraints (3b) - (3i). If* Q *is a connected component in* $G_{\mathbf{z}}$ *such that* $V_Q \cap D \neq \emptyset$*, then,* Q *does not contain any cycle.*

Proposition 2. *If* $(\mathbf{x}, \mathbf{z})$ *satisfies constraints (3b) - (3i), then there exists a path in* $G_{\mathbf{z}}$ *between any two destinations* $s, t \in D$*.*

The optimal solution to (3a) - (3i) is a graph $G_{\mathbf{z}}$ with one connected component containing all destinations. Non-destinations outside of this component are isolated vertices, as any potential links between them would be eliminated by the optimization.

Proposition 3. *If* $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ *is an optimal solution to (3a) - (3i), then one of the connected components of* $G_{\mathbf{z}}$ *is an optimal tree in Problem 1, and* $\sum_{(i,j)\in A} \sum_{s\in D} p_{ij} y_{ij}^s = P(G_{\mathbf{z}})$.

5 Inexact Methods

Solving the ILP model presented in the previous section provides the optimal power assignment, but the computation in large instances takes prohibitively long time. Hence, we now focus on algorithms with better trade-off between optimality and runtime. Any tree T in G spanning D is a feasible solution to SMT. We study the following methods:

1. Construction by MST (minimum-weight spanning tree), where all vertices are considered as destinations, and a MST is constructed over the set V.
2. Construction by BIP, where we regard the set of vertices as an instance of MEB, and apply the BIP algorithm over V.
3. Greedy Anticipating SMT algorithm described in the following Section 5.1.

The global impact of a local change suggests that the first two algorithms are rather myopic for our purposes. Unlike MST and MEB, the nature of SBT/SMT implies that an addition of an edge does not cause only a local change of power levels. Every time a new edge is appended, all nodes already included in the tree increase their contributions to the resulting cost, because the addition of an edge also increases the size of corresponding subtrees of every interior node. Therefore, the new cost cannot be calculated in constant time.

In general, the algorithms work in two phases. The first phase, construction, creates a spanning tree according to a certain strategy. Further improvements can be achieved in the second phase (refinement) which can be applied regardless of what construction method is used.

5.1 Greedy SMT Approach

We present in Alg. 1 GASMT, a greedy algorithm aimed to construct a Steiner tree $T = (V_T, E_T)$, with low SMT cost. The algorithm starts with T containing only a pre-defined root, and iteratively expands T by an edge until all destinations are present in V_T. The selection of the new edge is based on an anticipation of the entire resulting tree.

Algorithm 1 Greedy Anticipating SMT (GASMT)

Input: Complete graph $G = (V, E)$, root $r \in V$, destinations $D \subseteq V$
Output: Steiner tree $T = (V_T, E_T)$, $D \subseteq V_T \subseteq V$, $E_T \subseteq E$

```
 1: procedure BuildTree(G)
 2:     T ← ({r}, ∅)
 3:     while D ⊈ V_T do
 4:         bestCost ← ∞
 5:         Cand ← {{i, j} : i ∈ V_T, j = arg min{d_ik : k ∈ V \ V_T}}
 6:         for each {i, j} ∈ Cand do
 7:             T' ← AnticipateTree(T, i, j)
 8:             if P(T') < bestCost then
 9:                 bestCost ← P(T')
10:                 {i*, j*} ← {i, j}
11:         T ← (V_T ∪ {j*}, E_T ∪ {{i*, j*}})
        return T
12:
13: procedure AnticipateTree(T, i, j)
14:     T' ← (V_T ∪ {j}, E_T ∪ {{i, j}})
15:     Disconnected ← V \ V_T'
16:     for each v ∈ Disconnected ∩ D do
17:         u ← arg min{d_kv : k ∈ V_T'}
18:         T' ← (V_T' ∪ {v}, E_T' ∪ {{u, v}})
        return T'
```

Before a new edge is appended, we determine a set *Cand* of potential edges that can be selected: for every $u \in V_T$, we remember a potential edge linking u to the closest $v \in V \setminus V_T$ (line 5 in Alg. 1). For each candidate edge $\{i, j\}$, we build an anticipated tree spanning D. The edge $\{i, j\}$ is temporarily appended to T, which produces tree T'. Subsequently, all destinations that are not yet included in T' are connected one by one to the growing anticipated T' using the shortest possible edges. The candidate link resulting in the cheapest anticipated tree is then selected and added permanently to T. The purpose of the anticipation procedure is to predict the sizes of individual subtrees in the final tree. This allows a more realistic estimation of the resulting objective value, in contrast to the construction by MST and BIP. Non-destinations are disregarded in the anticipation procedure, because they do not alter the subtree sizes.

5.2 Refinement

Any construction algorithm can be followed by an additional phase refining the existing tree T. In particular, this phase handles non-destinations, and replaces expensive transmissions by cheaper ones.

Although the use of non-destinations may reduce the cost, the construction phase does not guarantee that all non-destinations do. Thus, cost reductions can be achieved by removing non-destinations that actually deteriorate the tree. How a non-destination v is processed depends on its degree:

- $\deg(v) = 1$: Non-destination leaves can immediately be deleted recursively.
- $\deg(v) = 2$: Let $(v_1, \ldots, v_m)$ be a maximal path in T such that $\deg(v_1) = \cdots = \deg(v_m) = 2$ and $v_1, \ldots, v_m \in V \setminus D$, and consider the two connected components T_1 and T_2 arising when the path is deleted from T. If there exists an edge $e \in E$ connecting T_1 and T_2 such that $P(T') < P(T)$, where $T' = (V_{T_1} \cup V_{T_2}, E_{T_1} \cup E_{T_2} \cup \{e\})$, the path is replaced by the best choice of e. If no such edge exists, T .
- $\deg(v) \geq 3$: Let $E(v)$ be the set of edges incident to v in T and let $T' = (V_T \setminus \{v\}, (E_T \setminus E(v)) \cup E_{\mathrm{MST}})$, where E_{MST} is the set of edges of a MST constructed over the set of v's neighbours in T. If $P(T') < P(T)$, the current tree is updated to T'.

The cost of the tree can be further improved by eliminating unnecessary transmissions by means of so called "sweep" operations [19]. After removal of an edge e, the vertices are partitioned into a cut. We then select and include the edge across the cut leading to the cheapest tree, possibly e itself. This procedure can be done for all edges, or only for selected ones - for example it makes sense to test only edges longer than a certain threshold.

6 Experimental Evaluation

We have implemented the ILP model as well as the inexact algorithms and compared numerically their performance in terms objective value and runtime.

The input parameters of the procedure generating individual instances are the number of all vertices and the number of destinations. It generates instances with the intended number of destinations and non-destinations with random coordinates uniformly distributed on a square. Finally, the power requirements are determined using $p_{ij} = \kappa d_{ij}^{\alpha}$ with $\kappa = 1$ and $\alpha = 2$. All experiments were run on an Intel Core 2 Quad CPU at 2.83 GHz and 7 GB RAM.

6.1 ILP Model

The integer programming formulation was implemented in AMPL and submitted to solver CPLEX [21] which computed optimal solutions as well as LP relaxations of the generated instances. The running time of determining the optimal solution for instances containing more than 22 vertices becomes excessively long, and so

we computed only the corresponding LP relaxations, which gives lower bounds on the objective function value.

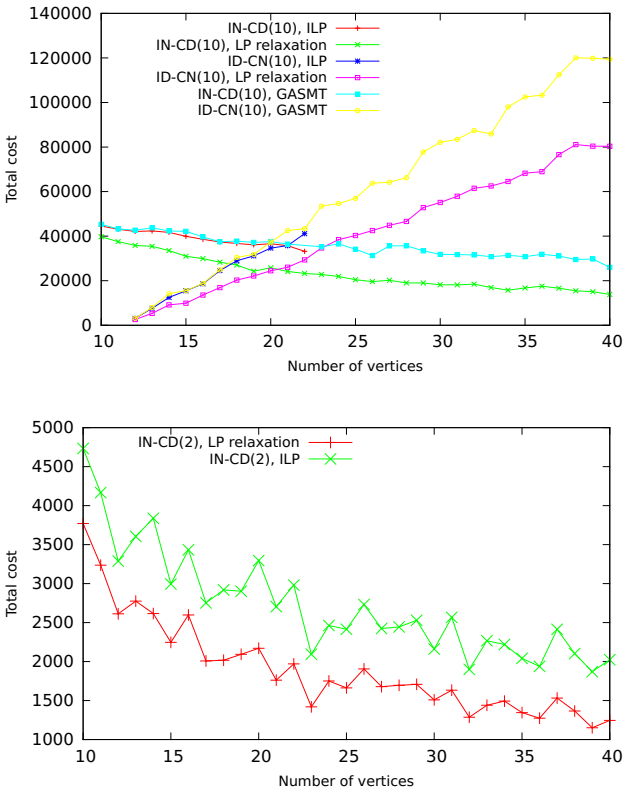


Fig. 1: Dependence of the total cost on the number of all vertices

Two instance settings were tested. In the first setting, we kept the number of non-destinations $|V \setminus D|$ constant, while increasing the number of destinations $|D|$. The abbreviation ID-CN is used to refer to this series of experiments, followed by $|V \setminus D|$ whenever it needs to be specified. In the second setting, $|D|$ was constant while $|V \setminus D|$ was increasing (IN-CD).

The first series of experiments concerns the change of the objectve function value with growing instance size. It is obvious from the graphs in Fig. 1, that in ID-CN, increasing the number of vertices also increases the total SMT cost. On the other hand, in IN-CD, the total cost decreases. This decline gradually mitigates, and it can be assumed that the average cost converges to a constant value. By way of contrast, the first graph in Fig. 1 also contains the costs obtained by the inexact algorithm. The difference between the optimum and the result of GASMT is almost negligible.

The second series of experiments shows how fast the CPU time grows with increasing number of nodes. It is apparent that the time used by the solver grows faster in ID-CN than in IN-CD, as seen in Fig. 2. Nevertheless, in both settings, the time grows exponentially. In the smallest instances, the CPU time is longer for IN-CD, because the number of destinations is higher than in ID-CN. This difference is gradually reduced. From approximately 20 vertices on, the solving time of the LP-relaxations in ID-CN becomes longer. In IN-CD(10) and ID-CN(10), every added destination causes an average increase in the ILP solution time by 89% and 208%, respectively.

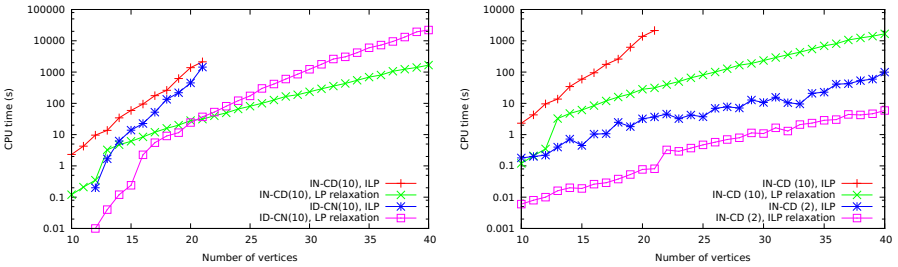


Fig. 2: Dependence of the solution time on the total number of vertices

6.2 Greedy and Heuristic Approach

The next set of experiments compares the objective value of inexact solutions produced by GASMT and the construction by MST and BIP discussed in Sect. 5. Each run of an inexact algorithm was followed by a refinement procedure, namely two iterations of non-destination removal and two iterations of sweep operations for every edge. The graphs in Fig. 3 show the results of two experimental settings, IN-CD(10) (left) and ID-CN(10) (right). Each column in the graphs corresponds to the average SMT cost calculated for 100 instances with fixed $|V|/|D|$ ratio.

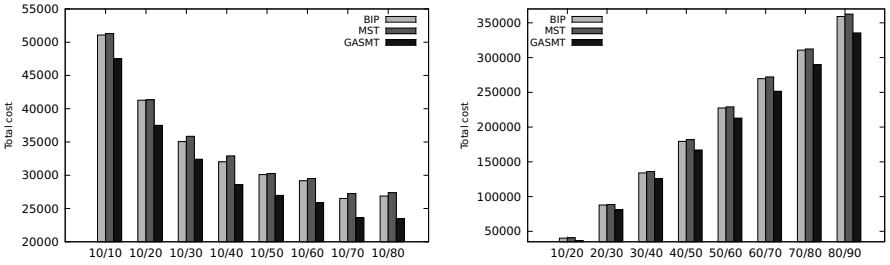


Fig. 3: Comparison of the greedy methods. The fractions on the x-axis determine the corresponding $|D|/|V|$ values.

It can be seen that the cost of the solutions produced by the construction by MST and BIP are similar, but GASMT is always perceptibly better. Nevertheless, worse time complexity of GASMT becomes apparent while processing instances of around 100 nodes, when the time spent on one instance is approximately units of minutes. On the contrary, the other two methods return the solution almost immediately.

7 Conclusion and Future Work

We have introduced a multicast version of SBT, a natural generalization of the problem. We have proposed a discrete optimization model together with the proof of its correctness. Due to the limited size of instances solvable in a practical time, heuristic and greedy approaches are also developed.

Moreover, we have conducted several numerical experiments using the CPLEX solver. It turned out that the presented ILP model can be used for solving instances with up to around 20 vertices. An increasing number of destinations causes a much faster growth of the solution time than an increasing number of non-destinations. In addition, these experiments give us insight into the cost reduction as a function of increasing number of non-destinations.

Further experiments involved the inexact methods. The GASMT algorithm presented in this work gives better results than the construction by MST and BIP applied on SMT. Moreover, the solutions provided by GASMT are close to the optimal ones determined by solving the ILP model.

