DISSERTATION | DOCTORAL THESIS

Titel | Title

Essays in Combinatorial Optimization for Graph-Based Problems

verfasst von | submitted by

Maryam Dehghan Chenary

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 794 370 403
Dissertationsgebiet lt. Studienblatt | Field of study as it appears on the student record sheet:	Business Analytics, Logistics and Operations Research
Betreut von | Supervisor:	emer. o. Univ.-Prof. Dipl.-Ing. Dr. Richard Hartl Ass.-Prof. Dr. Christian Tilk

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and encouraged me throughout my PhD journey. First and foremost, I am sincerely grateful to my supervisor, Prof. Richard Hartl, for his outstanding guidance, invaluable advice, and continuous support.

My heartfelt thanks also go to my co-supervisor, Prof. Christian Tilk, for his profound expertise, mentorship, and constructive feedback. As his first PhD student, I had the privilege of working within an inspiring and supportive academic environment. His openness to discussion, patience, and guidance on both theoretical and practical aspects of research have been invaluable to my development as a scholar.

I would also like to extend my sincere appreciation to the academic reviewers who graciously agreed to assess this thesis, and to the anonymous reviewers of my publications for their insightful comments and constructive feedback, which have greatly improved the quality of this work. Special thanks are due to the University of Vienna, in particular the Department of Business Decisions and Analytics and the Oskar Morgenstern Doctoral School, for their financial and institutional support throughout my doctoral studies. Their assistance made this research possible and provided an excellent academic environment for carrying out my work.

I am profoundly grateful to my husband, whose unwavering support, patience, and understanding sustained me through every challenge of this journey. His encouragement and love have been a constant source of motivation and strength.

Finally, my deepest thanks go to my family, who, although more than 5,000 kilometers away, have never been far from my heart. Their love has always felt less than a centimeter away, reminding me that distance can never weaken a genuine connection.

Preface

This thesis presents research that has been carried out and published over the past few years. Four papers have been published, and one manuscript is currently under submission. Each chapter corresponds to an individual research topic and is either a published paper or a submitted manuscript, with the exception of the introduction (Chapter 1) and the conclusion (Chapter 7), which were written specifically for this thesis to provide integration and reflection.

More precisely, Chapter 2 titled *On Combining Conventional Point-to-Point and Automated Waste Collection Systems*, has been published in the journal *Networks*. Chapter 3 refers to the article *An Exact Branch-Price-and-Cut Algorithm for Integrated Waste Collection Systems: Elementary Shortest Path and Hop Constraint Steiner Tree Problems*, which has been submitted to *European Journal of Operational Research*. Chapter 4, entitled *TSCDA: A Dynamic Two-Stage Community Discovery Approach*, was published in *Social Network Analysis and Mining (Springer)*. Chapter 5, *Discovering Communities in Networks: A Linear Programming Approach Using Max–Min Modularity*, appeared in the proceedings of the *Federated Conference on Computer Science and Information Systems* (FedCSIS). Finally, Chapter 6,*Gain and Pain in Graph Partitioning: Finding Accurate Communities in Complex Networks*,was published in *Algorithms* (MDPI).

The results and intermediate progress of this research have been presented and discussed at several international conferences, including the *AIRO Conference 2025*, *EURO 2024*, and *FedCSIS 2023*, as well as in specialized academic seminars and workshops on optimization and network science.

Contents

List of Figures **IX**

List of Tables **XI**

List of Algorithms **XIII**

List of Abbreviations **XV**

1 Introduction **1**

2 On Combining Conventional Point-To-Point and Automated Waste Collection Systems **7**
2.1 Introduction . . . 7
2.2 Literature review . . . 10
2.3 The combined waste collection problem . . . 13
2.4 The column-generation-based matheuristic . . . 14
2.5 Computational results . . . 19
2.6 Case study for Vienna, Austria . . . 24
2.7 Conclusions and outlook . . . 27
2.8 Appendix . . . 28

3 An Exact Branch-Price-and-Cut Algorithm for Integrated Waste Collection System: Elementary Shortest Path and Hop Constraint Steiner Tree Problems **37**
3.1 Introduction . . . 37
3.2 Literature Review . . . 39
3.3 Problem Description . . . 42
3.4 Full Decomposition-Based Column Generation for Tree Generation . . . 43
3.5 Partial Decomposition-Based Column Generation for Tree Generation . . . 50
3.6 Computational results . . . 52
3.7 Conclusions and outlook . . . 61
3.8 Appendix . . . 62

4 TSCDA: A dynamic two-stage community discovery approach **65**
4.1 Introduction . . . 65
4.2 TSCDA: Two-Stage Community Detection Algorithm . . . 69
4.3 Experiments . . . 74
4.4 Conclusion . . . 80

5 Toward an Optimal Solution to the Network Partitioning Problem 81
5.1 Introduction and related work 81
5.2 Mathematical modeling 84
5.3 Solution Approach 86
5.4 Computational results 88
5.5 Conclusion 91

6 Gain and Pain in Graph Partitioning: Finding Accurate Communities in Complex Networks 93
6.1 Introduction and literature review 93
6.2 Methodology 96
6.3 Evaluation of accuracy 105
6.4 Conclusion 109

7 Conclusion 111

Bibliography 115

Abstract 127

Zusammenfassung 129

List of Figures

2.1 Principle of a point-to-point waste collection system and an AWCS. . . . 8
2.2 Schematic of the CWCP. . . . 13
2.3 Impact of the tree capacity Q^T and the cost ratio f. . . . 21
2.4 Utilization of AWCS. . . . 21
2.5 Result for the subset $\{16, 17\}$ of the districts. The AWCS source o^T is shown as •, the vehicle depot o^R as ○, customers served by the AWCS as •, and customers served by truck as •. The level of the demand determines the size of the point. . . . 25
2.6 Impact of the tree capacity and cost ratio on the total cost and the percentage of customers in the tree. Each data point represents one instance. . . . 35
2.7 Visualization of different demand scenarios. . . . 36

3.1 Principle of a point-to-point waste collection system and an AWCS [53]. . 38
3.2 Schematic of the CWCP with a hop limit. . . . 43
3.3 Impact of considering SECs for FulLA and FulPO. . . . 57
3.4 Impact of considering SECs for ParLA and ParPO. . . . 58

4.1 Two communities C_1 and C_2 with influential vertices a and b, respectively. C_2 is a worse community since it has more disconnected node pairs than C_1. $q_1 = \frac{3}{12}$ and $q_2 = \frac{5}{12}$ demonstrate the superiority of C_1 over C_2. . . . 71
4.2 ARI (a) and NMI (b) performance rank of each algorithm applying to the LFR synthetic graphs, generated by considering four different mixing parameters μ. . . . 77
4.3 Average normalized ARI and NMI performance rank of different algorithms applying to the real-world networks, presented in Table 6.2. The average NMI and ARI values obtained by TSCDA are respectively equal to 0.605 and 0.645. . . . 78
4.4 Two heatmap diagrams show the performance of the discovered initial communities (CIC stage) in terms of (a) ARI and (b) NMI metrics. Left columns specify the results for when the distance factor is entirely ignored. The right columns show the results obtained by taking the distance factor into account. Be noted that each row represents a network. . . . 79

5.1 Illustration of a network and its communities as subsets of vertices with densely connected nodes within each subset and sparser connections to nodes outside the batch. . . . 82

5.2 Comparison between NMI values achieved by (i) blue curve: our method, (ii) red curve: method in [64], and (iii) green curve: the conventional Max-Min modularity. 89
5.3 Two networks from Table 6.1 and their detected communities using the proposed algorithm. 90
5.4 The time elapsed (in terms of seconds) for solving different methods. . . . 90

6.1 Comparison between NMI values achieved by our approach, the blue curve, and other understudy algorithms using real-world networks presented in Table 6.2. 107
6.2 Comparison between NMI ranks of each algorithm applying to the LFR synthetic graphs, generated by considering four different mixing parameters μ. 108
6.3 Execution time associated with each algorithm applied to the real-world networks presented in Table 6.2. 109

List of Tables

2.1 Comparison of the exact BaP algorithm and the matheuristic. 19
2.2 Impact of varying demand on the percentage of the average number of customers and total demand covered by the AWCS. 23
2.3 Subsets of instances defined by districts of Vienna and number of plastic waste bin locations per subset. 24
2.4 Aggregated results of the Vienna case study. 26
2.5 Detailed results of the exact BaP algorithm (BaP) and the matheuristic (MH). 29
2.6 Results for cost ratio $f = 1.1$. 30
2.7 Results for cost ratio $f = 2$. 31
2.8 Results for cost ratio $f = 3$. 32
2.9 Results for cost ratio $f = 5$. 33
2.10 Instances of the CVRPLIB in each group. 34

