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 ~~from~~ ~~among~~ the set of all feasible solutions to the problem instance. The optimization version of CLIQUE asks for ~~finding~~ ~~the~~ 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 ~~answer~~ CLIQUE problem for a given (G,k). The MAXIMUM CLIQUE problem can then be solved by answering its optimization version for $k = 1,2,\ldots$ until the answer is "no" for some k', and so the maximum clique has the size $k'-1$.

Throughout this thesis, we use several well-known concepts from ~~the~~ complexity theory, which we state in the following. Detailed explanations of the terminology can be found in any textbook on this topic such as [66].

An *algorithm* is a procedure that solves a given problem in a finite number of steps. 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 X′, if a polynomial computable function f exists where for every w, w is a yes-instance of X ⇔ f(w) is a yes instance of X′.*

Definition 10. *Decision problem X is NP-complete if it satisfies:*

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

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.

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 X is PSPACE-complete if it satisfies:*

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

If X satisfies only the second condition 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. *[75] 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 maximizations problems $\rho < 1$.

Definition 14. *The class APX is the set of NP optimization problems that have 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. *[75] A polynomial-time approximation scheme (PTAS) is a family of algorithms $\{A_\varepsilon\}$, where there is an algorithm for each $\varepsilon > 0$, such that $\mathcal{A}_\varepsilon$ is a $(1+\varepsilon)$-approximation algorithm for minimization problems, and $(1-\varepsilon)$-approximation algorithm for maximization problems.*

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

① The first four papers have the common characteristic that integer programming is an essential solution method. In this sense, the last two papers are distinguished. They are devoted to adversarial ... ← from the former

② expand a solution by adding new edges

Chapter 5

Contribution of the Thesis and Prospective Research

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 integer programming as an initial method for obtaining solutions. The last two papers are ~~more distant~~, as they study 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.

① distinguised

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

Paper I introduces 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 [80]. 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 ~~a~~ construction of a solution to ~~a~~ different problems, specifically MST and MEB, respectively, 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 ~~add edges to the growing solution~~, anticipate subtree sizes in the resulting solution, and thereby give an ~~estimation~~ of the final objective function each time a new edge is appended.

✓ The construction is

② value ← estimate

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: Shared Multicast Trees in Ad-hoc Wireless Networks

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 six time worse than the optimum, proving that the approximation ratio of the MST algorithm 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 are developed in Paper III, the first model is based on broadcast trees whereas the second one adapts several network flow techniques presented in [57]. 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 time necessary for obtaining a lower bound by solving an LP relaxation of the models, and their strength. LP relaxations of the strongest model yield 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. A comparison of the lower bounds produced during the course of CG method with lower bounds that are yielded during standard 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 [54], 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. Larger instances 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, and by exploiting the problem characteristics, we introduce a method that

① If LRA* is defined in previous sections, please provide a section reference.

iteratively solves a decision version of the ILP model and finally finds an optimal solution if given ~~a~~ sufficient time. This method is also applied ~~on~~ to the LP relaxation of the model, and the outcome indicates ⊥this method yields strong lower bounds, often coinciding with the optimum. Besides the continuous relaxation, analytical and combinatorial lower bounding techniques are studied, but 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 ⊥main ~~aspect~~ feature is an iterative construction of broadcast trees of restricted size. This algorithm is parametrized by the size limitation of the broadcast trees, and 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 logarithmic lower bound 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) [38] problem. Its main contribution lies in the proof that APP is PSPACE-hard. This result is achieved by demonstrating a polynomial-time reduction from the PSPACE-complete problem 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 CPF algorithm, in this case we selected LRA*. A destination can be regarded as a target node for defenders, the difference is that while a target is ~~a~~ part of the input, ⊥destination is determined by a solution method.

The strategies differ in the approach of target allocation. ~~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 successfully 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 among 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 there is an edge between two nodes if and only if agents placed at the nodes 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, 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.