A subject of continued research is a detailed theoretical study of the inexact methods. There are several interesting questions regarding this topic, like whether any inexact method is an approximation algorithm for SBT/SMT, or whether any method performs consistently better than others. There is also a substantial room for further improvements of the ILP model so that it can be applicable for larger instances. In particular, methods like strong valid inequalities, lazy constraints and user cuts could serve for this purpose.

References

1. Althaus, E., Calinescu, G., Mandoiu, I.I., Prasad, S., Tchervenski, N., Zelikovsky, A.: Power efficient range assignment in ad-hoc wireless networks. Wireless Communications and Networking. 3, 1889–1894 (2003)
2. Ambühl, C.: An Optimal Bound for the MST Algorithm to Compute Energy Efficient Broadcast Trees in Wireless Networks. ICALP'05 Proceedings of the 32nd international conference on Automata, Languages and Programming. 1139–1150 (2005)
3. Bauer, J., Haugland, D., Yuan, D.: New results on the time complexity and approximation ratio of the Broadcast Incremental Power algorithm. INFORMATION PROCESSING LETTERS. 109, 12, 615–619 (2009)
4. Bauer, J., Haugland, D., Yuan, D.: A fast local search method for minimum energy broadcast in wireless ad hoc networks. Operations Research Letters. 37, 2, 75–79 (2009)

5. Bein, D., Zheng, S. Q.: Energy efficient all-to-all broadcast in all-wireless networks. Information Sciences. 180, 10, 1781–1792 (2010)
6. Bhukya, W.N., Singh, A.: An effective heuristic for construction of all-to-all minimum power broadcast trees in wireless networks. Advances in Computing, Communications and Informatics. 74–79 (2014)
7. Blough, D.M., Leoncini, M., Resta, G., Santi, P.: On The Symmetric Range Assignment Problem In Wireless Ad Hoc Networks. Proceedings of the IFIP 17th World Computer Congress - TC1 Stream / 2Nd IFIP International Conference on Theoretical Computer Science: Foundations of Information Technology in the Era of Networking and Mobile Computing. 71–82 (2002)
8. Clementi, E.F., Penna, P., Silvestri, R.: On the power assignment problem in radio networks Mobile Networks and Applications - Discrete algorithms and methods for mobile computing and communications. 9, 2, 125–140 (2004)
9. Das, A. K., Marks, R. J., El-sharkawi, M.: Minimum power broadcast trees for wireless networks: Integer programming formulations. The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies. 245–248 (2003)
10. Das, A.K.; Marks, R.J., II; El-Sharkawi, M.; Arabshahi, P.; Gray, A.: Optimization methods for minimum power bidirectional topology construction in wireless networks with sectored antennas. Global Telecommunications Conference. 6, 3962–3968 (2004)
11. Das, A.K., Marks, R.J., El-sharkawi, M., Arabshahi, P., Gray, A.: r-Shrink: A heuristic for improving minimum power broadcast trees in wireless networks. Proceedings of the IEEE GLOBECOM'03. 1, 523–527 (2003)
12. Das, A.K., Marks, R.J., El-sharkawi, M., Arabshahi, P., Gray, A.: e-Merge : A heuristic for improving minimum power broadcast trees in wireless networks Technical Report, Department of Electrical Engineering, University of Washington. 2003.
13. Haugland, D., Yuan, D.: Wireless Network Design: Optimization Models and Solution Procedures. International Series in Operations Research & Management Science. New York, Springer. 219–246 (2011)
14. Liang, W.: Constructing Minimum-energy Broadcast Trees in Wireless Ad Hoc Networks. Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing. 112–122 (2002)
15. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. Annals of Discrete Mathematics. Elsevier Science. (1992)
16. Montemanni, R., Gambardella, L.M.: Exact Algorithms for the Minimum Power Symmetric Connectivity Problem in Wireless Networks. Computers and Operations Research. 32, 11, 2891–2904 (2005)
17. Papadimitriou, I., and Georgiadis, L.: Minimum-energy Broadcasting in Multi-hop Wireless Networks Using a Single Broadcast Tree. Mobile Networks and Applications. 11, 3, 361–375 (2006)
18. Wan, P., Clinescu, G., Yi C.: Minimum-Power Multicast Routing in Static Ad Hoc Wireless Networks. IEEE/ACM Transactions on Networking (TON). 12, 3, 507–514 (2004)
19. Wieselthier, J. E., Nguyen, G. D., Ephremides, A.: On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks. Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. 2, 585–594 (2000)
20. Yuan, D., Haugland, D.: Dual Decomposition for Computational Optimization of Minimum-Power Shared Broadcast Tree in Wireless Networks. IEEE Transactions on Mobile Computing. 12, 11, 2008–2019 (2012)
21. IBM ILOG CPLEX V12.1 User's Manual for CPLEX. 2009

Appendix A

Lemma 1. *Let* $(\mathbf{x}, \mathbf{z})$ *satisfy (3b) - (3i). A replacement of all directed arcs in* $G_{\mathbf{x}^s}$ *by undirected ones yields graph* $G_{\mathbf{z}}$*, for all* $s \in D$*.*

Proof. By constraint (3f), $\{i,j\} \in E_{\mathbf{z}}$ if and only if $(i,j) \in E_{\mathbf{x}^s}$ or $(j,i) \in E_{\mathbf{x}^s}$. The lemma then follows immediately. □

Lemma 2. *Let* $(\mathbf{x}, \mathbf{z})$ *satisfy (3b) - (3i) and* $s \in D$*. All arcs in a path* $(s = u_1, u_2, \ldots, u_k)$ *in digraph* $G_{\mathbf{x}^s}$ *are directed from* s *towards* u_k*.*

Proof. By (3g), we have $x^s_{u_2,s} = 0$, hence $x^s_{s,u_2} = 1$. Let us assume that there is a directed path from s to u_i. Then $x^s_{u_i,u_{i+1}} = 1$ holds, because if $x^s_{u_{i+1},u_i} = 1$, constraint (3c) would be violated (in case $u_i \in D$) or constraint (3d) would be violated (in case $u_i \in V \setminus D$), since we already assumed that $x^s_{u_{i-1},u_i} = 1$. In other words, the constraints (3c) - (3d) ensure that for any vertex there is no more than 1 inbound arc with x^s-value 1. The result then follows by induction on i. □

Proposition 1. *Let* $(\mathbf{x}, \mathbf{z})$ *satisfy constraints (3b) - (3i). If* Q *is a connected component in* $G_{\mathbf{z}}$ *such that* $V_Q \cap D \neq \emptyset$*, then,* Q *does not contain any cycle.*

Proof. By contradiction, let us assume that there exists a cycle $(c_1, c_2, \ldots, c_p = c_1)$ in Q. From Lemma 1, we have a corresponding directed subgraph C in $G_{\mathbf{x}^s}$ for any $s \in D$. Depending on the arc orientations, C is either a directed cycle or a directed acyclic graph.

If C is not a directed cycle, it contains at least one vertex c_i with $\deg^C_-(c_i) = 2$. This means that $x^s_{c_{i-1},c_i} = x^s_{c_{i+1},c_i} = 1$, which violates constraint (3c) if $c_i \in D$ or (3d) if $c_i \in V \setminus D$. Therefore, C is a directed cycle. As every $c_i \in C$ has $\deg^C_-(c_i) = \deg^C_+(c_i) = 1$, we have $x^s_{c_{i-1},c_i} = x^s_{c_i,c_{i+1}} = 1$, for all $s \in D$. Thus, C can not contain $s \in D$, because that would contradict constraint (3g), which says that the input degree of $s \in D$ in $G_{\mathbf{x}^s}$ is zero.

The remaining possibility is that C is a directed cycle containing only vertices in $V \setminus D$. As Q contains at least one destination $s \in D$, there is a path $(s = u_1, u_2, \ldots u_l)$ connecting s and C. Without loss of generality, let $c_1 = u_l$ be the vertex in C closest to s. By Lemma 2, we see that all edges of the path from s to c_1 in $G_{\mathbf{x}^s}$ are directed from s, which means that also $x^s_{u_{l-1},c_1} = 1$. However, that contradicts (3d), because we already assumed $x^s_{c_{p-1},c_1} = 1$ and $u_{l-1} \neq c_{p-1}$. It follows that the subgraph C is neither a cyclic graph nor a directed acyclic graph, which is a contradiction completing the proof. □

Proposition 2. *If* $(\mathbf{x}, \mathbf{z})$ *satisfies constraints (3b) - (3i), then there exists a path in* $G_{\mathbf{z}}$ *between any two destinations* $s, t \in D$*.*

Proof. Because (3g) implies that $\deg^{G_{\mathbf{x}^s}}_-(s) = 0$, the claim is equivalent to the following: Any maximal directed path in $G_{\mathbf{x}^s}$ to t starts in s. Assume the contrary, that the path starts in $j \neq s$. If $j \in D$, constraint (3c) ensures that there is one inbound arc (i,j), and hence the path can be extended by (i,j). Similarly, if

$j \in V \setminus D$, by (3d), if j has an outgoing arc (which is fulfilled by the assumption), there must be an inbound arc, which again contradicts the maximality of the path. According to Proposition 1, $G_{\mathbf{z}}$ is acyclic, and the proof is complete since the number of vertices in V is finite. □

Proposition 3. *If* $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ *is an optimal solution to (3a) - (3i), then one of the connected components of* $G_{\mathbf{z}}$ *is an optimal tree in Problem 1, and* $\sum_{(i,j)\in A} \sum_{s\in D} p_{ij} y^s_{ij} = P(G_{\mathbf{z}})$.

Proof. It follows from Prop. 2 that $G_{\mathbf{z}}$ has a connected component $d(G_{\mathbf{z}})$ spanning D, and from Prop. 1 that $d(G_{\mathbf{z}})$ is a tree. Also, it is obvious that for any tree T in G spanning D, there exists a feasible solution $(\mathbf{x}', \mathbf{y}', \mathbf{z}')$ to (3a) - (3i) such that $d(G_{\mathbf{z}'}) = T$. Consequently, we only need to show that for all $i \in V$,

$$\sum_{j\in V:(i,j)\in A} \sum_{s\in D} p_{ij} y^s_{ij} = c_{G_{\mathbf{z}}}(i) \tag{4}$$

as defined by (1).

Consider an arbitrary vertex i. If i is a leaf in $d(G_{\mathbf{z}})$, it must be a destination. Recall that i_1 denotes the most distant, in this case the only, neighbour of i in $G_{\mathbf{z}}$. Because of (3f) and (3g), we have $x^i_{ii_1} = 1$. Constraint (3h) then enforces $y^i_{ii_1} = 1$, which implies (4).

If i is an interior vertex, constraints (3c) and (3d) enforce that there is one inbound arc to i in $G_{\mathbf{x}^s}$. So, either $x^s_{i_1 i} = x^s_{ii_2} = 1$ or $x^s_{i_2 i} = x^s_{ii_1} = 1$. In the first case, (3h) assigns $y^s_{ii_2} = 1$. In the second case, $y^s_{ii_1} = 1$ holds. Due to minimization, all variables $y^s_{ii_j}$ with $j > 2$ are set to 0. As the first case corresponds to $s \in T_{i_1/i}$ and the second case to $s \in T \setminus T_{i_1/i}$, summing over all $s \in D$ yields (4). The contribution of i to (2) is then $nod(T_{i_1/i}) p_{ii_2} + nod(T \setminus T_{i_1/i}) p_{ii_1}$. □

By satisfying the constraints, we obtain a solution where each vertex is assigned powers necessary for relaying a message from a particular source. By optimality, the solution involves the power levels representing minimal SMT cost.

Paper II

6.2 The Shared Broadcast Tree Problem and MST

Marika Ivanova

Electronic Notes in Discrete Mathematics, vol 55, 2016

The Shared Broadcast Tree Problem and MST

Marika Ivanova [1]

Department of Infomatics,
University of Bergen,
Bergen, Norway

Abstract

The shared broadcast tree (SBT) problem in Euclidean graphs resembles the minimum spanning tree (MST) problem, but differs from MST in the definition of the objective function. The SBT problem is known to be NP-hard. In the current work, we analyse how closely the MST-solution approximates the SBT-solution, and we prove in particular that the approximation ratio is at least 6. Further, we conduct numerical experiments comparing the MST-solution and the optimum. The results show that the cost of the MST-solution is around 20% higher than the optimal cost.

Keywords: shared broadcast tree, MST, approximation algorithm

1 Introduction

The purpose of a broadcast communication in a wireless ad-hoc network is to route information from one source node to all other nodes. Given a set of devices and distances between them, the task is to assign a power to each node, so that the communication demands are met and the energy consumption is minimized, assuming their locations are fixed. The devices are able to both transmit and receive a signal, as well as dynamically adjust their power level.

[1] Email: marika.ivanova@uib.no

Omnidirectional antennas are used, and hence a message reaches all nodes within the communication range given by a power assigned to the sender, i.e. the maximum of the powers necessary to reach all intended recipients.

Minimum Energy Broadcast [3] (MEB) is the problem of constructing an optimal arborescence for broadcasting from a given source to all remaining nodes, such that the total power consumption is minimized. A separate tree has to be stored for each source. The idea of SBT [2][4] is to construct a common source-independent tree, instead of a set of individual arborescences. The power levels then depend merely on the immediate neighbour from which a message is received. This idea is based on the observation that a forwarded signal does not have to reach the neighbour from which it originally came.

The decentralized nature of wireless ad-hoc networks implies its suitability for applications, where it is not possible to rely on central nodes, or where network infrastructure does not exist. This is typical for various short-term events like conferences or fixtures. Simple maintenance makes them useful in emergency situations, military conflicts, and home networking.