3.1 Comparison of the LP relaxation performance of all four BPC algorithms. 53
3.2 Comparison of the IP performance of all four BPC algorithms. 55
3.3 Comparison of the IP performance of all four BPC algorithms on 7 instances unsolved at the root for at least one algorithm and hop limit value. 56
3.4 Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 50%. Results are aggregated over those 15 instances for which the LP-Relaxation could be solved for all scenarios and all algorithms. 59
3.5 Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 50% on 12 instances unsolved at the root for at least one of the full decomposition algorithms in at least one scenario. 60
3.6 Impact of hop limit (H), tree capacity (Q^T), and cost ratio (f) on AWCS utilization and total cost. 61
3.7 Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 33%. Results are aggregated over those 15 instances for which the LP-Relaxation could be solved for all scenarios and all algorithms. 63
3.8 Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 33% on 12 instances unsolved at the root for at least one of the full decomposition algorithms in at least one scenario. 64

4.1 Several connectivity-based quality functions for measuring the quality of a community C. $|E_C^{out}|$, $|E_C^{in}|$, $|C|$ and $deg(i)$ respectively represent the number of external connections of C, number of internal connections of C, the number of vertices belonging to C, and the degree of node i. 66
4.2 Real-world networks 75
4.3 Comparison results of different stages of TSCDA and also the total time elapsed in the algorithm. 79

5.1 Networks under-study 88

6.1 Several connectivity-based quality functions for measuring the quality of a community C. $|E_C^{out}|$, $|E_C^{in}|$, $|C|$ and $deg(i)$ respectively represents the number of external connections of C, number of internal connections of C, the number of vertices belonging to C, and the degree of node i. 94
6.2 Networks under-study 106
6.3 Used values for the algorithm's input parameters k_1 and k_2 associated with each of the real-world networks depicted in Table 6.2. 107

List of Algorithms

1 Creating Initial Communities (CIC) 73
2 Updating Communities (UC) 74
1 Initial partitioning (IP) 98
2 Partition Improvement Communities (PI) 99
3 The proposed algorithm for community discovery 105

List of abbreviations

CWCP	Combined Waste Collection Problem
AWCS	Automated Waste Collection System
ARP	Arc Routing Problem
VRP	Vehicle Routing Problem
CVRP	Capacitated Vehicle Routing Problem
LP	Linear Programming
ILP	Integer Linear Programming
RMP	Restricted Master Problem
BaP	Branch-and-Price
SPPRC	Shortest Path Problem with Resource Constraints
MST	Minimum Spanning Tree
NWSTP	Node-Weighted Steiner Tree Problem
MH	Matheuristic
HiDem	High Demand in one District
HiDemR	High Demand Randomly Distributed
SL	Small and Large demands
U	Uniform
SCap	Small Vehicle Capacity
LCap	Large Vehicle Capacity
Q	Quadrant
STP	Steiner Tree Problem
HSTP	Hop-Constrained Steiner Tree Problem
STPRH	Steiner Tree Problem with Revenues and Hop Constraints
STPRBH	Steiner Tree Problem with Revenues, Budget, and Hop Constraints

HMSTP Hop-Constrained Minimum Spanning Tree Problem

BPC Branch-Price-and-Cut

SEC Subtour Elimination Constraints

TreeSEC Tree Subtour Elimination Constraints

TreeSECu Lifted Tree Subtour Elimination Constraints

SRI Subset-Row Inequalities

NACut Node-Arc-Cut Constraint

FulLA Layer-assignment-based formulation in full decomposition

FulPO Partial-ordering-based formulation in full decomposition

ParLA Layer-assignment-based formulation in partial decomposition

ParPO Partial-ordering-based formulation in partial decomposition

TSCDA Two-Stage Community Discovery Algorithm

CIC Creating Initial Communities

UC Updating Communities

IP-MM Integer Programming formulation of Max–Min Modularity

IPs-MM Sparse model for IP-MM

LFR Lancichinetti–Fortunato–Radicchi

EC External Connectivity metric

IC Internal Connectivity metric

MIE Mixed-Internal-External metrics

NMI Normalized Mutual Information

ARI Adjusted Rand Index

Chapter 1

Introduction

The increasing complexity of modern networked systems has created a pressing need for efficient optimization methods capable of handling large-scale, interdependent structures. Such systems arise in both engineered systems for operational research applications and complex network interactions. Optimizing these interconnected systems poses significant computational challenges, often requiring advanced combinatorial techniques to generate high-quality solutions within reasonable time frames.

This thesis addresses these challenges by developing and evaluating combinatorial optimization methods tailored to two state-of-the-art classes of network-based problem structures: In this respect, the first part of the thesis focuses on the design and optimization of urban logistics networks, addressing a novel combined waste collection system, which combines traditional truck-based collection with an automated pneumatic pipe network. The second part investigates complex network analysis, with an emphasis on *community detection* and network partitioning based on a refined definition of node influence and optimization-driven formulations. Despite their distinct evolution and application domains, both fields share a common methodological foundation: problems are formulated as mathematical programming models in which meaningful substructures, such as routes, trees, or communities, must be optimized under given constraints using classical operations research techniques.

Accordingly, this thesis establishes a conceptual bridge between *structure design* in operational systems and *structure discovery* in complex networks, linking them under the common objective of developing efficient algorithms to analyze and optimize interconnected systems in pursuit of the overarching research questions: How can the CWCP jointly determine (i) whether each collection point is assigned to truck service or to the automated system, and (ii) the corresponding truck routes and a cost-minimizing pneumatic tree? How can exact and matheuristic optimization techniques be adapted to CWCP to develop a holistic solution approach based on a set-partitioning formulation and handle computationally challenging extensions? How can large-scale complex networks be partitioned into meaningful communities using connectivity-based metrics that account for node influence? How can linear programming formulations be leveraged to find optimal or near-optimal community partitions in complex networks?

To answer these questions methodologically, the thesis integrates a broad spectrum of optimization paradigms, algorithmic strategies, including heuristic, matheuristic, and exact methods, decomposition techniques, and computational validation to tackle the inherent computational complexity of the problems studied. Each method is carefully selected and tailored to the specific characteristics of the respective application, ensuring both efficiency and theoretical accuracy. Moreover, the developed algorithms are rigorously validated through computational experiments on benchmark instances and real-world data.

1.0.1 Thesis Overview

This thesis by publication consists of five articles, each of which has been published in or submitted to scientific journals, collectively addressing the research questions outlined above. The chapters are summarized as follows:

Chapter 2 introduces, for the first time, the *Combined Waste Collection Problem* (CWCP). Existing studies reviewed in the literature highlight that both the conventional *point-to-point*, truck-based collection system and the *Automated Waste Collection System* (AWCS), in which waste is transported through a network of underground pipes to a central terminal, exhibit distinct strengths and weaknesses when implemented independently [131, 162]. Motivated by these findings, a hybrid approach is proposed that leverages the advantages of both systems while mitigating their inherent limitations, with the aim of achieving a more efficient waste-collection strategy. The CWCP is defined on an undirected complete graph in which each edge is associated with two transportation costs, one corresponding to the truck-based system and the other to the pneumatic network. The problem is modeled as a two-stage decision process: at the first stage, each collection point is assigned to one of the two collection systems; at the second stage, a *Capacitated Vehicle Routing Problem* (CVRP) is solved for the truck-assigned nodes, and a cost-minimal rooted tree is constructed for the AWCS nodes. Both stages and the respective problems are interdependent, making the optimization of the overall system a challenging task. A holistic solution approach is developed based on a set-partitioning formulation utilizing route and tree variables. Due to the large number of variables, the formulation is solved heuristically using a column-generation-based matheuristic.

The column-generation paradigm, first formalized under the Dantzig–Wolfe decomposition principle, solves a *restricted master program* (RMP) with a manageable subset of variables and dynamically introduces new variables (columns) with negative reduced costs by solving a *pricing problem*. Iteratively, the RMP and subproblem solutions converge to the linear relaxation of the master formulation. In integer settings, column generation is integrated with branch-and-bound, leading to the well-known *branch-and-price* framework. In the CWCP, the resulting subproblems are an *elementary shortest-path problem with capacity constraints* and a variant of the *node-weighted Steiner tree problem*. Since solving the tree subproblem exactly is computationally demanding, a heuristic method is employed for this subproblem, while the routing subproblem is solved exactly using dynamic programming labeling algorithms [96].

The proposed algorithm is tested on benchmark datasets, including extended variants of well-known CVRP instances, as well as on real-world data from the city of Vienna. Computational results indicate that the matheuristic produces near-optimal feasible solutions on small instances, with the largest deviation from the optimal upper bound remaining below one percent. For larger instances, the matheuristic frequently outperforms an exact branch-and-price algorithm within a reasonable time limit, highlighting its effectiveness in scenarios where exact methods are computationally limited. The study further demonstrates that integrating an AWCS into the collection system is particularly beneficial when certain collection points exhibit high demand relative to vehicle capacities. In most cases, the combined system reduces overall collection costs compared to a traditional truck-only approach. Moreover, the results emphasize the importance of strategic facility placement: for example, optimally locating the AWCS

depot yields average cost savings of approximately five percent compared to random placement. In the majority of high-demand scenarios, the AWCS operates at or near its maximum capacity.
The full paper corresponding to this chapter can be found in DehghanChenary et al. [53].

In Chapter 3, building on the strategic CWCP formulation of Chapter 2 [53], a key practical limitation is addressed by incorporating a maximum branch-length (hop) constraint into the AWCS design in order to capture the suction–pressure restrictions inherent to pneumatic pipeline systems. With this addition, the AWCS pricing problem becomes a variant of the *Steiner Tree Problem with Revenues and Hop Constraints* (STPRH), thereby extending the overall framework to a *CWCP with a hop limit*. This enhancement aligns closely with real-world AWCS installations, where pipeline-length restrictions are fundamental and consistently noted throughout the literature. To solve this enriched problem, the first exact algorithms for the CWCP with a hop limit are developed through the design of four *branch-price-and-cut* (BPC) methods based on two alternative decomposition strategies: (i) a *full decomposition*, in which routing and tree generation are handled independently in two pricing problems, and (ii) a *partial decomposition*, where the tree structure is modeled directly in the master problem and column generation is applied only to the routing part.

Since the tree pricing problem had previously been identified as the principal computational bottleneck in CWCP studies [53], this challenge is further addressed by implementing and examining two state-of-the-art STPRH formulations originally proposed by Jabrayilov and Mutzel [99] (partial-ordering) and Sinnl and Ljubić [159] (layer-assignment). Unlike the sparse instances typically examined in the STPRH literature, the considered setting requires solving these models on complete graphs, which considerably increases their computational difficulty. Combining the two decomposition approaches with these two formulations yields four distinct BPC algorithms.

Extensive computational experiments on benchmark instances were conducted to evaluate the four BPC algorithms. The results show that the choice of decomposition strategy has a decisive impact on computational performance. Full decomposition approaches yield tighter bounds and solve more small instances to optimality, whereas partial decomposition scales more reliably to larger instances and consistently produces high-quality solutions. Across both strategies, the partial ordering formulation outperforms the layer-assignment formulation. Moreover, the partial-ordering-based formulation in the partial decomposition algorithm achieves the best-known upper bound in most cases. The experiments further indicate that instance characteristics and modeling choices strongly influence solvability. In particular, certain parameter settings, such as larger hop limits, higher tree-to-route cost ratios, and smaller tree capacities, consistently lead to easier instances and improved algorithmic performance. In addition, implementation enhancements, most notably the inclusion of subtour elimination constraints, significantly improve computational efficiency, especially for the partial-decomposition variants.
This work has been submitted to the *European Journal of Operational Research*.

Chapter 4 focuses on the analysis of large-scale, complex networks, with a primary emphasis on community detection, that is, the identification of densely interconnected groups of nodes that represent functional or structural subunits of a network [136, 135]. In this chapter, a new connectivity-based quality metric is introduced that explicitly

incorporates the impact of each community's most influential vertex on the remaining nodes in that community, thereby bridging structural cohesion with centrality-driven notions of influence [115, 176]. The proposed metric is designed to simultaneously minimize the number of outgoing (inter-community) edges and the radius of each community, thus favouring communities that are both well separated from the rest of the network and internally compact. Building on this metric, *TSCDA*, a dynamic two-stage community discovery approach, is developed [63]. In the first stage, high-quality initial communities are constructed by selecting a set of highly influential nodes that are sufficiently distant from one another, in the spirit of classical local-search and partition-refinement strategies used in graph partitioning [105, 13]. The second stage then refines these initial communities through a local search procedure, conceptually related to refinement steps in multilevel modularity heuristics, label-propagation–based methods, and other two-phase algorithms [28, 143]. Extensive experiments on a diverse collection of real-world and synthetic benchmark networks, including well-known social and biological graphs and the LFR-class benchmarks, demonstrate that TSCDA achieves community partitions that compare favourably with state-of-the-art algorithms in terms of normalized mutual information, adjusted Rand index, and related evaluation measures [92, 50], while maintaining scalability on networks with hundreds of thousands of vertices.
One can refer to Ferdowsi et al. [63] for the full version of this study.

Chapter 5 investigates the community detection problem through the lens of *Max-Min Modularity*, a robust variant of the classical modularity measure proposed initially by Newman and Girvan [136] and Newman [134], and subsequently refined to mitigate some of its known limitations, such as the resolution limit and sensitivity to small perturbations [81, 41]. Building on the integer and linear programming formulations for the Max-Min Modularity maximization problem introduced by Ferdowsi and Khanteymoori [64], a new, significantly more compact integer model is proposed that drastically reduces the number of variables and constraints while provably preserving the same set of optimal solutions. The resulting formulation is conceptually related to mathematical-programming approaches to modularity and graph partitioning [3, 7], and exploits structural insights from clique-partitioning and clustering models with redundant constraints [83, 127]. To solve this model efficiently, a solution procedure is designed that combines an intelligently chosen initial feasible solution, generated using the first stage of the TSCDA algorithm of Chapter 4 [63], with a row-and-column generation strategy embedded in a branch-and-bound framework. Extensive experiments on twelve widely used real-world benchmark networks with established or widely accepted community structure [184, 31] demonstrate that the proposed exact approach consistently attains higher Max-Min Modularity values and yields communities that align more closely with ground-truth partitions than existing heuristic and exact modularity-based methods [28, 42]. At the same time, the compactness of the model and the tailored strategy lead to substantial computational savings, making high-quality Max-Min Modularity optimization practical for medium-sized networks that are out of reach for earlier formulations.
The reader can find this work in Ferdowsi and DehghanChenary [61].

Chapter 6 develops a hybrid community-detection framework that unifies the connectivity-based quality metric of Chapter 4 with the Max-Min Modularity model of Chapter 5 to uncover highly accurate communities in complex networks. The approach begins by ap-

plying the fast two-stage heuristic of TSCDA [63] to obtain high-quality initial partitions, and then uses these communities to construct a new *complementary graph* that more faithfully encodes the absence of meaningful relationships between pairs of nodes that are not directly connected. On this enriched representation, the Max-Min Modularity maximization problem is reformulated as a revised integer programming model, and it is shown that an equivalent sparse formulation with substantially fewer variables yields identical optimal solutions while reducing memory requirements and solution time, in the spirit of recent work on compact modularity and partitioning formulations [3, 7]. By integrating a problem-specific row-generation strategy with the heuristic initialization and by leveraging insights from hybrid and metaheuristic community-detection schemes [160, 39], an optimization procedure is designed that adaptively balances exact and heuristic steps depending on network size and structure. This results in near-optimal solutions for large graphs at a fraction of the cost of purely exact methods, while retaining strong guarantees for small and medium instances [64, 61]. Comprehensive experiments on a broad set of real-world and synthetic networks, evaluated with standard community-quality and agreement metrics [114, 38], show that the proposed hybrid framework consistently outperforms classical modularity-based heuristics [133, 28] and recent high-quality optimization approaches, achieving superior partitions while remaining scalable to networks with tens of thousands of nodes [185, 77].
This work can be found in Ferdowsi and DehghanChenary [62].

Chapter 7 concludes the thesis by summarizing the main contributions and outlining directions for future research.

For all chapters that correspond to already published work, I have obtained the right to include the submitted version in my thesis. The only changes made, and the only differences to the published versions, concern formatting, structural changes (renumbering of sections), and updates to cited references.