We model a wireless network as a complete graph $G = (V, E)$, where V corresponds to the network nodes (points in $\mathbb{R}^2$), and the edges E correspond to the potential links between them. The energy requirement for transmission from i to j is denoted by $p_{ij} = \kappa d_{ij}^{\alpha}$, where d_{ij} is the Euclidean distance between i and j, α is an environment-dependent parameter (typically valued between 2 and 4) and κ is a constant. In this work, we use $\alpha = 2$ and $\kappa = 1$. Let $T = (V, E_T)$, $E_T \subseteq E$ be a spanning tree of G. Then $T_{i/j}$ denotes the subtree of T consisting of all vertices k such that the path from k to j visits i, as introduced in [4]. For a non-leaf node i in T, i_1 and i_2 denote the first and the second most distant neighbour of i in T, respectively. If i is a leaf, i_2 is not defined, and we let $p_{ii_2} = 0$. If a message is generated at a node k in $T_{i_1/i}$ then i needs power p_{ii_2} to relay the message to i_2 and other neighbours in $T \setminus T_{i_1/i}$. Power p_{ii_1} is needed to relay messages initiated in $T \setminus T_{i_1/i}$. Assuming that all nodes initiate messages equally frequently, the SBT problem is to construct a spanning tree T minimizing the objective function

$$P(T) = \sum_{i \in V} |T_{i_1/i}| p_{ii_2} + |T \setminus T_{i_1/i}| p_{ii_1}. \tag{1}$$

2 MST as an approximate solution to the SBT problem

Since the SBT problem is NP-hard, inexact solutions are often considered. Because any spanning tree is a feasible solution, the MST-solution yields one

such approximation. This approach is also valid for MEB, where MST approximates the optimum with factor 6 [1]. We define the *MST approximation ratio* ρ as the supremum, taken over all SBT instances, of the ratio between the power consumptions in the MST solution and an optimal SBT.

Theorem 2.1 *The MST approximation ratio for SBT is at least 6.*

Proof. For an integer $k \geq 2$, let G_k be a complete Euclidean graph with a node o located in the center of a unit circle, nodes $t_1, \ldots, t_6$ evenly distributed on the circumference, and nodes $s_{i1}, \ldots, s_{ik}$, $(i = 1, \ldots, 6)$, evenly distributed on the radial line $[o, s_{ik}] \subset [o, t_i]$, where s_{ik} is located $1/k$ units from o. Thus, since arc costs p_{uv} are the square of arc lengths d_{uv}, we have $p_{uv} = 1/k^4$ for $u = s_{ij}$, $v = s_{i,j+1}$, whereas $p_{uv} = (1 - 1/k)^2$ for $u = s_{ik}$, $v = t_i$. A possible MST (denoted T_k) of G_k consists of the 6 paths $(o, s_{i1}, \ldots, s_{ik}, t_i)$. For this tree, the objective function (1) evaluates to

$$P(T_k) = \underbrace{6\left(1 - k^{-1}\right)^2}_{t_i} + \underbrace{6\left[(6k+6)\left(1-k^{-1}\right)^2 + k^{-4}\right]}_{s_{ik}} + \underbrace{(6k-5)(6k+7)k^{-4}}_{o, s_{i1}, \ldots, s_{i,k-1}}.$$

Another spanning tree of G_k is the star T_k^* centered at node o. For this solution, (1) evaluates to

$$P(T_k^*) = \underbrace{6}_{t_i} + \underbrace{6\sum_{i=1}^{k}\left(i\frac{1}{k^2}\right)^2}_{s_{i1}, \ldots, s_{ik}} + \underbrace{6k+7}_{o}.$$

Thus, the MST-approximation ratio satisfies $\rho \geq \frac{P(T_k)}{P(T_k^*)}$. Since $\lim_{k\to\infty} \frac{P(T_k)}{P(T_k^*)} = 6$, the claim follows. □

3 Numerical Experiments

The SBT problem can be modelled as a MILP [2][4], and moderately sized instances can be solved. We have generated instances of a specific number of nodes with random coordinates distributed uniformly on a square, and compared the MST-solution to the optimal one. The MILP solver CPLEX is used to compute the optimal solution. Each number of nodes is tested in 100 instances. Although the theoretical approximation ratio suggests that MST is not very suitable for SBT, the experimental results summarized in Tab. 1 reveal that in practice, MST represents a feasible solution with objective value approximately 1.2 times the optimum. This factor does not seem to change much with growing number of nodes. However, calculation of the optimum

Table 1
Average SBT costs of MST and optimal solutions for various instance sizes.

Number of nodes	10	12	14	16	18	20
P(OPT)	46268	56060	66747	69727	84250	94039
P(MST)	9432	68833	80195	84262	101816	119679
P(MST)/P(OPT)	1.198	1.232	1.206	1.210	1.209	1.271

for larger instances takes prohibitively long time, so we have access only to limited data. The largest ratio observed in the experiments is 1.59.

4 Conclusion and Future Work

This paper studies the relation between MST and the optimal solution to SBT in terms of the objective value. It has been shown that the MST approximation ratio is at least 6. Numerical experiments suggest that even though there are instances where MST is nearly 60% above the optimum, it represents a good solution in the vast majority of cases. The current research leads to several interesting questions that merit further investigation. A prominent question is whether there exists a constant upper bound on the MST-approximation ratio. For the related MEB problem, approximation algorithms with constant performance guarantee are well studied. Adapting these methods and the corresponding analysis to SBT is a research question to be pursued.

References

[1] Ambühl, C.: An Optimal Bound for the MST Algorithm to Compute Energy Efficient Broadcast Trees in Wireless Networks. Automata, Languages and Programming. 32nd International Colloquium, ICALP 2005, Lisbon, Portugal. 1139–1150 (2005)

[2] Papadimitriou, I., and Georgiadis, L.: Minimum-energy Broadcasting in Multi-hop Wireless Networks Using a Single Broadcast Tree. Mobile Networks and Applications, 11, 3 361–375 (2006)

[3] Wieselthier, J. E., Nguyen, G. D., Ephremides, A.: Energy-efficient broadcast and multicast trees in wireless networks. Mob. Netw. Appl. 7, 6, 481-492 (2002)

[4] Yuan, D., Haugland, D.: Dual Decomposition for Computational Optimization of Minimum-Power Shared Broadcast Tree in Wireless Networks. IEEE Transactions on Mobile Computing, Vol 12, no 11. 2008–2019 (2012)

Paper V

6.5 Area Protection in Adversarial Path-finding Scenarios with Multiple Mobile Agents on Graphs

Marika Ivanova, Pavel Surynek, Katsutoshi Hirayama

In: *Proceedings of the 10th International Conference on Agents and Artificial Intelligence* (ICAART 2018), pp. 184-191, Funchal, Madeira, Portugal, SciTe Press, 2018, ISBN 978-989-758-275-2.

Area Protection in Adversarial Path-finding Scenarios with Multiple Mobile Agents on Graphs

A Theoretical and Experimental Study of Strategies for Defense Coordination

Marika Ivanová[1], Pavel Surynek[2] and Katsutoshi Hirayama[3]
[1]*Department of Informatics, University of Bergen, Thormhlensgt. 55, 5020 Bergen, Norway*
[2]*National Institute of Advanced Industrial Science and Technology (AIST), 2-3-26, Aomi, Koto-ku, Tokyo 135-0064, Japan*
[3]*Kobe University, 5-1-1, Fukaeminami-machi, Higashinada-ku, Kobe 658-0022, Japan*
marika.ivanova@uib.no, pavel.surynek@aist.go.jp, hirayama@maritime.kobe-u.ac.jp

Keywords: Graph-based Path-finding, Area Protection, Area Invasion, Asymmetric Goals, Mobile Agents, Agent Navigation, Defensive Strategies, Adversarial Planning.

Abstract: We address a problem of area protection in graph-based scenarios with multiple agents. The problem consists of two adversarial teams of agents that move in an undirected graph. Agents are placed in vertices of the graph and they can move into adjacent vertices in a conflict-free way in an indented environment. The aim of one team - *attackers* - is to invade into a given area while the aim of the opponent team - *defenders* - is to protect the area from being entered by attackers. We study strategies for assigning vertices to be occupied by the team of defenders in order to block attacking agents. We show that the decision version of the problem of area protection is PSPACE-hard. Further, we develop various on-line vertex-allocation strategies for the defender team and evaluate their performance in multiple benchmarks. Our most advanced method tries to capture bottlenecks in the graph that are frequently used by the attackers during their movement. The performed experimental evaluation suggests that this method often defends the area successfully even in instances where the attackers significantly outnumber the defenders.

1 INTRODUCTION

We address an *Area Protection Problem* (APP) with multiple mobile agents moving in a conflict-free way. APP can be regarded as a modification of known problem of *Adversarial Cooperative Path Finding* (ACPF) (Ivanová and Surynek, 2014) where two teams of agents compete in reaching their target positions. Unlike ACPF, where the goals of teams of agents are symmetric, the adversarial teams in APP have different objectives. The first team of *attackers* consists of agents whose goal is to reach a pre-defined target location in the area being protected by the second team of *defenders*. Each attacker has a unique target in the protected area. The opponent team of defenders tries to prevent the attackers from reaching their targets by occupying selected locations so that they cannot be passed by attackers.

Another distinction between ACPF and APP is a definition of victory of a team. A team in ACPF wins if all its agents reach their targets and agents of no other team manage to do so earlier. In APP, the team of defenders wins if all attackers are kept out of their targets. Our effort is to design a strategy for the defending team, so the success is measured from the defenders' perspective. It is often not possible to prevent all attackers from reaching their targets, and so the following objective functions can be pursued:

1. maximize the number of target locations that are not captured by the corresponding attacker
2. maximize the number of target locations that are not captured by the corresponding attacker within a given time limit
3. maximize the sum of distances between the attackers and their corresponding targets
4. minimize the time spent at captured targets

The common feature of APP and ACPF is that once a location is occupied by an agent, it cannot be entered by another agent until it is first vacated by the agent which occupies it (opposing agent cannot push the agent out). This is utilized both in competition for reaching goals in ACPF, where agents may try to slow down the opponent by occupying certain locations, as

Ivanová, M., Surynek, P. and Hirayama, K.
Area Protection in Adversarial Path-finding Scenarios with Multiple Mobile Agents on Graphs - A Theoretical and Experimental Study of Strategies for Defense Coordination.
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 1*, pages 184-191
ISBN: 978-989-758-275-2

well as in APP, where it is a key tool for the defenders for keeping attackers out of the protected area.

APP has many real-life motivations from the domains of access denial operations both in civil and military sector, robotics with adversarial teams of robots or other type of penetrators (Agmon et al., 2011), and computer games.

Our contribution consists in analysis of computational complexity of APP. In particular, we show that APP is PSPACE-hard. Next, we suggest several online solving algorithms for the defender team that allocate selected vertices to be occupied so that attacker agents cannot pass into the protected area. We identify suitable vertex allocation strategies for diverse types of APP instances and test them thoroughly.

1.1 Related Work

Movements of agents at low reactive level are assumed to be planned by some *cooperative path-finding - CPF* (*multi-agent path-finding* - MAPF) (Silver, 2005; Ryan, 2008; Wang and Botea, 2011) algorithm where agents of own team cooperate while opposing agents are considered as obstacles. In CPF the task is to plan movement of agents so that each agent reaches its unique target in a conflict free manner.

There exist multiple CPF algorithms both complete and incomplete as well as optimal and suboptimal under various objective functions.

A good trade-off between the quality of solutions and the speed of solving is represented by suboptimal/incomplete search based methods which are derived from the standard $A*$ algorithm. These methods include *LRA**, *CA**, *HCA**, and *WHCA** (Silver, 2005). They provide solutions where individual paths of agents tend to be close to respective shortest paths connecting agents' locations and their targets. Conflict avoidance among agents is implemented via a so called reservation table in case of *CA**, *HCA**, and *WHCA** while *LRA** relies on replanning whenever a conflict occurs. Since our setting in APP is inherently suitable for a replanning, the algorithm *LRA** is a candidate for underlying CPF algorithm for APP. Moreover, *LRA** is scalable for large number of agents.

Aside from CPF algorithms, systems with mobile agents that act in the adversarial manner represent another related area. These studies often focus on patrolling strategies that are robust with respect to various attackers trying to penetrate through the patrol path (Elmaliach et al., 2009).

Theoretical or empirical works related to APP also include studies on pursuit evasion (Hespanha et al., 1999; Vidal et al., 2002) or predator-prey (Benda et al., 1986; Haynes and Sen, 1995) problems. The Tileworld (Pollack and Ringuette, 1990) also provides an experimental environment to evaluate planning and scheduling algorithms for a team of agents. A major difference between these works and the concept of APP is that, unlike the previous works, we assume the agents in each team perform CPF algorithms, which provide a new foundation of team architecture.

1.2 Preliminaries

The environment is modeled by an undirected unweighted graph $G = (V,E)$. We restrict the instances to 4-connected grid graphs with possible obstacles. The team of attackers and defenders is denoted by $A = \{a_1,\ldots,a_m\}$ and $D = \{d_1,\ldots d_n\}$, respectively. Continuous time is divided into discrete time steps. Agents are placed in vertices of the graph at each time step so that at most one agent is placed in each vertex. Let $\alpha_t : A \cup D \to V$ be a uniquely invertible mapping denoting configuration of agents at time step t. Agents can wait or move instantaneously into adjacent vertex between successive time steps to form the next configuration α_{t+1}. Abiding by the following movement rules ensures preventing conflicts:

- An agent can move to an adjacent vertex only if the vertex is empty, or is being left at the same time step by another agent
- A pair of agents cannot swap along a shared edge
- No two agents enter the same adjacent vertex at the same time

We do not assume any specific order in which agents perform their conflict free actions at each time step. Our experimental implementation moves all attackers prior to moving all defenders at each time step. The mapping $\delta^A : A \to V$ assigns a unique target to each attacker. The task in APP is to find a strategy of movement for defender agents so that the area specified by δ^A is protected.