Chapter 2

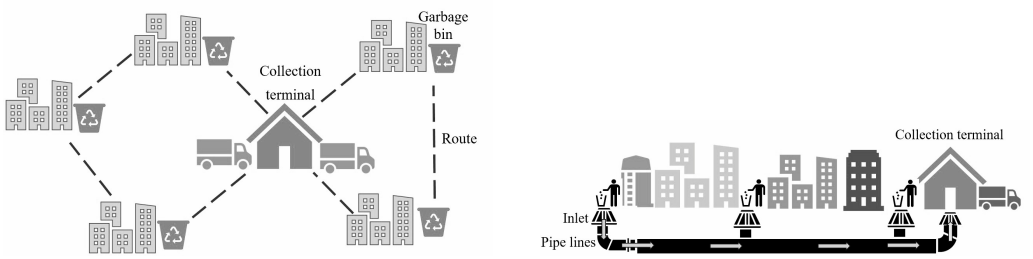
On Combining Conventional Point-To-Point and Automated Waste Collection Systems

Published in: *Networks*
On Combining Conventional Point-To-Point and Automated Waste Collection Systems.
M.DehghanChenary, R. F. Hartl, S. Irnich, & C. Tilk. © 2025 The Authors.
10.1002/net.22283

Abstract The global demand for sustainable waste management has spurred initiatives to improve the efficiency of urban waste collection, discussing the advantages and disadvantages of different systems. We analyze a new combined waste collection problem that uses two systems simultaneously: a point-to-point system, in which waste is collected using trucks, and an automated system, where waste from inlets is transported through a network of pipes. The resulting combined waste collection problem is a two-stage decision problem: At the first stage, for each collection point, it must be decided whether it is served by truck or the pneumatic system. At the second stage, a Capacitated Vehicle Routing Problem (CVRP) must be solved for the collection points served by truck, and a cost-minimal tree must be determined for the collection points assigned to the pneumatic system. Both stages and the respective problems are interdependent, making the optimization of the whole system a difficult task. We develop a holistic solution approach based on a set-partitioning formulation utilizing route and tree variables. Because of the large number of variables, we solve the formulation heuristically using a column-generation-based matheuristic. The resulting subproblems are an elementary shortest-path problem with capacity constraints and a variant of the node-weighted Steiner tree problem. Our approach is empirically evaluated on two datasets, an extended variant of the well-known CVRP benchmark and real-world data from Vienna. The results indicate that the proposed matheuristic can provide high-quality solutions to realistic instances of the combined waste collection problem.

2.1 Introduction

Sustainable waste management is a global challenge because it is linked to all three dimensions of sustainability: economic, environmental, and social. The surge in global waste generation, driven by population growth, urbanization, and industrialization, underscores the urgent need for effective waste management. The United Nations projects that more than 50% of the world's population lives in urban areas, a figure expected to reach 60% by 2030 [150]. This significant urban population emphasizes the critical role of waste collection and transportation as central components of sustainable municipal solid waste management plans for cities.



(a) Point-to-point waste collection system (b) Automatic waste collection system (AWCS)

Figure 2.1: Principle of a point-to-point waste collection system and an AWCS.

In this context, improving waste collection efficiency through a practical methodology can result in significant cost savings within current waste management systems. In addition, proper management of solid waste collection methods to reduce costs can impact the environment and human health, support economic development, and improve quality of life. Efficiency is more important now than ever. As a result, numerous municipalities, particularly those in developed countries, are being forced to evaluate the cost-effectiveness and environmental impact of their solid waste management systems, with a particular focus on optimizing waste collection routes [6].

From an economic perspective, waste collection and transportation can account for more than 70% of the total waste management budget [168]. This underscores the importance of waste management in low-income countries, and the following examples illustrate the potential of efficient waste management strategies: In Malaysia, approximately 50% of the local governmental budget is allocated to solid waste management, with more than half of this sum spent on collection activities [1]. Despite the significant cost associated with inconsistent waste management practices and policies, they often coincide with increasing environmental consequences. For example, Kinobe et al. [107] demonstrated that optimizing waste collection and disposal processes in Kampala, Uganda, could substantially reduce greenhouse gas emissions.

The *point-to-point collection system* is the most common form of waste collection that can be found in many cities, where solid waste is picked up from collection points using trucks or other vehicles, see Figure 3.1a. To optimize the routes of these vehicles [cf. 89], the predominant modeling and solution approaches represent the collection systems as *Vehicle Routing Problems* [VRPs, 172] or *Arc Routing Problems* [ARPs, 44]. Using a VRP to model waste collection problems is useful when there are single collection points scattered over an area, e.g., when citizens must bring certain types of waste from their households to dedicated waste collection points. In contrast, ARPs better model curbside collection in densely populated urban areas, e.g., door-to-door collection systems in which each household is a collection point.

Point-to-point collection systems provide large flexibility for organizing the collection process, but they come with some inherent disadvantages [108]: collection causes noise disturbances, and trucks hinder and even block inner-city traffic in narrow streets, hereby producing additional greenhouse gas emissions. Customers and garbage collectors have to cope with limited space for waste containers, adverse weather, overflowing bins, and waste accumulation, which sometimes causes pest problems.

Alternatively, *automated waste collection systems* [AWCS, 108] dispose of waste that is dropped into so-called *inlets*, which can be installed inside or outside of buildings. Waste discharge valves are then opened, and the waste falls into a pipeline system that transports it to a collection terminal. The generation of airflows by fans situated in close proximity to the containers within the collection terminal, in conjunction with the utilization of air-inlet valves located near the inlets, facilitate this process [125, 141]. At the collection terminal, each waste fraction is separated into dedicated containers for eventual transportation by truck to processing

and disposal facilities. The whole process is depicted in Figure 3.1b. Recently, AWCSs have attracted increasing attention, especially for use in modern residential areas with apartment blocks in European and Asian cities. AWCSs reduce inner-city traffic caused by waste collection trucks and can thus contribute to more sustainable waste collection.

In addition to the environmental benefits, AWCSs improve hygiene, provide easy 24/7 access for users, and require less labor [108]. Moreover, AWCSs enhance recycling rates, since they make source separation simpler and more profitable for users. Oh et al. [137] confirmed the positive impact of an AWCS in South Korea, where the AWCS reduced the general waste by 20% per capita compared to a city using conventional garbage trucks. AWCSs use less land than other waste collection systems. On the downside, building pneumatic waste collection systems presents challenges, including significant upfront investment, high energy consumption, limitations in handling large items, and concerns related to pipeline failures.

To summarize, each system has its own strengths and weaknesses, and none of the scientific articles we found has definitively favored the pneumatic system as an exclusive solution. In the following, we propose and analyze the integration of an AWCS into a point-to-point waste collection system; that is, each collection point may either be served by a truck or connected to the pneumatic system. More precisely, we consider a real-world *combined waste collection problem* (CWCP) in which citizens bring certain types of waste from their households to dedicated waste collection points. For this type of collection by truck, the VRP is an appropriate model. From an economic and strategic perspective, the design of the conventional part of the combined system can be modeled as a *capacitated vehicle routing problem* (CVRP), where trucks must operate within their capacity to serve customers, i.e., the collection points [97]. We intentionally propose a simple and stylized model in order to focus on aspects such as feasibility and cost-effectiveness. Hence, we omit practical operational and tactical extensions such as multi-day scenarios and periodic plans in the point-to-point collection system.

Thus, the CWCP is more of a strategic planning problem of sizing and the choice of the AWCS network than providing detailed routing decisions. A similar view is often taken in location routing problems, which can be defined as "location planning with tour planning aspects taken into account" [130]. As operational routing and strategic location decisions (or in our case routing and AWCS design decisions) are clearly interrelated, a separate solution of the respective planning problem bears a high risk of ending up with suboptimal solutions, see [149] for a detailed discussion.

The AWCS-related part of the combined system that uses the pneumatic pipeline is related to the *node-weighted Steiner tree problem* (NWSTP). Given a node- and edge-weighted graph as well as a subset of its nodes (called *terminal nodes*), the NWSTP is the problem of finding a tree in the graph that covers the nodes while minimizing the costs and weights of the nodes and edges of the tree. Segev [152] considers the special case of the single-terminal NWSTP, where only a single terminal node, called the *source*, must always be included in the tree as the root node. This special case reflects the challenge faced in our application: the source represents the collection terminal where the incoming waste fractions are finally processed, and the terminal nodes being served by the AWCS are not known in advance but must be determined. In our study, the size of the tree is additionally limited by a given maximum capacity, i.e., a single-terminal cardinality-constrained NWSTP has to be considered. The latter cardinality constraint reflects that an AWCS must be installed and operated within various constraints, including limited budgets, restricted space for underground pipelines, and limited energy resources for generating suction in the pipes, limiting capacity ensures better planning and optimization. Note that this special case is closely related to the *prize-collecting Steiner tree problem* [PCSTP, 118] in which the set of terminal nodes is empty.

Paper organization The literature review in Section 3.2 includes waste collection system models and comparisons. Section 3.3 formally defines the CWCP. In Section 3.4, we present the heuristic column-generation algorithm. Computational results are presented and discussed

in Section 3.6. A case study with data from the city of Vienna is presented in Section 2.6. We briefly summarize our most important findings in Section 3.7 and point out the challenges ahead for future research.

2.2 Literature review

In this section, we first provide a concise overview of the literature on solution methods for analyzing AWCSs and point-to-point waste collection systems and, subsequently, highlight studies that have compared these two systems.

2.2.1 Modelings of waste collection systems

Vehicle routing problems (VRPs) have been studied extensively for over sixty years and have received considerable attention in theoretical research and real-world applications [172]. They form the basis of many routing models and have been extended in various directions. Waste collection problems at the tactical and operational levels are often modeled as a *periodic vehicle routing problem* (PVRP). In the PVRP, a set of customers must be visited one or more times within a given time horizon. The PVRP has been examined by Cordeau et al. [45]. Early heuristics for the PVRP were first introduced by Russell and Igo [147] and Beltrami and Bodin [26], the latter in the context of a waste collection problem. We refer the interested reader to [89] for a comprehensive survey on waste collection routing problems. Due to our strategic point of view, the routing problem in the waste collection system is modeled as a CVRP with a single period. The CVRP primarily revolves around the vehicle capacity constraint, which considers volume and weight limits per trip. The problem is well studied and instances with a few hundred customers can be solved to optimality by *branch-and-price* [BaP, 139].

BaP is currently the leading method for solving many classes of VRPs exactly [49]. It is a branch-and-bound algorithm in which linear relaxations of the problem are solved by means of column generation. Column generation is an iterative method for solving linear programs with many variables. In the context of VRPs, the column generation subproblem (the pricing problem) must generate negative reduced-cost routes. The master problem is a (possibly extended) set partitioning model, while the subproblem is a variant of the *shortest path problem with resource constraints* [SPPRC, 96]. Every feasible solution to the SPPRC describes a feasible route in the VRP.

Contemporary research on AWCSs revolves around optimizing operations through innovative mobilization techniques [65, 66]. Béjar et al. [24] focused on minimizing the energy costs of AWCSs by determining the optimal combination of airspeed for waste transportation and intervals for emptying the inlets. The system is structured as a tree, originating at a root node representing the waste collection terminal. Edges represent the pipes of the pneumatic waste collection system, and nodes represent inlets or junctions where pipes are connected. In related work, Béjar et al. [23] addressed the challenge of determining collection points to be emptied at a given time by formulating it as a constrained integer programming problem. Fernández et al. [65] extended the work in [24, 23] and provided a more detailed explanation of the problem, using constrained integer programming to address a simplified problem. Later, Fernàndez Camon et al. [66] introduced an innovative approach to define operation plans for AWCSs. They modeled daily automated vacuum waste collection operations as a Markov decision process and used approximate dynamic programming to enhance the performance of pneumatic waste collection systems in terms of energy efficiency and operational costs.

2.2.2 Comparison on waste collection systems

This section focuses on studies that have compared the AWCSs with point-to-point collection systems in terms of cost and environmental impact.

The investment costs associated with AWCSs vary considerably depending on the size of the system. The sizing depends on factors such as the population served, the number of inlets, the length of the network, and the number of waste fractions collected. Miller et al. [125] detailed the costs and benefits of AWCSs and point-to-point waste collection systems, emphasizing the variability in costs and impact based on AWCS design characteristics and point-to-point system features.

Teerioja et al. [162] conducted a hypothetical economic comparison between AWCSs and point-to-point systems in densely populated Helsinki, Finland. The study concluded that the point-to-point system was five times more economical than AWCS in already established residential areas due to the substantial installation costs. The authors suggested potential cost reductions in new residential areas but emphasized critical factors such as investment costs and the economic value of land. Their paper is one of the few scientific studies focusing on the implementation of AWCSs in densely populated areas with established urban functions.

Kamga et al. [104] evaluated the feasibility of implementing an AWCS in Manhattan, New York, an already developed dense urban area, using the current transportation infrastructure. They found that the direct operating costs associated with the envisioned pneumatic system, including container transport from the pneumatic terminal to the transfer station, would be 30% lower than conventional manual/truck collection. However, the substantial initial capital investment would result in pneumatic systems costing three to six times more than conventional collection. Despite the high investment required for AWCSs, several studies [see, e.g., 131] have highlighted their lower operating costs compared to point-to-point collection systems.

Nakou et al. [131] specifically undertook a financial and economic evaluation of the implementation of an AWCS in Athens, Greece. Their results supported the preference of the AWCS over the point-to-point collection system due to its cost-effectiveness, reduced operating costs, and positive environmental impacts, particularly on air quality in the urban environment. In addition, Oh et al. [137] reported that a city using an AWCS generates less general waste per capita than a city using conventional garbage trucks. AWCSs offer environmental benefits and reduced operating costs compared to conventional systems.

Several studies have used *life cycle assessment* (LCA) methods to assess and compare different strategies in an integrated waste management system, considering global warming, waste reduction, and resource recovery. Usón et al. [169] compared an AWCS and a point-to-point collection system using diesel fuel in Valdespartera, Zaragoza, Spain. The study revealed that the AWCS had a better environmental performance than the point-to-point collection system at near or full utilization. However, at lower utilization of around 13%, the environmental benefits of the AWCS were compromised, resulting in three times higher CO_2 emissions during collection and transport compared to a point-to-point collection system.

Punkkinen et al. [141] conducted another LCA study on a hypothetical AWCS in Helsinki, Finland. Their study compared a point-to-point waste collection system with a pneumatic alternative considering different waste fractions. They identified benefits of the pneumatic system, including lower local CO_2 emissions, less congestion and noise due to reduced truck circulation, improved hygiene, and reduced contamination. However, the study also confirmed that the AWCS could be less beneficial overall than the point-to-point collection system due to its high electricity consumption and manufacturing impacts. The source of electricity plays a critical role in the overall sustainability assessment.

Chàfer et al. [37] also analyzed several waste collection systems using LCA methods, including trucks with different power sources and AWCS. They stressed the significant influence of the energy source on the LCA results, with variations of up to 80%. Due to its lower electricity consumption, the truck collection system expressed lower ecological impacts in a predominantly fossil energy framework. The AWCS performed better in environments with renewable energy sources such as hydro, wind, and various other forms of renewable energy. Additionally, Iriarte et al. [94] conducted an LCA study to quantify and compare the potential environmental impacts of three selective waste collection systems: mobile pneumatic, multi-container, and point-to-

point systems for densely populated urban areas. A sensitivity analysis showed that the mobile pneumatic system had the greatest environmental impact and the highest energy demand. They also found that greenhouse gas emissions from pneumatic collection depend on the type of material being collected.

In conclusion, no scientific study has thoroughly examined the concurrent deployment and operational integration of an AWCS with a conventional point-to-point waste collection system. Research has identified scalability and cost-effectiveness as the two main challenges in optimizing an AWCS. Scalability is influenced by variables such as the number of collection inlets, waste fraction types, pipeline length, and population density within the service area. Cost-effectiveness is determined by factors such as the type and the amount of energy required for suction, the associated CO_2 emissions, and the complexity of system installation in either newly developed urban sectors or existing residential areas.

2.2.3 Related variants of the Steiner tree problem

The subproblem of our column-generation-based matheuristic presented in Section 3.4 is a variant of the *Steiner tree problem* (STP) in graphs, a classic NP-hard combinatorial optimization problem. For a given edge-weighed graph $G = (V, E)$, non-negative edge costs $(c_e)_{e\in E}$, and a set S (the terminal nodes), the STP asks for a tree (V', E') in G with $S \subseteq V' \subseteq V$ minimizing $\sum_{e\in E'} c_e$.

Segev [152] introduced the NWSTP, which has additional (possibly negative) node weights $(p_v)_{v\in V}$ and the objective to minimize $\sum_{e\in E'} c_e + \sum_{v\in V'} p_v$ (the interpretation of p_v is that of a *penalty* (if $p_v > 0$) or a *profit* (if $p_v < 0$) resulting from the nodes of the tree; relevant only for the non-terminal nodes). The objective is equivalent to minimizing $\sum_{e\in E'} c_e - \sum_{v\notin V'} p_v$, which is, in turn, equivalent to maximizing $\sum_{v\notin V'} p_v - \sum_{e\in E'} c_e$ or $\sum_{v\in V'}(-p_v) - \sum_{e\in E'} c_e$. Also the NWSTP is strongly NP-hard, even if the set of terminal nodes is restricted to contain only a single node [152]. In this case, $S = \{r\}$ for some $r \in V$, and the tree is then a *rooted* Steiner tree (with root node r). In contrast to the STP for which constant-factor approximation algorithms exist, the NWSTP cannot be approximated within a factor better than logarithmic (unless P = NP) [46].