We state APP as a decision problem as follows:

Definition 1. *The decision APP problem: Given an instance* $\Sigma = (G,A,D,\alpha_0,\delta^A)$ *of APP, is there a strategy of movement for the team D of defenders, so that agents from the team A of attackers are prevented from reaching their targets defined by* δ^A.

In many instances it is not possible to protect all targets. We are therefore also interested in the optimization variant of the APP problem:

Definition 2. *The optimization APP problem: Given an instance* $\Sigma = (G,A,D,\alpha_0,\delta^A)$ *of APP, the task is to find a strategy of movement for the team D of defenders such that the number of attackers in A that reach their target defined by* δ^A *is minimized.*

ICAART 2018 - 10th International Conference on Agents and Artificial Intelligence

2 THEORETICAL PROPERTIES

APP is a computationally challenging problem as shown in the following analysis. In order to study theoretical complexity of APP, we need to consider the decision variant of APP. Many game-like problems are PSPACE-hard, and APP is not an exception. We reduce the known problem of checking validity of Quantified Boolean Formula (QBF) to it.

The technique of reduction of QBF to APP is inspired by a similar reduction of QBF to ACPF, from which we borrow several technical steps and lemmas (Ivanová and Surynek, 2014). We describe the reduction from QBF using the following example. Consider a QBF formula in prenex normal form

$$\varphi = \exists x \forall a \exists y \forall b \exists z \forall c \\ (b \vee c \vee x) \wedge (\neg a \vee \neg b \vee y) \wedge \\ (a \vee \neg x \vee z) \wedge (\neg c \vee \neg y \vee \neg z) \quad (1)$$

This formula is reduced to the APP instance depicted in Fig. 4. Let n and m be the number of variables and clauses, respectively. The construction contains three types of graph gadgets.

For an **existentially** quantified variable x we construct a diamond-shape gadget consisting of two parallel paths of length $m+2$ joining at its two endpoints.

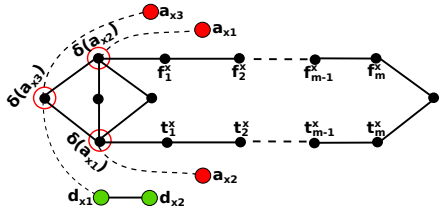


Figure 1: An existentially quantified variable gadget.

There are 4 paths connected to the diamond at specific vertices as depicted in Fig. 1. The gadget further contains three attackers and two defenders with initial positions at the endpoints of the four joining paths. The vertices in red circles are targets of specified attackers. The only chance for defenders d_{x1} and d_{x2} to prevent attackers a_{x3} and a_{x1} from reaching their targets is to advance towards the diamond and occupy $\delta_A(a_{x3})$ by d_{x2} and either $\delta_A(a_{x1})$ or $\delta_A(a_{x2})$ by d_{x1}.

For every **universally** quantified variable a there is a similar gadget with a defender d_{a1} and an attacker a_{a1} whose target $\delta^4(a_{a1})$ lies at the leftmost vertex of the diamond structure (see Fig. 2). The defender has to rush to the attacker's target and occupy it, because otherwise the target would be captured by the attacker.

Moreover, there is a gadget in two parts for each **clause** C depicted in Fig. 3. It contains a simple path

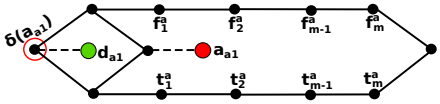


Figure 2: A universally quantified variable gadget.

p of length $\lfloor n/2 \rfloor + 1$ with a defender d_C placed at one endpoint. The length of p is chosen in order to ensure a correct time of d_C's entering to a variable gadget, so that gradual assignment of truth values is simulated. E.g., if a variable occurring in C stands in the second $\forall\exists$ pair of variables in the prefix (the first and last pair is incomplete), then p is connected to the corresponding variable gadget at its second vertex. The second part of the clause is a path of length k, with one endpoint occupied by attacker a_C whose target $\delta^4(a_C)$ is located at the other endpoint. The length k is selected in a way that the target $\delta^4(a_C)$ can be protected if the defender d_C arrives there on time, which can happen only if it uses the shortest path to this target. If d_C is delayed by even one step, the attacker a_C can capture its target. These two parts of the clause gadget are connected through variable gadgets.

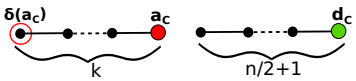


Figure 3: A clause gadget.

The connection by edges and paths between variable and clause gadgets is designed in a way that allows the agents to synchronously enter one of the paths of the relevant variable gadget. A gradual evaluation of variables according to their order in the prefix corresponds to the alternating movement of agents. A defender d_C from clause C moves along the path of its gadget, and every time it has the opportunity to enter some variable gadget, the corresponding variable is already evaluated.

If there is a literal in φ that occurs in multiple clauses, setting its value to true causes satisfaction of all the clauses containing it. This is indicated by a simultaneous entering of affected agents to the relevant path. Each clause defender d_C has its own vertex in each gadget of a variable present in C, at which d_C can enter the gadget. This allows a collision-free entering of multiple defenders into one path of the gadget.

Theorem 1. *The decision problem whether there exists a winning strategy for the team of defenders, i.e. whether it is possible to prevent all attackers from reaching their targets in a given APP instance is PSPACE-hard.*

Proof. Suppose φ to be valid. To better understand validity of φ we can intuitively ensure that variables

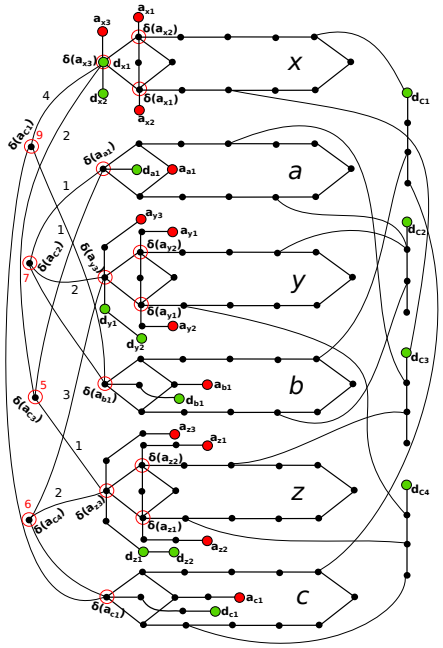


Figure 4: A reduction from TQBF to APP. Black points represent unoccupied vertices. If two points are connected by a line without any label, it means there is an edge between them. A line with a label k indicates that the two points are connected by a path of k internal vertices. Initial positions of attackers and defenders are represented by red and green nodes, respectively. A red circle around a node means that the node is a target of some attacker. For simplicity we do not fully display the second part of the clause gadget. Instead, there is a red number near the target of a clause gadget that indicates the distance of the attacker aiming to that target. A vertex with an agent is labeled by the agent's name. Labels of targets specify the associated agents.

are assigned gradually according to their order in the prefix. For every choice of value of the next $\forall$-variable there exists a choice of value for the corresponding $\exists$-variable so that eventually the last assignment finishes a satisfying valuation of φ. The strategy of assigning $\exists$ variables can be mapped to a winning strategy for defenders in the APP instance constructed from φ. Every satisfying valuation guides the defenders towards vertices resulting in a position where all targets are defended. Every time a variable is valuated, another agent in the constructed APP instance is ready to enter the upper path, if the variable is evaluated as true, or the lower path, otherwise. Note that the vertices on the paths are labeled as t and f indicating the truth value that is simulated by passing through a path. When the evaluated variable x is existentially quantified, the defender d_{x1} enters the upper or lower path. In case of universally quantified variable a, the entering agent is the attacker a_{a1}. Since the valuation satisfies φ, every clause C_j has at least one variable q causing the satisfaction of C_j. That is modeled by the situation where defenders d_{q1} and d_{Cj} meet each other in one of the diamond's paths, which enables either the defender d_{q2} (in case q is existentially quantified) or d_{q1} (in case q is universally quantified) to advance towards the target $\delta^A(d_{C1})$. The situation for an existentially quantified variable is explained by Fig. 5.

Whenever there exists a winning strategy for the constructed APP instance, the defenders must arrive in all targets on time. This is possible only if variable defenders and clause defenders meet on one of the paths in a diamond gadget, and only if all defenders use the shortest possible paths. The variable agents' selection of upper or lower paths determines the evaluation of corresponding variables. An advancement of variable and clause defenders that leads to meeting of the defenders at adjacent vertices, and a subsequent protection of targets indicates that the corresponding variable causes satisfaction of the clause. □

3 DESTINATION ALLOCATION

Solving APP in practice is a challenging problem due to its high computational complexity. Our solving approaches are based on a technique called *destination allocation*. The basic idea is to assign a destination vertex to each defender and subsequently use some CPF algorithm modified for the environment with adversaries to lead each defender to its destination. A defender may be allocated to any vertex, including the attackers' targets. Destination allocation can be divided into two basic categories: *single-stage*, where agents are allocated to destinations only once at the beginning, and *multi-stage*, where destinations can be reassigned any time during the agents' course. This work focuses merely on the single-stage destination allocation and uses the LRA* algorithm for controlling agents' movement.

The defenders are initially not allocated to any destination and do not have any information about the intended target of any attacker. However, the defenders have a full knowledge of all target locations in the protected area. The task in this setting is to allocate each defender agent to some location in the graph, so that via its occupation, defenders try to optimize a given objective function.

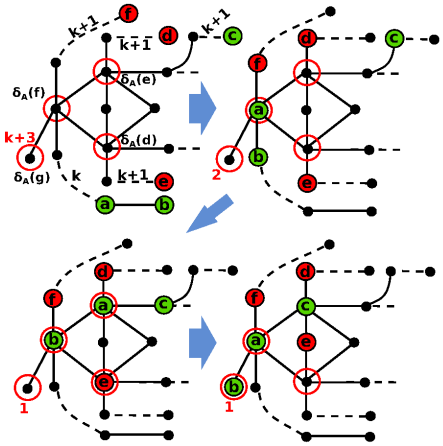


Figure 5: An example of agents' advancement at an existential variable clause. It is defenders' turn in each of the four figures. The top left figure shows the initial position of the agents. The value k depends on the order of the corresponding variable in the prefix. When agents reach the positions in the second figure, the corresponding variable is about to be evaluated which is analogous to defender a entering one of the paths, which prevents the attackers d and e from exchanging their position and reaching the targets. If a moves to the upper path, as happens in the third figure, the agent c from a clause gadget has the opportunity to enter the upper path where the two defenders meet. Attacker e can enter the target $\delta^A(d)$, which is nevertheless not its intended goal. Finally, the defenders can protect all targets by a train-like movement resulting in the position in the last figure. Also note the gradual approaching of the undisplayed attacker g to its target $\delta^A(g)$. The distance between g's current location and the target is indicated by the red number.

We describe several simple destination allocation strategies and discuss their properties. The first two methods always allocate one defender to some attacker's target. The advantage of this approach is that if a defender manages to capture a target, it will never be taken by the attacker. This can be useful in scenarios where the number of defenders is similar to the number of attackers. Unfortunately, such a strategy would not be very successful in instances where the attackers significantly outnumber the defenders. This issue is addressed in the last strategy, in which we attempt to exploit the obstacle structure and succeed even with a smaller number of defenders.

3.1 Random Allocation

For the sake of comparison, we consider the simplest strategy, where each defender agent is allocated to a random target of an attacker agent. Neither the locations of agents nor the underlying grid graph structure is exploited.

3.2 Greedy Allocation

The greedy strategy is a slightly improved approach. Defenders are one by one in an arbitrary order allocated to the closest available target of an attacker.

3.3 Bottleneck Simulation Allocation

Simple target allocation strategies do not exploit the structure of underlying graph. Hence, a natural next step is to occupy by defenders those vertices that would divert attackers from the protected area as much as possible with the help of graph structure. The aim is to successfully defend the targets even with small number of defenders. As our domains are 4-connected grids with obstacles, we can take advantage of the obstacles already occurring in the grid and use them as addition to vertices occupied by defenders. Figure 6 illustrates a grid where the defenders could easily protect the target area even though they are outnumbered by the attackers. Intuitively as seen in the example, hard to overcome obstacle for attacking team would arise if a bottleneck on expected trajectories of attackers is blocked.

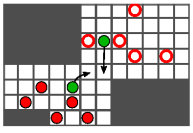


Figure 6: An example of bottleneck blocking. Solid red and green circles represent attackers and defenders, respectively. Empty red circles are the attackers' target locations.

In order to discover bottlenecks of a general shape, we develop the following simulation strategy. This method is based on the assumption that as attackers move towards the targets, vertices close to bottlenecks are entered by the attackers more often than other vertices. This observation suggests to simulate the movement of attackers and find frequently visited vertices. As defenders do not share the knowledge about paths being followed by attackers, frequently visited vertices are determined by a simulation in which paths of attackers are estimated.

There can be several vertices with the highest frequency of visits, so the final vertex is selected by another criterion. The closer a vertex is to the defenders, the better chance the defenders have to capture it before the attackers pass through it. On the grounds of

that, our implementation selects the vertex of maximum frequency with the shortest distance to an approximate location of defenders.