The *prize-collecting Steiner tree problem* [PCSTP, 27, 118] has no set S of terminal nodes but otherwise identical input data. Polynomial transformations between NWSTP and PCSTP exploit that a node $v \in V$ is certainly included in the PCSTP tree if it has a weight $p_v < -\sum_{e\in E} c_e$, see [152, p. 5].

In all competitive exact and heuristic algorithms for NWSTP and PCSTP, preprocessing a given instance is an important algorithmic component. The basic ideas were originally proposed in [57]. We notice that merging adjacent nodes (e.g., with the minimum adjacency test described in [118]) is not applicable when additional cardinality constraints are present, as for the AWCS subproblem.

Some exact approaches for NWSTP and PCSTP rely on directed problem formulations with Steiner arborescences (rooted directed trees). For example, the branch-and-cut algorithm of Ljubić et al. [118] uses such a model with an exponential family of *connectivity inequalities* (CIs). They are able to solve (preprocessed) instances with more than 10,000 edges, but preprocessing and optimization together often require minutes of computation time. In [46], the NWSTP is solved with a problem-tailored combinatorial branch-and-bound algorithm. It includes Lagrangean lower bounding procedures, a Lagrange heuristic, and a specific branching scheme. Also here preprocessing is essential and consumes substantial computation time.

The branch-and-cut algorithm for a variant of the NWSTP in [36] integrates a constraint that ensures a minimum quantity to be collected by the tree nodes V'. (This is opposed to our cardinality constraint.) As in our application, a large number of instances has to be solved within a branch-and-price-type of matheuristic. Therefore, like our approach (see Section 3.4.2), the approach in [36] must rely on heuristic pricing procedures.

2.3 The combined waste collection problem

The CWCP that we study can be defined on an undirected complete graph (V, E) with node set V and edge set E. The node set V comprises $\{o^T, o^R\} \cup C$, where o^T represents the collection terminal of the AWCS, o^R represents the point where trucks start and end their tour, and $C = \{1, 2, \ldots, n\}$ denotes the set of collection points. To simplify the wording, we use the shorter terms *source* for o^T, *depot* for o^R, and *customers* for (elements of) C in the following. Each customer $i \in C$ has a positive integer *demand* q_i, which represents the quantity of waste that has to be collected when served by truck (*not* relevant for the pneumatic network). A number K of homogeneous trucks, each with a capacity of Q^R, is available at the depot o^R for point-to-point collection. The pneumatic network is modeled as a cardinality-constrained tree rooted at the source o^T. It can serve up to Q^T customer irrespective of their demand. Each edge $\{i, j\} \in E$ is associated with two cost values: the routing cost c_{ij} for the trucks and the cost d_{ij} for installing and using the pneumatic system on the edge. We assume that the triangle inequality holds for both (c_{ij}) and (d_{ij}). It is important to note that these costs do not solely encompass direct monetary expenses; they can also reflect the environmental damage caused. The objective is to find a set of feasible vehicle routes and at most one feasible rooted tree that serve all customers at minimum cost.

The CWCP is the following two-stage decision problem: On the first stage, assign each customer to one of the subsystems, i.e., to the pneumatic system or to the point-to-point collection system served by trucks. Not more than Q^T customer nodes can be assigned to the pneumatic system. In the second stage, solve a CVRP with a fleet of K vehicles for the truck customers and construct a minimum-cost tree rooted at o^T for the remaining customers. Figure 3.2 shows the partitioning of the customer set C into the two subsets (blue and red points) and two routes and a tree covering these two subsets. Clearly, the first-stage and second-stage decisions are interdependent, making the CWCP at least as difficult as the CVRP and the NWSTP.

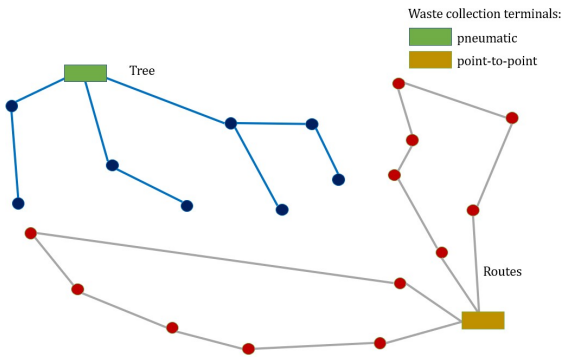


Figure 2.2: Schematic of the CWCP.

More formally, a rooted tree t is a cycle-free connected subgraph (V', E') of (V, E) which contains the source, i.e., $o^T \in V'$. In what follows, we only consider rooted trees and can simply speak of a *tree*. A tree is feasible, if $|V' \cap C| \leq Q^T$. The cost d_t of a feasible tree t covering a given set of nodes $V' = \{o^T, i_1, \ldots, i_m\}$ can be computed by constructing a *minimum spanning tree* (MST) over $\{o^T, i_1, \ldots, i_m\}$ since the graph is complete and the triangle inequality holds for (d_{ij}). A route $r = (r_0, r_1, \ldots, r_m, r_{m+1})$ starts and ends at the depot, i.e., $r_0 = r_{m+1} = o^R$, and visits the customers $r_1, \ldots, r_m \in C$ in the given order. It is feasible, if $\sum_{j=1}^{m} q_{r_j} \leq Q^R$. The cost of a route is $c_r = \sum_{j=0}^{m} c_{r_j, r_{j+1}}$. Let T denote the set of feasible trees, and let R denote the set of feasible routes.

Next, we state the set-partitioning formulation of the CWCP that constitutes the basis of the column-generation-based matheuristic. The model features two types of variables, i.e., binary variables Υ_t for each feasible tree $t \in T$ and binary variables λ_r for each feasible route $r \in R$. Moreover, parameters a_{ir} and b_{it} indicate whether customer $i \in C$ is served by route $r \in R$ or by tree $t \in T$, respectively.

$$z_{CWCP} = \min \sum_{t \in T} d_t \Upsilon_t + \sum_{r \in R} c_r \lambda_r \tag{2.1a}$$

$$\text{subject to} \sum_{t \in T} b_{it} \Upsilon_t + \sum_{r \in R} a_{ir} \lambda_r = 1 \qquad \forall i \in C \tag{2.1b}$$

$$\sum_{t \in T} \Upsilon_t \leq 1 \tag{2.1c}$$

$$\sum_{r \in R} \lambda_r \leq K \tag{2.1d}$$

$$\Upsilon_t \in \{0, 1\} \qquad \forall t \in T \tag{2.1e}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in R \tag{2.1f}$$

The objective (3.1a) minimizes the total cost. Constraints (3.1b) ensure that each customer is either served by the tree or one of the routes. The constraint (3.1c) states that at most one tree is selected. Similarly, constraint (3.1d) is the fleet-size constraint. Note that exclusive collection by truck is possible. The domain of the tree and route variables is stated in (3.1e) and (3.1f).

2.4 The column-generation-based matheuristic

Small instances of the CWCP can be solved with a BaP algorithm [54] which is a branch-and-bound approach that uses column generation to solve in each node of the search tree the linear relaxation of Formulation (3.1). Column generation considers smaller linear relaxations, known as *restricted master programs* (RMPs), in which the sets of feasible routes and trees (R and T) are replaced by subsets $\bar{R} \subset R$ and $\bar{T} \subset T$. An RMP is solved, variables (columns) with negative reduced cost are identified and added to the RMP, and this process is repeated until no variables with negative reduced cost exist, which is proven by solving the subproblems to optimality. Compared to many other BaP approaches for variants of the CVRP, the difficulty is here the solution of the subproblem that identifies negative reduced-cost trees. For larger problem instances as those that we consider in Sections 3.6 and 2.6, it will turn out that the exact solution of the tree subproblems is too time-consuming. Therefore, we use a matheuristic algorithm in which the tree subproblems are solved heuristically. A heuristic pricing approach has also been used in [36], where many instances of a variant of the NWSTP had to be solved. We mention here that our NWSTP instances often have more than 10,000 edges and that the established preprocessing techniques alone would consume substantial computing time (note that not all are applicable here). Thus, the exact solution appears impractical in our context, even for relatively small instances (see Section 2.5.1).

We start by describing the tree-generation approaches in Section 3.4.2, followed by the labeling algorithm for route generation in Section 3.4.1, branching in Section 3.4.3, and the entire matheuristic in Section 2.4.4. For a given RMP, let $(\pi_i)_{i \in C}$ be the dual prices of the partitioning constraints (3.1b), μ_T be the dual price of the convexity constraint (3.1c), and μ_R be the dual price of the fleet-size constraint (3.1d).

2.4.1 Tree generation

The task of the tree-generation problem is to identify feasible trees $t \in T$ with negative reduced cost $\bar{d}_t = d_t - \sum_{i \in C} b_{it}\pi_i - \mu_T$. For this purpose, a cardinality-constrained NWSTP in which the source is the only terminal node must be solved. We present an adapted version of the single-commodity flow formulation of the NWSTP by Segev [152], in which trees are modeled as *directed* trees in the corresponding complete digraph (V, A). In particular, all edges $\{i, j\} \in E$ of the underlying graph (V, E) are replaced by pairs of anti-parallel arcs (i, j) and (j, i) with identical cost. The set of arcs that enter and leave node $i \in V$ are denoted by $\delta^-(i)$ and $\delta^+(i)$, respectively. Three types of variables are used to describe a directed tree, in which each tree node has in-degree 1 except for the source o^T which has in-degree 0: (i) binary variables u_i for $i \in C$ indicate whether node $i \in V$ belongs to the tree, (ii) binary variables x_{ij} indicate whether arc (i, j) belongs to the tree, and (iii) continuous variables f_{ij} describe the flow on arc (i, j).

$$\min \sum_{(i,j) \in A} d_{ij} x_{ij} - \sum_{i \in C} \pi_i u_i - \mu_T \tag{2.2a}$$

$$\text{subject to} \sum_{(j,i) \in \delta^-(i)} x_{ji} = u_i \qquad \forall i \in C \tag{2.2b}$$

$$\sum_{(j,i) \in \delta^-(i)} f_{ji} - \sum_{(i,j) \in \delta^+(i)} f_{ij} = u_i \qquad \forall i \in C \tag{2.2c}$$

$$f_{ij} \leq Q^T x_{ij} \qquad \forall (i,j) \in A \tag{2.2d}$$

$$u_{o^T} = 1 \tag{2.2e}$$

$$u_{o^R} = 0 \tag{2.2f}$$

$$u_i \in \{0, 1\} \qquad \forall i \in C \tag{2.2g}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A \tag{2.2h}$$

$$f_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{2.2i}$$

The objective (2.2a) minimizes the reduced cost of the constructed tree. Constraints (2.2b) state that if a node $i \in C$ belongs to the tree, then there must be exactly one arc entering it. Flow-conservation constraints (2.2c) and tree-size constraints (2.2d) prevent cycles and enforce together with constraints (2.2b) a tree structure. Constraint (2.2e) ensure that the source node o^T is included in the tree. Similarly, constraint (2.2f) excludes node o^R. The domains of the variables are stated in (2.2g)–(2.2i).

Computational results in Section 3.6 show that solving this formulation directly with a MIP solver (CPLEX) is too time-consuming. Especially, as the model needs to be solved repeatedly. An alternative would be the use of branch-and-cut algorithm using a formulation with exponential family of cycle-elimination constraints, see Section 2.2.3. Such an approach appears limited to (very) small instances, as explained above. Therefore, we use the following greedy heuristics based on Prim's algorithm and swap-based local search algorithms.

Greedy heuristics

Given a weighted undirected graph (V, E, d_{ij}), Prim's algorithm for MST starts with a singleton set $S = \{i\}$ for an arbitrary $i \in V$ and iteratively extends S by a node j. The next

node j is chosen by selecting an edge $\{i, j\}$ with $i \in S$ and $j \notin S$ having smallest cost d_{ij}. The selected edges define the resulting tree.

We use a modified version of Prim's algorithm to heuristically solve the subproblem (2.2). In this heuristic, the set S always contains the source o^T. To incorporate the dual prices $(\pi_i)_{i \in C}$, we define *modified costs* $p_{ij} = d_{ij} - \pi_j$. Note that this definition is asymmetric with respect to i and j so that we will consider arcs $(i, j) \in A$ as defined above instead of edges. As before, we extend the set S iteratively by adding a node j to S. We choose the node $j \in C \setminus S$ that minimizes the modified cost p_{ij} over all arcs $(i, j) \in A$ with $i \in S$ and $j \in C \setminus S$. To ensure feasibility, we stop when the tree contains Q^T customer nodes. The reduced cost of the resulting tree is the difference of sum of the modified costs of the included arcs and μ_T.

Since this greedy heuristic is extremely fast, we repeat it for all customers $i \in C$ setting the initial set to $S = \{o^T, i\}$, i.e., the first added arc is $(o^T, i) \in A$. Note that an optimal tree may contain less than Q^T customers. Therefore, we consider all intermediate trees in which the sum of the modified cost is smaller than μ_T, i.e., the current tree has a negative reduced cost.

Local search

We apply the following local search to all trees $t \in \bar{T}$ whose variables Υ_t are in the basis of the current RMP. The underlying neighborhood allows (1,1)-swaps and (2,2)-swaps, i.e., the exchange of one (two) customers included in the tree t with one (two) customers not in the tree. For this purpose, let $C(t) \subset C$ denote the customers included in the tree $t \in T$. Both neighborhoods contain only feasible moves, since the number $|C(t)|$ of customers remains unchanged. To compute the reduced cost of a neighbor solution t', we solve an MST on $\{o^T\} \cup C(t')$ with Prim's algorithm subtracting $\sum_{i \in C(t')} \pi_i$ and μ_T. As before, we consider all trees with negative reduced cost found during the local search.

The size of the (1,1)-swap neighborhoods is quadratic so that it can be explored completely. In contrast, the size of the (2,2)-swap neighborhood is $\mathcal{O}(|C|^4)$ so that a complete local search is too time-consuming, especially when applied repeatedly. Therefore, we restrict (2,2)-swaps to the most promising swaps considering the dual prices π_i of the involved nodes. More precisely, for a given tree t, we compute for each combination (i, j, k, l) with $i, j \in C(t)$, $k, l \notin C(t)$, and $i \neq j, k \neq l$ the improvement in terms of dual prices, i.e., $\pi_i + \pi_j - \pi_k - \pi_l$. Only the 2000 combinations with the smallest value are tested by computing the associated MST.

2.4.2 Route generation

The task of the route-generation subproblem is to identify feasible routes $r \in R$ with negative reduced cost $\bar{c}_r = c_r - \sum_{i \in C} a_{ir} \pi_i - \mu_R$. To generate negative reduced cost routes, a *shortest path problem with resource constraints* (SPPRC) has to be solved [96]. The SPPRC is a well-known subproblem in column-generation algorithms and numerous articles on BaP for variants of the VRP describe solution approaches [see 49]. Dynamic programming labeling algorithm are the most effective ones. As the topic is well-known and has been widely studied, we only sketch the techniques used here for the CWCP.

Labeling algorithms iteratively extend partial paths by moving forward from an origin to a destination node in the given network. A label stores the necessary information on the resource consumption up to the endpoint of a partial path. Labels are propagated along the network arcs by *resource extension functions* [REFs, 95]. Dominance procedures are invoked to identify and discard those partial paths and their labels that cannot lead to an optimal subproblem solution.

For the CWCP, a label $\mathcal{L}$ associated with a partial path $p = (o^R, \ldots, i)$ comprises the resources $(T_i^{rdc}, T_i^{load}, T_i^{vis,k})$ defined as follows:

T_i^{rdc}: the reduced cost of path p;
T_i^{load}: the total demand served by path p; and
$T_i^{vis,k}$: the number of times that customer $k \in C$ is served along the path p.