After obtaining such a frequently visited vertex, we then search its vicinity. If we find out that there is indeed a bottleneck, its vertices are assigned to some defenders as their destinations. Under the assumption that the bottleneck is blocked by defenders, the paths of attackers may substantially change. For that reason we estimate the paths again and find the next frequent vertex of which vicinity is also explored and so on. The whole process is repeated until all available defenders are allocated to a destination, or until no more bottlenecks are found. The high-level description of this procedure is expressed by Algorithm 1. The input of the algorithm is the graph G and sets

Data: $G = (V, E)$, D, A
Result: Destination allocation δ^D
$D_{\text{available}} = D$; // Defenders to be allocated
$F = \emptyset$; // Set of forbidden locations
$\delta'_A =$ Random guess of δ_A;
while $D_{available} \neq \emptyset$ **do**
  for $a \in A$ **do**
    $p_a = \text{shortestPath}(\alpha_0(a), \delta'_A(a), G, F)$;
  end
  $f(v) = |\{p_a : a \in A \wedge v \in p_a\}|$
  $w \in \arg\max_{v \in V} f(v)$;
  $B = \text{exploreVicinity}(w)$;
  if $B \neq \emptyset \vee |D| < |B|$ **then**
    D' such that $D' \subseteq D_{\text{available}}, |D'| = |B|$;
    assignToDefenders(B, D');
    $D_{available} = D_{\text{available}} \setminus D'$;
    $F = F \cup B$
  else
    break ;
  end
end
assignToRandomTargets($D_{available}$);

Algorithm 1: Bottleneck simulation procedure.

D and A of defenders and attackers, respectively. During the initialization phase, we create the set $D_{available}$ of defenders that are not yet allocated to any destination. Next, we create the set F of so called forbidden nodes. The following step takes attackers one by one and every time makes a random guess which target is an attacker aiming for, resulting in the mapping δ'. The algorithm then iterates while there are available defenders. In each iteration, we construct a shortest path from each attacker a between its initial position $\alpha_0(a)$ and its estimated target location $\delta'(a)$. A vertex w from among the vertices contained in the highest number of paths is then selected, and its surroundings is searched for bottlenecks. If a bottleneck is found, the set of vertices B is determined in order to block the bottleneck. The set D' contains a sufficient number of available defenders that are subsequently allocated to the vertices in B. Agents from D' are no longer available, and vertices from B are now forbidden. The pathfinding in the following iterations will therefore avoid vertices in B. If no bottleneck is found, it is likely that agents have a lot of freedom for movement, and blocking bottlenecks is not a suitable strategy. The loop is left and the remaining available agents are assigned to random targets from $T_{available}$.

The search of the close vicinity of a frequently used vertex w is carried out by an expanding square centered at w. We start with distance 1 from w and gradually increase this value[1] up to a certain limit. In every iteration we identify the obstacles in the fringe of the square and keep them together with obstacles discovered in previous iterations. Then we check whether the set of obstacles discovered so far forms more than one connected components. If that is the case, it is likely that we encountered a bottleneck. We then find the shortest path between one connected component of obstacles and the remaining components. This shortest path is believed to be a bottleneck in the map, and its vertices are assigned to the available defenders as their new destinations.

In order to discover subsequent bottlenecks in the map, we assume that the previously found bottlenecks are no longer passable. They are marked as forbidden and in the next iteration, the estimated paths will not pass through them. The procedure *shortestPath* returns the shortest path between given source and target, that does not contain any vertices from the set F of forbidden locations.

In this basic form, the algorithm is prone to finding "false" bottlenecks in instances with an indented map that contains for example blind alleys. It is possible to avoid undesired assigning vertices of false bottlenecks to defenders by running another simulation which excludes these vertices. If the updated paths are unchanged from the previously found ones, it means that blocking of the presumed bottleneck does not affect the attackers movement towards the targets, and so there is no reason to block such a bottleneck.

4 EXPERIMENTAL EVALUATION

Experimental evaluation is focused on competitive comparison of suggested destination allocation strategies with respect to the objective 2. - maximization

[1]Two locations are considered to be in distance 1 from each other if they share at least one point. Hence, a location that does not lie on the edge of the map has 8 neighbors.

of the number of locations not captured by attackers within a given time limit.

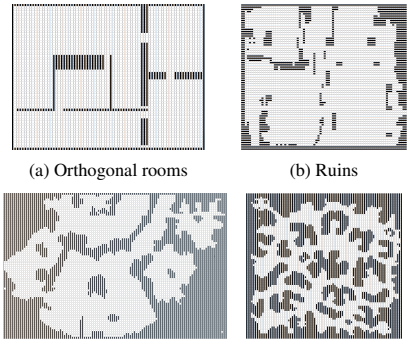
Our hypothesis is that the random strategy would perform as worst since it is completely uninformed. All the simple strategies are expected to be outperformed by more advanced bottleneck simulation.

We implemented all suggested strategies in Java as an experimental prototype. Our testing scenarios use maps of various structures and initial configurations of agents. Our choice of testing scenarios is focused on comparing studied strategies and discovering what factors have a significant impact on their success.

As the following sections show, different strategies are successful in different types of instances. It is therefore important to design the instances with a sufficient diversity, in order to capture strengths and weaknesses of individual strategies.

4.1 Instance Generation and Types

The instances used in the practical experiments are generated using a pseudo random generator, but in a controlled manner. An instance is defined by its map, the ratio $|D| : |A|$ and locations of individual defenders, attackers and their targets. These entries form an input of the instance generation procedure. Further, we select rectangular areas inside which agents of both teams and the attackers' targets are placed randomly. We use 4 maps with increasingly complicated obstacle structure depicted in Fig. 7. Each map size is in the order of thousands of vertices.



(a) Orthogonal rooms (b) Ruins

(c) Waterfront (d) Dark forest

Figure 7: Maps.

In the main set of experiments, each map is populated with agents of 3 different $|D| : |A|$ ratios, namely $1 : 1$, $1 : 2$ and and $1 : 10$, with fixed number of attackers $|A| = 100$. Each of these scenarios is further divided into two types reflecting a relative positions of attackers and defenders. The type *overlap* assumes that the rectangular areas for both teams have an identical location on the map, while the teams in the type *separated* have distinct initial areas. The maximum number of agents' moves is set to 150 for each team.

Note that the individual instances are never completely fair to both teams. It is therefore impossible to make a conclusion about a success rate of a strategy by comparing its performance on different maps. The comparison should always be made by inspecting the performance in one type of instance, where we can see the relative strength of the studied algorithms.

4.2 Results

The performed experiments compare random, greedy, and simulation strategy in different instance settings. Each entry in Table 1 is an average number of attackers that reached their targets at the end of the time limit. The average value is calculated for 10 runs in each settings, always with a different random seed. Random and greedy strategies have very similar results in all positions and team ratios. It is apparent and not surprising that with decreasing $|D| : |A|$ ratio, the strength of these strategies decreases. The simulation strategy gives substantially better results in all settings. Also note that in case of overlapping teams, the simulation strategy scores similarly in all $|D| : |A|$ ratios.

Table 1: Average number of agents that eventually reached their target in the map Orthogonal rooms.

Team position	$\|D\| : \|A\|$	RND	GRD	SIM
Overlapped	1:1	40.4	49.2	21.0
	1:2	56.7	56.5	20.8
	1:10	67.8	64.7	24.7
Separated	1:1	39.0	40.7	10.3
	1:2	57.7	50.1	13.3
	1:10	78.5	69.9	30.2

Table 2 contains results of an analogous experiment conducted on the map Ruins. The random strategy performs well in instances with many attackers. The dominance of the simulation strategy is apparent here as well.

Maps Waterfront and Dark forest contain very irregular obstacles and many bottlenecks, and are therefore very challenging environments for all strategies. In the Dark forest map, random and greedy methods are more suitable than the simulation strategy in instances with equal team sizes, as oppose to the scenarios with lower number of defenders, where the bottleneck simulation strategy clearly leads. In the separated scenario, the simulation strategy is even worse

Table 2: Average number of agents that eventually reached their target in the map Ruins.

Team position	$\|D\| : \|A\|$	RND	GRD	SIM
Overlapped	1:1	36.8	49.4	17.7
	1:2	80.0	63.5	33.0
	1:10	92.5	88.9	58.2
Separated	1:1	9.5	33.6	11.8
	1:2	47.6	34.4	11.8
	1:10	85.6	85.9	14.7

in all tested ratios (see Tab. 3 and Tab. 4). This behavior can be explained by the fact that occupying all relevant bottlenecks in such a complex map is harder than occupying targets in the protected area. In contrast, bottlenecks in the Waterfront map have more favourable structure, so that those relevant for the area protection can be occupied more easily.

Table 3: Average number of agents that eventually reached their target in the map Waterfront.

Team position	$\|D\| : \|A\|$	RND	GRD	SIM
Overlapped	1:1	32.0	41.7	37.1
	1:2	60.6	63.8	39.8
	1:10	77.8	72.9	51.7
Separated	1:1	15.8	19.3	10.7
	1:2	46.4	37.6	9.8
	1:10	75.3	65.5	14.9

Table 4: Average number of agents that eventually reached their target in the map Dark forest

Team position	$\|D\| : \|A\|$	RND	GRD	SIM
Overlapped	1:1	21.6	37.9	48.8
	1:2	53.7	42.6	37.8
	1:10	60.9	51.9	38.4
Separated	1:1	35.3	35.9	61.5
	1:2	40.6	41.3	59.6
	1:10	65.1	67.0	66.0

5 CONCLUDING REMARKS

We have shown the lower bound for computational complexity of the APP problem, namely that it is PSPACE-hard. Theoretical study of ACPF (Ivanová and Surynek, 2014) showing its membership in EXPTIME suggests that the same upper bound holds for APP but it is still an open question if APP is in PSPACE. In addition to complexity study we designed several practical algorithms for APP under the assumption of single-stage vertex allocation. Performed experimental evaluation indicates that our *bottleneck simulation* algorithm is strong even in case when defenders are outnumbered by attacking agents. Surprisingly, our simple *random* and *greedy* algorithms turned out to successfully block attacking agents provided there are enough defenders.

For future work we plan to design and evaluate algorithms under the assumption of multi-stage vertex allocation. As presented algorithms have multiple parameters we also aim on their optimization. Another generalization motivated by practical applications in robotics is APP with communication maintenance.

REFERENCES

Agmon, N., Kaminka, G. A., and Kraus, S. (2011). Multi-robot adversarial patrolling: Facing a full-knowledge opponent. *J. Artif. Intell. Res.*, 42:887–916.

Benda, M., Jagannathan, V., and Dodhiawala, R. (1986). On optimal cooperation of knowledge sources - an empirical investigation. Technical Report BCS–G2010–28, Boeing Advanced Technology Center.

Elmaliach, Y., Agmon, N., and Kaminka, G. A. (2009). Multi-robot area patrol under frequency constraints. *Ann. Math. Artif. Intell.*, 57(3-4):293–320.

Haynes, T. and Sen, S. (1995). Evolving behavioral strategies in predators and prey. In *Proc. of Adaption and Learning in Multi-Agent Systems, IJCAI'95 Workshop*, pages 113–126.

Hespanha, J. P., Kim, H. J., and Sastry, S. (1999). Multiple-agent probabilistic pursuit-evasion games. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, volume 3, pages 2432–2437 vol.3.

Ivanová, M. and Surynek, P. (2014). Adversarial cooperative path-finding: Complexity and algorithms. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014*, pages 75–82.

Pollack, M. E. and Ringuette, M. (1990). Introducing the tileworld: Experimentally evaluating agent architectures. In *Proc. of the 8th National Conference on Artificial Intelligence*, pages 183–189. AAAI Press.

Ryan, M. R. K. (2008). Exploiting subgraph structure in multi-robot path planning. *J. Artif. Intell. Res.*, 31:497–542.

Silver, D. (2005). Cooperative pathfinding. In *Proc. of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, 2005*, pages 117–122.

Vidal, R., Shakernia, O., Kim, H. J., Shim, D. H., and Sastry, S. (2002). Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Trans. Robotics and Autom.*, 18(5):662–669.

Wang, K. C. and Botea, A. (2011). MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. *J. Artif. Intell. Res.*, 42:55–90.

Paper VI

6.6 Maintaining Ad-Hoc Communication Network in Area Protection Scenarios with Adversarial Agents

Marika Ivanova, Pavel Surynek, Diep Thi Ngoc Nguyen

In: *Proceedings of the 31st International Florida Artificial Intelligence Research Society Conference* (FLAIRS 2018), pp. 348-353, Melbourne, FL, USA, AAAI Press, 2018.

Maintaining Ad-Hoc Communication Network in Area Protection Scenarios with Adversarial Agents

Marika Ivanová
Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Bergen, Norway
marika.ivanova@uib.no

Pavel Surynek
Faculty of Information Technology
Czech Technical University
Czech Republic
pavel.surynek@fit.cvut.cz

Diep Thi Ngoc Nguyen
AIRC, National Institute of Advanced
Industrial Science and Technology (AIST)
Japan
diep.nguyen@aist.go.jp

Abstract

We address a problem of area protection in graph-based scenarios with multiple mobile agents where connectivity is maintained among agents to ensure they can communicate. The problem consists of two adversarial teams of agents that move in an undirected graph shared by both teams. Agents are placed in vertices of the graph; at most one agent can occupy a vertex; and they can move into adjacent vertices in a conflict free way. Teams have asymmetric goals: the aim of one team - *attackers* - is to invade into given area while the aim of the opponent team - *defenders* - is to protect the area from being entered by attackers by occupying selected vertices. The team of defenders need to maintain connectivity of vertices occupied by its own agents in a visibility graph. The visibility graph models possibility of communication between pairs of vertices.

We study strategies for allocating vertices to be occupied by the team of defenders to block attacking agents where connectivity is maintained at the same time. To do this we reserve a subset of defending agents that do not try to block the attackers but instead are placed to support connectivity of the team. The performance of strategies is tested in multiple benchmarks. The success of a strategy is heavily dependent on the type of the instance, and so one of the contributions of this work is that we identify suitable strategies for diverse instance types.

INTRODUCTION

In this work we study a generalization of *Area Protection Problem* (APP) with connectivity maintenance (APPC). APP is already a computationally hard problem (Ivanova, Surynek, and Hirayama 2018) (namely PSPACE-hard). In addition to APP, where two teams of mobile agents move in an undirected graph in a conflict free way, a possibility of communication among agents is required in APPC. APP itself can be regarded as a modification of known problem of *Adversarial Cooperative Path Finding* (ACPF) (Ivanová and Surynek 2014) where two teams of agents compete in reaching their target positions. Unlike ACPF, where the goals of teams of agents are symmetric - agents of each team try to reach their targets as first, the adversarial teams in APP have different objectives. The first team of *attackers* contains agents whose goal is to reach a pre-defined target location in the area being protected by the second team of *defenders*. Each attacker has a unique target in the protected area, and each target is assigned to exactly one attacker. The opponent team of defenders tries to prevent the attackers from reaching their targets by occupying selected locations, referred to as *destinations*, so that they cannot be passed by attackers. Specially in APPC, we require that vertices occupied by the defender team always form a connected subgraph with respect to the visibility graph. We assume that both teams use the same *cooperative path-finding* (CPF) algorithm for reaching temporarily selected locations.

Another distinction between ACPF and APP is a definition of victory of a team. A team in ACPF wins if all its agents reach their targets and agents of no other team manage to do so earlier. In APP, the team of defenders wins if all attackers are kept out of their targets. Our effort is to design destination allocation strategies for the defending team. A hard constraint that can never be violated will be that defending agents always form a connected component. Success of the strategy is measured from the defenders' perspective via an objective function which plays a role of soft constraint (area protection may not be perfect). The following objective functions can be pursued:

1. maximize the number of target locations that are not captured by the corresponding attacker
2. maximize the number of targets that are not captured by the corresponding attacker within a given time limit
3. maximize the sum of distances between the attackers and their corresponding targets
4. minimize the time spent at captured targets

The common feature of APP, APPC and other related problems is that once a location is occupied by an agent it cannot be entered by another agent until it is first vacated by the agent which occupies it (opposing agent cannot push it out). This property represents a key tool for the defenders to protect the area.

APPC has many real-life motivations from the domains of access denial operations, robotics with adversarial teams of robots or other type of penetrators (Agmon, Kaminka, and Kraus 2011), and computer games. In most practical applications, agents of given team need to communicate with

each other while individual robots can communicate at short visual range only as it has been already done in contemporary multi-robot systems Hence it needs to be ensured that the communication reaches every agent of the team. Such property can be modeled as connectivity over the visibility graph whose edges represent possibility of communication between pairs of vertices.

Our contribution consists in suggesting several on-line solving strategies for defenders that determine suitable vertices to be occupied by defenders so that attacker agents cannot pass into the protected area, and connectivity in the defender team is maintained. We conduct an experimental evaluation of the proposed strategies and assess their suitability for diverse types of APPC instances.

Communication Maintenance

APPC requires, in addition to APP, that any two defenders are able to communicate with each other at any time during their movement. The communication within the team of defenders is modeled by a connectivity of subgraph of the visibility graph induced by the set of occupied vertices. The visibility graph is derived from the graph in which agents move; it has the same set of vertices, but the set of edges is different in general. This assumption allows us to model the inability to communicate of two agents, if there is an obstacle between them.

In various practical applications of APP, the possibility of sending messages among the agents is often demanded. Agents may be equipped by an omnidirectional antenna or visual communication device (such as LEDs and IR sensors (Rubenstein et al. 2014)), and hence a message reaches all nodes within the communication range of its sender. This feature is often referred to as wireless advantage (Wieselthier, Nguyen, and Ephremides 2002). We assume that the agents have equal and constant communication range, and that they can also work as transceivers, which means that they have the ability to both transmit and receive a signal.

Related Work

APPC, APP, as well as ACPF share the way how movement of agents is treated with the basic variant of *cooperative path-finding problem - CPF* (*multi-agent path-finding* - MAPF) (Silver 2005; Ryan 2008; Wang and Botea 2011). In CPF, the task is to plan the movement of agents so that each agent reaches its unique target in a conflict free manner. Movements of agents in APPC at the low reactive level are assumed to be planned by some CPF algorithm where agents of own team cooperate while the opposing agents are considered as obstacles.

There exist multiple CPF algorithms both complete and incomplete as well as optimal and sub-optimal under various objective functions. A good compromise between quality of solutions and the speed of solving is represented by suboptimal/incomplete search based methods which are derived from the standard $A*$ algorithm. These methods include *LRA**, *CA**, *HCA**, and *WHCA** (Silver 2005). They provide solutions where individual paths of agents tend to be close to respective shortest paths connecting agents' locations and their targets. Conflict avoidance among agents is implemented via a so called reservation table in case of *CA**, *HCA**, and *WHCA**. Another approach is *LRA** which plans path for individual agents independently using A*, and relies on replanning whenever a conflict occurs. Since our setting in APPC is inherently suitable for a replanning algorithm, *LRA** is a candidate for the underlying CPF algorithm for APPC. Moreover *LRA** is scalable for large number of agents which is expected to happen in APPC.

Aside from CPF algorithms, systems with mobile agents that act in the adversarial manner represent another related area. These studies often focus on patrolling strategies that are robust with respect to various attackers trying to penetrate through the patrol path (Elmaliach, Agmon, and Kaminka 2009). Theoretical works related to APP also include studies on *pursuit evasion* (Vidal et al. 2002) or *predator-prey* (Haynes and Sen 1995) problems. The major difference between these works and the concept of APP/APPC is that we consider a relatively higher number of agents and our agents are more limited in their abilities.

DEFINITIONS AND ASSUMPTIONS

The environment in APPC is modeled by an undirected unweighted graph $G = (V, E)$. In this work we restrict the instances to 4-connected grid graphs with possible missing vertices indicating obstacles. Agents are not considered as obstacles. The team of attackers and defenders is denoted by $A = \{a_1, \ldots, a_m\}$ and $D = \{d_1, \ldots d_n\}$, respectively. Continuous time is divided into discrete time steps. At each time step agents are placed in vertices of the graph so that at most one agent is placed in each vertex. Let $\alpha_t : A \cup D \to V$ be a uniquely invertible mapping denoting configuration of agents at time step t. Agents can wait or move instantaneously into adjacent vertex between successive time steps to form the next configuration α_{t+1}. Abiding by the following movement rules ensures preventing conflicts:

- An agent can move to an adjacent vertex only if the vertex is empty, or is being vacated at the same time step by another agent
- No two agents enter the same vertex at the same time
- A pair of agents cannot swap across an edge

We do not assume any specific order in which agents perform their conflict free actions at each time step. However, our experimental implementation moves all attacking agents prior to moving all defender agents at each time step. The mapping $\delta^A : A \to V$ assigns a unique target to each attacker. The task in APP is to move defender agents so that area specified by δ^A is protected. This task can be equivalently specified as a search for strategy of destination assignments for the defender team. That is, we are trying to find an injective mapping $\delta^D_t : D \to V$ which specifies where defender agents should proceed at time step t as a response to previous attackers movements. The superscripts A and D are sometimes dropped when there is no danger of confusion.

From APP to APPC

APPC generalizes APP by considering connectivity constraints. As we assume that G is always a grid graph we can introduce connectivity constraints in the following way.

Consider an embedding of G in a plane such that all edges have length 1 and each vertex $v \in V$ has coordinates (x_v, y_v). The physical location l_v represented by v is the unit square area centered at $C_v = (x_v, y_v)$. Furthermore, let O denote the set of square locations representing obstacles.

Let r be the visibility range, i.e. the maximum distance between two locations such that two agents located at them can communicate together. The locations l_u and l_v can communicate with each other if the line segment C_uC_v does not intersect any obstacle and the length of the shortest path p_{uv} from u to v is at most r; shortly we say that l_u is visible from l_v and vice-versa. The visibility graph $G_r = (V, E_r)$ for a visibility range r contains edges between every two vertices that are mutually visible, formally: $(u,v) \in E_r \Leftrightarrow C_uC_v \cap O = \emptyset \wedge |p_{uv}| \leq r$. For any $S \subseteq V$ we use $G_r[S]$ in order to denote a subgraph of G_r induced by S. Finally, let S_t be the set of vertices occupied by defenders in time step t. Formally, $S_t = \{\delta_t^D(d) : d \in D\}$.

APPC is stated as a decision problem as follows:

Definition 1. *The decision APPC problem: Given an instance $\Sigma = (G, A, D, \alpha_0, \delta^A, r)$ of APPC, is there a strategy of destination allocations $\delta_t^D : D \to V$ such that $G_r[S_t]$ is connected for each $t = 0, 1, 2, \ldots$, and such that the team D of defenders is able to prevent all attackers from reaching their targets by moving defending agents according to δ_t^D.*

Typically it is not possible to protect all targets. We are therefore also interested in the optimization variant:

Definition 2. *The optimization APPC problem: Given an instance $\Sigma = (G, A, D, \alpha_0, \delta^A, r)$ of APPC, the task is to find a strategy of destination allocations $\delta_t^D : D \to V$ such that $G_r[S_t]$ is connected for each $t = 0, 1, 2, \ldots$, and such that the team D of defenders minimizes the number of attackers that reach their target by moving defenders according to δ_t^D.*

DESTINATION ALLOCATION

Since solving APPC in practice is a challenging problem due to its high computational complexity, designed methods are inexact and heuristic. Owing to the large search space we do not consider game-theoretic approaches even though the problem can be regarded as a two-player game. Our solving approaches are based on a technique called *destination allocation*. The basic idea is to assign a destination vertex to each defender and subsequently use some CPF algorithm adapted for the environment with adversaries, and to lead each defender to its destination while satisfying the connectivity constraint. A defender may be allocated to any vertex, including the attackers' targets. Solving approaches can be divided into two basic categories: *single-stage* and *multi-stage*. In single stage methods, targets are assigned to defenders only once at the beginning, as opposed to multi-stage methods, where the destinations can be reassigned any time during the agents' course. In our implementation, once all defenders are allocated to some destinations, they try to get to the desired locations using adapted LRA* algorithm. This work focuses merely on the single-stage methods. In all the studied strategies, every agent is allocated to exactly one location and every location is assigned to at most one defender. Attackers react on defenders by replanning within LRA* if they find that their original path is blocked. Let us recall several destinations allocation strategies and discuss their properties (Ivanova, Surynek, and Hirayama 2018).

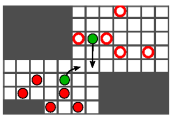


Figure 1: An example of bottleneck blocking. The defenders (green circles) may protect all the targets (empty red circles) from attackers (red circles) if they move to locations marked by the two arrows.

Random Allocation

For the sake of comparison, we consider a strategy, where each defender is allocated to a random target of an attacker. Neither the agent location nor the underlying grid graph structure is exploited.

Greedy Allocation

A greedy strategy is slightly improved approach. It takes the defenders one by one in a random order and allocates them to their closest target of an attacker. The greedy as well as the random strategy do not consider initial locations of attackers and do not exploit the structure of underlying graph in any way. These two methods always allocate defenders to given targets of attackers. The advantage of this approach is that if a defender manages to reach its assigned target, it will never be captured by the attacker aiming for that target. This can be useful in scenarios where the number of defenders is similar to the number of attackers. Unfortunately, such a strategy would not be very successful in instances where attackers significantly outnumber defenders.

Bottleneck Simulation Allocation

The idea behind the bottleneck simulation strategy is to gain some information from the map structure and the positions of attackers and assign defenders to vertices that would divert attackers from the protected area as much as possible. The aim is to successfully defend the targets even with a small number of defenders, as illustrated in Fig. 1.

We attempt to identify strategic bottlenecks and block them by defenders. In order to discover bottlenecks of general shape, we develop the following simulation strategy exploiting the underlying grid graph. The basic idea is that as attackers move towards the targets, they are expected to pass through vertices close to a bottleneck more often than through other vertices. This observation suggests to simulate the movement of the attackers and find frequently visited vertices. As defenders do not share the knowledge about paths being followed by attackers, frequently visited vertices are determined by a simulation in which paths of attackers are estimated.

After obtaining such a frequently visited vertex, we then explore its vicinity up to a given distance. If we find out

that there is indeed a bottleneck, its vertices are assigned to some defenders as their new destinations. Under the assumption that the bottleneck is blocked by defenders, the paths of attackers may substantially change. For that reason we estimate the paths again and find the next frequent vertex of which vicinity is searched for bottlenecks. The whole process is repeated until all available defenders are allocated to a destination, or until no more bottlenecks are found. Alg. 1 describes this procedure more formally.

Data: $G = (V, E)$, D, A
Result: Destination allocation δ^D
$D_{\text{available}} = D$; // Defenders to be allocated
$F = \emptyset$; // Set of forbidden locations
δ'_A = Random guess of δ_A;
while $D_{available} \neq \emptyset$ **do**
  for $a \in A$ **do**
    /* find the shortest path in G between an attacker a and its estimated target, that avoids passing through the forbidden locations in F */
    $p_a = \text{shortestPath}(\alpha_0(a), \delta'_A(a), G, F)$;
  end
  $f(v) = |\{p_a : a \in A \wedge v \in p_a\}|$; // Frequency of v
  $w \in \arg\max_{v \in V} f(v)$;
  $B = \text{exploreVicinity}(w)$; // Search a bottleneck
  if $B \neq \emptyset$ **then**
    $D' \subseteq D_{\text{available}}, |D'| = |B|$;
    assignToDefenders(B, D');
    $D_{available} = D_{\text{available}} \setminus D'$;
    $F = F \cup B$
  else
    /* If no new bottleneck is found, assume all have been already discovered */
    break ;
  end
end
/* If there are some defenders without a destination left, they will be allocated randomly */
assignToRandomTargets($D_{available}$);

Algorithm 1: Bottleneck simulation procedure

Another approach for bottleneck detection could be based on finding cut-set in a graph. However, our bottleneck simulation prefers bottlenecks frequently used by attackers.

CONNECTIVITY MAINTENANCE

The requirement of preserving the possibility of communication is modeled by a connectivity maintenance of subgraph of the *visibility graph* induced by the defenders' locations. The first task is therefore to create the visibility graph, which depends on the positions of obstacles in the map and a predetermined visibility range. The agents move using an adaptation of the LRA* algorithm that keeps the visibility subgraph connected. Paths are planned such that the first step of a defender must lead to a position which induces a connected visibility subgraph. Defenders follow paths computed by LRA* and whenever an agent is about to enter an occupied location, or if the next move would disconnect the communication subgraph, its path is recalculated.

The movement determined by such an approach will surely maintain the connectivity, however, in many instances, some defenders will not be able to reach the destination locations assigned to them. Our effort is to modify the allocation strategies so that the number of defenders that are not able to reach their assigned destinations is minimized.

Intuitively, defenders should be allocated to their destinations so that in the most optimistic case, when they all reach their desired locations, the connectivity is preserved. This constraint is not guaranteed to be satisfied in general. We propose the following approach to tackle this issue.

Initially, the defenders are partitioned into two sub-sets, *communicators* D_c and *occupiers* D_o, with a selected ratio $|D_c| : |D_o|$. The occupiers are allocated to attackers' targets according to one of the allocation strategies described above. In the best scenario, all defenders manage to reach their destinations. It is easy to check, for example by using BFS or DFS on the induced subgraph, whether the ideal final position of defenders maintains connectivity. If the connectivity is violated, the defenders reserved as communicators are allocated to targets so that the subgraph of the visibility graph induced by the defenders' target locations has as few connected components as possible. Fig. 2 depicts a situation where the three occupiers reached the attackers' targets assigned to them, but they alone are not able to communicate. Nevertheless, a suitable placement of two communicators ensures that the communication can take place. In fact, the

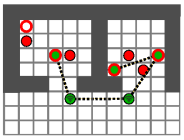


Figure 2: Three occupiers managed to reach the targets, but due to the wall they are not able to communicate. The presence of two communicators enables the communication via links marked by the dashed lines.

question whether it is possible to allocate destinations for D_c so that the desired position allows a communication among all defenders is already difficult.

Proposition 1. *Let Σ be an APPC instance with the set of defenders $D = D_c \cup D_o$, and let $\delta^{D_o}(d)$ for each occupier $d \in D_o$ be already determined. The decision problem whether there exists a destination allocation $\delta^{D_c} : D_c \to V$ to communicators in D_c such that all defenders maintain connectivity of the visibility graph at their final positions is NP-complete.*

Sketch of proof. The problem is obviously NP, because checking a connectivity can be done in polynomial time. In order to prove the NP-hardness, we reduce the known NP-

complete problem of *Vertex Cover (VC)* to our problem. Let $H = (V_H, E_H)$ be an instance of VC. For each $e \in E_H$ we create $v_e \in V$ such that V_e is a destination assigned to some occupier $d \in D_o$. For each $u \in V_H$ we construct $v_u \in V$ such that for all $e \in E_H$ incident with u we create $\{v_u, v_e\} \in E$. Vertices v_u s. t. $u \in V_H$ form a complete subgraph of G. Finally, we set $|D_c| = k$. Now H has a vertex cover of size at most k if and only if it is possible to assign destinations to communicators so that the connectivity of G_r is maintained in the desired position given by δ^{D_o}. □

Let T_o and T_c be the set of targets allocated to occupiers and communicators, respectively. If the induced subgraph $G_r[T_o]$ has several connected components, the used modification of LRA* algorithm could not lead all of them to their targets, because it would cause a loss of communication ability. At this point the set of communicators comes into play. The aim is to find target locations for communicators so that the graph $G_r[T_o \cup T_c]$ is connected. First, the connected components of $G_r[T_o]$. are identified. We then iterate while there are available communicators and connected components to be covered by them. In every iteration, a location l from which a communicator can cover a set of connected components that contains maximum number of targets allocated to occupiers is selected together with the set of covered connected components. The location l is subsequently assigned to the closest unallocated communicator. For a more formal explanation see Alg. 2.

Data: $G_r = (V, E_r)$, D_o, D_c, T_o
Result: Destination allocation δ^{D_c}
$T_c = \emptyset$; // Destinations assigned to communicators
while $D_c \neq \emptyset$ **do**
  $\mathcal{C}$ = connected components of $G_r[T_o \cup T_c]$;
  while $\mathcal{C} \neq \emptyset$ **do**
    /* A pair of a locatoin l and a subset $\mathcal{C}'$ of connected components covered by l that minimizes the number of vertices in $\mathcal{C}'$ */
    $(l, \mathcal{C}') = \arg \max_{\mathcal{C}' \in \mathcal{C}, l \in V} \{ \sum_{C \in \mathcal{C}'} |C| : \exists v \in C : (v, l) \in E_r \}$;
    /* An available agent closest to l */
    $a = \arg\min_{a \in D_c} \{ |p_{\alpha_0(a), l}| \}$;
    $\delta^{D_c}(a) = l$; // assign destination to agent
    $T_c = T_c \cup \{l\}$;
    $\mathcal{C} = \mathcal{C} \setminus \mathcal{C}'$;
    $D_c = D_c \setminus \{a\}$;
    if $D_c = \emptyset$ **then**
      break ;
    end
  end
end

Algorithm 2: Destination allocation to communicators

PRELIMINARY EXPERIMENTS

The aim of experimental evaluation is to compare individual strategies described above with their counterparts adapted to connectivity maintenance. We would like to find out whether the adaptation improves the success rate of a strategy and also how instance types affect its performance.

Our hypothesis is that when there is a sufficient number of defenders, the adaptation has little or no effect. We predict that in instances, where defenders are outnumbered by attackers, the adaptation increases the success rate of the corresponding strategy. Furthermore, it is likely that the simulation strategy is worse when the connectivity maintenance is required, because the identified bottlenecks may be far from each other, which makes it difficult to preserve communication among them.

We implemented all suggested strategies in Java as an experimental prototype. In our testing scenarios we use maps of different structure with various initial configurations of attackers and defenders. Our choice of testing scenarios is focused on comparing performance of the strategies and discovering what factors have impact on their success.

Different strategies are successful in different types of instances. It is therefore important to design the instances with a sufficient diversity, in order to capture strengths and weaknesses of individual strategies.

Instance generation and types

The instances used in the practical experiments are generated using a pseudo random generator, but in a controlled manner. An instance is defined by its map, the ratio $|A| : |D|$ and locations of individual defenders, attackers and their targets. These three entries form an input of the instance generation procedure. Further, we select rectangular areas inside which agents of both teams and the attackers' targets are placed randomly. The experiments are conducted on 3 different maps that vary in their structure (see Fig. 3).

(a) Orthogonal rooms (b) Ruins (c) Waterfront

Figure 3: Three different maps used in the evaluation

Each map is populated with agents of 3 different $|D| : |A|$ ratios, namely 1 : 1, 1 : 2 and and 1 : 5, with fixed number of attackers $|A| = 50$. The maximum number of moves of the agents is set to 150 for each team. Note that the individual instances are never completely fair to both teams. It is therefore impossible to make a conclusion about a success rate of a strategy by comparing its performance on different maps. The comparison should always be made by inspecting the performance in one type of instance, where we can see the relative strength of the studied algorithms.

Experimental results

The following set of experiments compares random, greedy, simulation strategy and their communication counterparts in different instance settings. Each of the following tables contains results associated with one map.

Each entry in the tables shows an average number of attackers that reached their targets at the end of the time limit. The average value is calculated for 10 runs in each settings, always with a different random seed. Random and greedy strategies have very similar results in all positions and team ratios. It is apparent and not surprising that with decreasing $|D| : |A|$ ratio, the strength of defensive strategies decreases.

Table 1: Average number of agents that eventually reached their target in the map Orthogonal rooms

$\|D\| : \|A\|$	RND	RND-C	GRD	GRD-C	SIM	SIM-C
1:1	26.0	29.0	25.5	29.1	20.8	28.3
1:2	41.0	39.6	39.4	40.5	29.3	31.7
1:5	48.1	45.7	46.1	46.8	46.9	46.8

We focused on evaluation of the effect of using communicating agents in implemented target allocation strategies. For each target allocation strategy we compare the standard version and the version with communicating agents.

Tab. 1 shows results for Orthogonal rooms map. It can be observed that using communicators is beneficial in case of random strategy where defenders tend to be outnumbered by attackers. On the other hand, communicators cause no improvement in Ruins map (Tab. 2). Small improvement of

Table 2: Average number of agents that eventually reached their target in the map Ruins.

$\|D\| : \|A\|$	RND	RND-C	GRD	GRD-C	SIM	SIM-C
1:1	21.5	21.1	24.8	24.7	18.3	18.6
1:2	42.1	40.2	39.0	40.3	37.1	36.9
1:5	47.1	47.1	46.0	46.2	44.3	43.8

the bottleneck simulation strategy can be observed in Waterfront map (Tab. 3) again in cases when defenders are outnumbered. Both types of maps where communicators turned out to be beneficial appear to have the structure of large open spaces separated by narrow bottlenecks.

Table 3: Average number of agents that eventually reached their target in the map Waterfront

$\|D\| : \|A\|$	RND	RND-C	GRD	GRD-C	SIM	SIM-C
1:1	20.7	21.6	18.9	18.5	20.8	21.9
1:2	35.2	31.2	30.7	31.4	35.8	33.5
1:5	41.6	41.4	40.7	40.7	42.3	41.3

CONCLUSION AND FUTURE WORK

We have designed several practical algorithms for APPC. We extended previous algorithms for APP with a technique of connectivity maintenance. This is done by dividing defending agents into two groups - occupiers and communicators. The role of occupiers is to protect the area while communicators are placed so that they cover as largest part of the protected area as possible in order to support connectivity among occupiers. Performed experimental evaluation indicates that the effect of using dedicated agents as communicators is much smaller than expected but there is some in maps having the structure of large open spaces separated by bottlenecks. One possible explanation of this behavior is that several defenders are not able to reach their targets because the ability of communication would be lost during their movement and this is not significantly affected by the target allocation. Hence, for the future work we plan to design and evaluate algorithms with more sophisticated mechanism for connectivity maintenance. A more promising direction seems to be an adaptation of LRA* rather than modifications of the allocation strategies.

References

Agmon, N.; Kaminka, G. A.; and Kraus, S. 2011. Multi-robot adversarial patrolling: Facing a full-knowledge opponent. *J. Artif. Intell. Res.* 42:887–916.

Elmaliach, Y.; Agmon, N.; and Kaminka, G. A. 2009. Multi-robot area patrol under frequency constraints. *Ann. Math. Artif. Intell.* 57(3-4):293–320.

Haynes, T., and Sen, S. 1995. Evolving beharioral strategies in predators and prey. In *Proc. of Adaption and Learning in Multi-Agent Systems, IJCAI'95 Workshop*, 113–126.

Ivanová, M., and Surynek, P. 2014. Adversarial cooperative path-finding: Complexity and algorithms. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014*, 75–82.

Ivanova, M.; Surynek, P.; and Hirayama, K. 2018. Area protection in adversarial path-finding scenarios with multiple mobile agents on graphs - a theoretical and experimental study of strategies for defense coordination. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, 184–191. INSTICC.

Rubenstein, M.; Ahler, C.; Hoff, N.; Cabrera, A.; and Nagpal, R. 2014. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems* 62(7):966–975.

Ryan, M. R. K. 2008. Exploiting subgraph structure in multi-robot path planning. *J. Artif. Intell. Res.* 31:497–542.

Silver, D. 2005. Cooperative pathfinding. In *Proc. of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, 2005*, 117–122.

Vidal, R.; Shakernia, O.; Kim, H. J.; Shim, D. H.; and Sastry, S. 2002. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Trans. Robotics and Autom.* 18(5):662–669.

Wang, K. C., and Botea, A. 2011. MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. *J. Artif. Intell. Res.* 42:55–90.

Wieselthier, J. E.; Nguyen, G. D.; and Ephremides, A. 2002. Energy-efficient broadcast and multicast trees in wireless networks. *Mob. Netw. Appl.* 7(6):481–492.

Appendix A

List of Computationally Hard Problems

The list below consists of NP-hard and PSPACE-hard problems that appear throughout this thesis. The problems are stated either in their decision or optimization form, depending on the context they are used. The decision versions are in fact NP-complete (PSPACE-complete), nevertheless, the NP-hardness (PSPACE-hardness) holds regardless. In some instances, several very similar problems differing only in small details are discussed in the thesis. In these cases, only the basic variants are stated in this list .

A.1 NP-hard Problems

A.1.1 Satisfiability

BOOLEAN SATISFIABILITY (SAT)
Input: A Boolean formula φ.
Question: Is there a truth assignment to variables of φ such that φ evaluates to TRUE?

3-SAT
Input: A Boolean formula φ in 3-CNF.
Question: Is there a truth assignment to variables of φ such that φ evaluates to TRUE?

3-3-SAT
Input: A Boolean formula φ in 3-CNF with an additional requirement that each variable appears at most three times and each literal at most twice.
Question: Is there a truth assignment to variables of φ such that φ evaluates to TRUE?

PLANAR 3-SAT [42]
Input: A set of unnegated variables $X = \{x_1, \ldots, x_n\}$ and negated variables $\bar{X} = \{\bar{x}_1, \ldots, \bar{x}_n\}$, a collection of clauses C over $X \cup \bar{X}$ such that

1. The graph $G = (X \cup C, E)$ with edge set $E = \{\{x, c\} : x \in c \vee \bar{x} \in c\}$ is planar.
2. Each clause $e \in C$ contains two or three literals $y \in X \cup \bar{X}$.

Question: Does there exist a truth assignment for the variables such that each clause is satisfied?

PLANAR 3,4-SAT
Input: Same as PLANAR 3-SAT with an additional requirement that each clause $c \in C$ contains exactly three literals and that each variable x_i appears in at most four clauses.
Question: Does there exist a truth assignment for the variables such that each clause is satisfied?

A.1.2 Graph Theory

CLIQUE
Input: A graph G and a an integer k.
Question: Does there exist a complete subgraph of G of size at least k in G?

MINIMUM STEINER TREE
Input: A graph $G = (V,E)$, non-negative edge-weights c, and $D \subseteq V$.
Question: Find a minimum-weight tree in G spanning D.

TRAVELLING SALESMAN PROBLEM
Input: A complete graph $G = (V,E)$ with edge weights $c : E \mapsto \mathbb{R}_0^+$ and an integer k.
Question: Is there a path $P = (V,E')$ such that $E' \subseteq E$ and $\sum_{e \in E'} c(e) \leq k$?

VERTEX COVER
Input: A graph $G = (V,E)$ and an integer k.
Question: Is there a subset $V' \subseteq V$ such that $\{u,v\} \in E \Rightarrow u \in V'$ or $v \in V'$ of size at most k?

A.1.3 Sets

3D-MATCHING
Input: Finite disjoint sets X, Y, and Z, a set $T \subseteq X \times Y \times Z$, and an integer k.
Question: Does there exist a set $M \subseteq T$ with $|M| \geq k$ such that for any two distinct triples $(x_1,y_1,z_1) \in M$ and $(x_2,y_2,z_2) \in M$, we have $x_1 \neq x_2, y_1 \neq y_2$, and $z_1 \neq z_2$?

SET COVER
Input: A universe $U = \{1,2,\ldots,n\}$, a collection $\mathcal{S}$ of sets such that $\cup_{S \in \mathcal{S}} S = U$, and an integer k.
Question: Is there a sub-collection $\mathcal{S}'$ of $\mathcal{S}$ of size at most k such that $\cup_{S \in \mathcal{S}'} S = U$?

A.1.4 Wireless Networks

MINIMUM ENERGY BROADCAST (MEB)
Input: A directed graph $G = (V,A)$, a source $s \in V$, and power requirements $p : A \mapsto \mathbb{R}_0^+$ such that $p_{ij} = p_{ji}$ for each $(i,j) \in A$.
Question: Find a power vector $(P_1,P_2,\ldots,P_n) \in \mathbb{R}^n$ of minimum component sum such that there exists a path from s to each $t \in V \setminus \{s\}$ in $G^P = (V,A^P)$, where $A^P = \{(i,j) \in A : p_{ij} \leq P_i\}$.

MINIMUM ENERGY MULTICAST (MEM)
Input: A directed graph $G = (V,A)$, a subset $D \subseteq V$, a source $s \in D$, and power requirements $p : A \mapsto \mathbb{R}_0^+$ such that $p_{ij} = p_{ji}$ for each $(i,j) \in A$.
Question: Find a power vector $(P_1,P_2,\ldots,P_n) \in \mathbb{R}^n$ of minimum component sum such that there exists a path from s to each $t \in D \setminus \{s\}$ in $G^P = (V,A^P)$, where $A^P = \{(i,j) \in A : p_{ij} \leq P_i\}$.

RANGE ASSIGNMENT PROBLEM (RAP)
Input: A directed graph $G = (V,A)$ and power requirements $p : A \mapsto \mathbb{R}_0^+$ such that $p_{ij} = p_{ji}$ for each $(i,j) \in A$.
Question: Find a power vector $(P_1,P_2,\ldots,P_n) \in \mathbb{R}^n$ of minimum component sum such that the induced graph $G^P = (V,A^P)$ is strongly connected, where $A^P = \{(i,j) \in A : p_{ij} \leq P_i\}$.

MINIMUM SHARED BROADCAST TREE (SBT)
Input: A graph $G = (V,E)$ and power requirements $p : A \mapsto \mathbb{R}_0^+$ such that $p_{ij} = p_{ji}$ for each $(i, j) \in A$.
Question: Find a spanning tree T in G minimizing

$$\sum_{i \in V} |T_{i_1/i}| p_{ii_2} + |T \setminus T_{i_1/i}| p_{ii_1},$$

where for an edge $\{i, j\}$ in T, $T_{i/j}$ denotes the subtree of T consisting of nodes whose path to j contains (i, j), and i_1 and i_2 denote the closest and second closest neighbor of i in T, respectively.

MINIMUM SHARED MULICAST TREE (SMT)
Input: A graph $G = (V,E)$, a subset $D \subseteq V$, and power requirements $p : A \mapsto \mathbb{R}_0^+$ such that $p_{ij} = p_{ji}$ for each $(i, j) \in A$.
Question: Find a tree T in G spanning D minimizing

$$\sum_{i \in V} \mu(T_{i_1/i}) p_{ii_2} + \mu(T \setminus T_{i_1/i}) p_{ii_1},$$

where for an edge $\{i, j\}$ in T, $T_{i/j}$ denotes the subtree of T consisting of nodes whose path to j contains (i, j), and where $\mu(S) = |V_S \cap D|$ for a subtree $S = (V_S, E_S)$ of T, and i_1 and i_2 denote the closest and second closest neighbor of i in T, respectively.

A.1.5 Broadcasting in Graphs

MINIMUM BROADCAST TIME (MBT)
Input: A graph $G = (V,E)$ and a subset $S \subseteq V$.
Question: Find the smallest integer $t \geq 0$ for which there exists a sequence $V_0 \subseteq \cdots \subseteq V_t$ of node sets and a function $\pi : V \setminus S \mapsto V$, satisfying:

1. $V_0 = S$ and $V_t = V$,
2. for all $v \in V \setminus S, \{v, \pi(v)\} \in E$,
3. for all $k = 1, \ldots t$ and all $v \in V_k$, $\pi(v) \in V_{k-1}$, and
4. for all $u, v \in V_k \setminus V_{k-1}$, $\pi(u) = \pi(v)$ only if $u = v$.

MINIMUM BROADCAST GRAPH
Input: An integer n.
Question: Construct a graph $G = (V,E)$ with the minimum possible number of edges such that for every node $s \in V$ there exists a sequence $V_0 \subseteq \cdots \subseteq V_{\lceil \log_2(n) \rceil}$ of node sets and a function $\pi : V \setminus \{s\} \mapsto V$, satisfying:

1. $V_0 = \{s\}$ and $V_{\lceil \log_2(n) \rceil} = V$,
2. for all $v \in V \setminus \{s\}, \{v, \pi(v)\} \in E$,
3. for all $k = 1, \ldots, \lceil \log_2(n) \rceil$ and all $v \in V_k$, $\pi(v) \in V_{k-1}$, and
4. for all $u, v \in V_k \setminus V_{k-1}$, $\pi(u) = \pi(v)$ only if $u = v$.

A.1.6 Path Finding for Multiple Robots

MULTI-ROBOT PATH FINDING

Input: An environment modeled by a graph $G=(V,E)$, a set of agents R with an initial and a target node for each agent defined by injective functions $\lambda_0 : R \mapsto V$ and $\lambda_+ : R \mapsto V$, respectively.

Question: Find a path $(\lambda_0(a), \lambda_1(a), \ldots, \lambda_{t(a)}(a) = \lambda_+(a))$ with possible repetition of nodes in G for each agent $a \in R$ from its initial node to its target node satisfying

- $\forall a \in R, \forall i \in \{0,\ldots,t(a)-1\} : \lambda_i(a) = \lambda_{i+1}(a) \vee \{\lambda_i(a), \lambda_{i+1}(a)\} \in E$, (that is, an agent moves along edges or stays at a node),
- $\forall a,a' \in R, \forall i \in \{0,\ldots,t(a)\} : \lambda_i(a) = \lambda_i(a') \Leftrightarrow a = a'$, (that is, a node can be occupied by at most one agent at a time),
- $\forall a,a' \in R, a \neq a', \forall u,v \in V, u \neq v, \forall i \in \{0,\ldots,t(a)-1\} : ((\lambda_i(a) = u) \& (\lambda_{i+1}(a) = v) \& (\lambda_i^{-1}(v) = a')) \Rightarrow (\lambda_{i+1}(a') \notin \{u,v\})$, (that is, if an agent a located at node u at time i moves to a node v that was occupied at time step i by some agent a' different from a, agent a' must move away at the same time, and two agents are not allowed to exchange their positions within one time step),

minimizing the makespan, i.e., $\min \max_{a \in R} t(a)$.

PEBBLE MOTION ON GRAPH

Input: An environment modeled by a graph $G=(V,E)$, a set of agents R with an initial and a target node for each agent defined by injective functions $\lambda_0 : R \mapsto V$ and $\lambda_+ : R \mapsto V$, respectively.

Question: Find a path $(\lambda_0(a), \lambda_1(a), \ldots, \lambda_{t(a)}(a) = \lambda_+(a))$ with possible repetition of nodes in G for each agent $a \in R$ from its initial node to its target node satisfying

- $\forall a \in R, \forall i \in \{0,\ldots,t(a)-1\} : \lambda_i(r) = \lambda_{i+1}(r) \vee \{\lambda_i(r), \lambda_{i+1}(r)\} \in E$, (that is, an agent moves along edges or stays at a node),
- $\forall a,a' \in R, \forall i \in \{0,\ldots,t(a)\} : \lambda_i(a) = \lambda_i(a') \Leftrightarrow a = a'$, (that is, a node can be occupied by at most one agent at a time),
- $\forall a,a' \in R, a \neq a', \forall u,v \in V, u \neq v, \forall i \in \{0,\ldots,t(a)-1\} : ((\lambda_i(a) = u) \& (\lambda_{i+1}(a) = v)) \Rightarrow (\lambda_i(a') \neq v)$, (that is, an agent is not allowed to move to a node that is simultaneously being left by another agent, it can only move to an unoccupied node),

minimizing the makespan, i.e., $\min \max_{a \in R} t(a)$.

A.2 PSPACE-hard Problems

A.2.1 Satisfiability

TRUE QUANTIFIED BOOLEAN FORMULA (TQBF)

Input: A fully quantified Boolean formula φ.

Question: Is φ equivalent to TRUE?

A.2.2 Path Finding for Multiple Robots with Adversarial Teams

ADVERSARIAL COOPERATIVE PATH FINDING (ACPF)

Input: An environment modeled by a graph $G = (V,E)$, a set of agents R with an initial and a target node for each agent defined by injective functions $\lambda_0 : R \mapsto V$ and $\lambda_+ : R \mapsto V$, respectively, and a partition of R into adversarial teams $T_0, \ldots, T_{k-1}$.

Question: Is there a strategy deciding a next move for agents of T_0 so that all agents of T_0 arrive at their targets before any other team does so, assuming the following movement rules are satisfied:

- $\forall a \in R, \forall i \in \mathbb{Z}_0^+ : \lambda_i(a) = \lambda_{i+1}(a) \vee \{\lambda_i(a), \lambda_{i+1}(a)\} \in E$, (that is, an agent moves along edges or stays at a node),
- $\forall a, a' \in R, \forall i \in \mathbb{Z}_0^+ : \lambda_i(a) = \lambda_i(a') \Leftrightarrow a = a'$, (that is, a node can be occupied by at most one agent at a time),
- $\forall a, a' \in R, a \neq a', \forall u, v \in V, u \neq v, \forall i \in \mathbb{Z}_0^+ : ((\lambda_i(a) = u) \& (\lambda_{i+1}(a) = v) \& (\lambda_i^{-1}(v) = a')) \Rightarrow (\lambda_{i+1}(a') \notin \{u, v\})$, (that is, if an agent a located at node u at time i moves to a node v that was occupied at time step i by some agent a' different from a, agent a' must move away at the same time, and two agents are not allowed to exchange their positions within one time step),
- $\forall i \in \mathbb{Z}_0^+, \forall a \in R \setminus T_{i(\mathrm{mod}\, k)} : \lambda_i(a) = \lambda_{i+1}(a)$, (the only agents that can move between time step i and $i+1$ are those from the team $T_{i(\mathrm{mod}\, k)}$),

regardless of the movement of agents from other teams?

AREA PROTECTION PROBLEM (APP)

Input: An environment modeled by a graph $G = (V,E)$, and a set of agents R partitioned into the set of defenders T_0 and the set of attackers T_1, with an initial node for each agent defined by injective functions $\lambda_0 : R \mapsto V$, and a target node for each attacker defined by $\lambda_+ : T_1 \mapsto V$.

Question: Is there a strategy deciding a next move for the defenders so that no attacker reaches its target, assuming the following movement rules are satisfied:

- $\forall a \in R, \forall i \in \mathbb{Z}_0^+ : \lambda_i(a) = \lambda_{i+1}(a) \vee \{\lambda_i(a), \lambda_{i+1}(a)\} \in E$, (that is, an agent moves along edges or stays at a node),
- $\forall a, a' \in R, \forall i \in \mathbb{Z}_0^+ : \lambda_i(a) = \lambda_i(a') \Leftrightarrow a = a'$, (that is, a node can be occupied by at most one agent at a time),
- $\forall a, a' \in R, a \neq a', \forall u, v \in V, u \neq v, \forall i \in \mathbb{Z}_0^+ : ((\lambda_i(a) = u) \& (\lambda_{i+1}(a) = v) \& (\lambda_i^{-1}(v) = a')) \Rightarrow (\lambda_{i+1}(a') \notin \{u, v\})$, (that is, if an agent a located at node u at time i moves to a node v that was occupied at time step i by some agent a' different from a, agent a' must move away at the same time, and two agents are not allowed to exchange their positions within one time step),
- $\forall i \in \mathbb{Z}_0^+, \forall a \in R \setminus T_{i(\mathrm{mod}\, 2)} : \lambda_i(a) = \lambda_{i+1}(a)$, (the only agents that can move between time step i and $i+1$ are those from the team $T_{i(\mathrm{mod}\, 2)}$),

regardless of the movement of attackers?

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS

uib.no

ISBN: 9788230846056 (print)
9788230847053 (PDF)