The initial label $\mathcal{L}_{o^R}$ is given by the resource values $(0, 0, \mathbf{0})$. Propagating a label $\mathcal{L}_i$ referring to node i over an arbitrary edge $\{i, j\}$ towards a node j results in a new label $\mathcal{L}_j$ at j. For convenience, we define $\pi_{o^R} = \mu_R$ and $q_{o^R} = 0$. The associated resource values for the partial path associated with $\mathcal{L}_j$ are computed with the help of the following REFs:

$$\begin{aligned} T_j^{rdc} &= T_i^{rdc} + c_{ij} - (\pi_i + \pi_j)/2 \\ T_j^{load} &= T_i^{load} + q_j \\ T_j^{vis,k} &= \begin{cases} T_i^{vis,k} + 1, & \text{if } k = j \\ T_i^{vis,k}, & \text{otherwise} \end{cases} \end{aligned}$$

The new label $\mathcal{L}_j$ is feasible if the capacity and the elementary constraints are fulfilled, i.e., $T_j^{load} \leq Q^R$ and $T_j^{vis,k} \leq 1$ for all $k \in C$. To avoid enumerating all feasible paths, provable redundant labels are eliminated through a dominance procedure. Since all resources are non-decreasing and resources are bounded only from above, we can use the standard dominance procedure: A label $\mathcal{L}_{i,1} = (T_{i,1}^{rdc}, T_{i,1}^{load}, T_{i,1}^{vis,k})$ is dominated and can be discarded if there exists a label $\mathcal{L}_{i,2} = (T_{i,2}^{rdc}, T_{i,2}^{load}, T_{i,2}^{vis,k})$ which ends at the same node i and has not larger resource consumption, i.e., $T_{i,2}^{rdc} \leq T_{i,1}^{rdc}$, $T_{i,2}^{load} \leq T_{i,1}^{load}$, and $T_{i,2}^{vis,k} \leq T_{i,1}^{vis,k}$ for all $k \in C$.

Bidirectional labeling has become a quasi-standard for solving SPPRCs [145, 164]. For the CWCP, defining backward labels is trivial because the underlying graph is symmetric. Intentionally, we also defined the above REFs symmetrically. All forward labels can be directly interpreted as backward labels (such an implicit bidirectional labeling was, e.g., described in [29]). We use the resource T^{load} as the monotone resource, which means only labels that fulfill the half-way condition $T^{load} \leq Q^R/2$ are extended. The merge procedure considers all nodes $i \in C \cup \{o^R\}$ and merges pairs of forward labels $\mathcal{L}_{i,1} = (T_{i,1}^{rdc}, T_{i,1}^{load}, T_{i,1}^{vis,k})$ and $\mathcal{L}_{i,2} = (T_{i,2}^{rdc}, T_{i,2}^{load}, T_{i,2}^{vis,k})$ if (i) $T_{i,1}^{load} + T_{i,2}^{load} - q_i \leq Q^R$, (ii) $T_{i,1}^{vis,k} + T_{i,2}^{vis,k} \leq 1$ for all $k \in C \setminus \{i\}$, and (iii) $T_{i,1}^{rdc} + T_{i,2}^{rdc} < 0$. The last condition ensures that only routes with a negative reduced cost are generated.

To further accelerate the solution process, we use the ng-path relaxation [18], which is a parameterized relaxation that allows some non-elementary routes according to predefined neighborhoods for each node. We define neighborhoods for each node $i \in C$ that contains the ten nearest customers according to the routing costs.

In addition, we use the following partial pricing: We solve the route generation problem heuristically on reduced networks with up to 5, 10, and 15 ingoing and outgoing arcs per node. Only if all three heuristics fail, the route generation problem is solved over the complete network. The arcs in the reduced networks are chosen according to the lowest reduced cost in the current column-generation iteration.

2.4.3 Branching

Let $(\Upsilon_t^*, \lambda_r^*)$ be a fractional primal solution of the current RMP. We first branch on the total number of routes if $K^* = \sum_{r \in R} \lambda_r^*$ is fractional. Two children nodes result from the constraints $\sum_{r \in R} \lambda_r \leq \lfloor K^* \rfloor$ and $\sum_{r \in R} \lambda_r \geq \lceil K^* \rceil$. While the first constraint simply replaces the fleet-size constraint (3.1d) of the RMP, the second one introduces a new constraint in the RMP with a non-positive dual price, which can be handled in the same manner as μ_R.

If the number of routes is integral, we branch on edges enforcing either that the edge is used by the tree or a route, or not used at all. To this end, we compute, for each edge $\{i, j\} \in E$, the value

$$e_{ij}^* = \sum_{t \in T} g_{ijt} \Upsilon_t^* + \sum_{r \in R} h_{ijr} \lambda_r^*,$$

where binary indicators g_{ijt} and h_{ijr} have the value 1, if and only if route r or tree $t \in T$ contains the edge (or one of its corresponding anti-parallel arcs), respectively. We select an

Algorithm 1: Matheuristic

```
1  Set T̄ ← ∅, R̄ ← {r_o}, 𝒩 ← {N_o}, (Υ•_t, λ•_r) ← ∅, UB ← ∞
2  while 𝒩 ≠ ∅ and time limit not reached do
3      select next B&B node N ∈ 𝒩 and drop it from 𝒩
4      repeat
5          (z_RMP, (Υ*_t, λ*_r), π, μ_T, μ_R) ← solve RMP(N, T̄, R̄) as LP
6          T_new ← Generate trees(π, μ_T, Υ*_t)                    // Section 4.1
7          add T_new to T̄
8          R_new ← Generate routes(π, μ_R)                         // Section 4.2
9          add R_new to R̄
10     until T_new ∪ R_new = ∅
11     if (Υ*_t, λ*_r) is integral then
12         if z_RMP < UB then
13             UB ← z_RMP, (Υ•_t, λ•_r) ← (Υ*_t, λ*_r)
14     else
15         (z_MIP, (Υ*_t, λ*_r)) ← solve RMP(N, T̄, R̄) as MIP
16         if z_MIP < UB then
17             UB ← z_MIP, (Υ•_t, λ•_r) ← (Υ*_t, λ*_r)
18         (N_1, N_2) ← Branching(N)                               // Section 4.3
19         add N_1 and N_2 to 𝒩
20 return ((Υ•_t, λ•_r), UB)
```

edge with a value e^*_{ij} closest to 0.5. The zero-branch can be enforced directly on the underlying network by removing the edge $\{i, j\}$ and the arcs (i, j) and (j, i) from the respective graphs. Trees and routes incompatible with this branching decision are temporarily removed from the RMP. The one-branch introduces a new constraint $\sum_{r\in R} g_{ijr}\lambda_r + \sum_{t\in T} h_{ijt}\Upsilon_t \geq 1$ into the RMP. Its dual price can be directly incorporated into the modified costs in the tree-generation procedures (Sections 3.4.2 and 2.4.1) and into the reduced cost of arcs in the labeling algorithms (Section 3.4.1).

2.4.4 The matheuristic

As usually done in column-generation algorithms, we use the current dual prices to guide our search for promising trees and routes as explained in Sections 3.4.2 and 3.4.1, respectively. The method is embedded in a branch-and-bound framework, where at each node of the branch-and-bound tree, a feasible solution and an upper bound are computed with the help of a MIP solver using the (integer) Formulation (3.1) with the variables generated up to this point. Since the pricing problem for generating trees is solved only heuristically, the LP bounds computed at each node of the branch-and-bound tree are not valid lower bounds; in the following, we call them *approximate lower bounds*.

A pseudo code of the matheuristic is given in Algorithm 1. In the first line, the root node N_0 of the branch-and-bound tree is created as one with an RMP in which only a single ("dummy") route r_o covering all customers at a high cost is present, i.e., $a_{ir_o} = 1$ for all $i \in C$ and $c_{r_0} = M$ using a large cost value M. Additionally, initial values are assigned to the restricted set of trees and routes, the open search tree nodes $\mathcal{N}$, and the *best known solution* (BKS) $(\Upsilon^\bullet_t, \lambda^\bullet_r)$ with cost UB. The main-loop starts in Line 2 and runs as long as there are branch-and-bound nodes to process and the given time limit is not reached. In Line 3, the next node to process is chosen as one with the smallest approximate lower bound. In the inner loop (Lines 4–10), this

branch-and-bound is processed in the following way. First, the RMP is solved with the primal simplex algorithm of CPLEX resulting in the objective value z_{RMP} and new dual prices (Line 5). Second, new route and tree variables with negative reduced cost are generated and added to the RMP (Lines 6–9). The procedure is repeated until no more variables with negative reduced cost can be found. If the solution of the branch-and-bound node is integer and improving, we update the BKS. Otherwise, we solve the current RMP to integer optimality with a MIP solver and update the BKS if possible. Moreover, branching is applied.

2.5 Computational results

The following experiments were conducted on the high performance computing cluster MOGON II of Johannes Gutenberg University Mainz using computing nodes equipped with Intel Xeon E5-2630v4 (Broadwell) processors running at 2.2GHz. The performance of a single thread of the cluster is slightly worse compared to that of a standard desktop processor. All algorithms are coded in C++ and compiled with GCC 11.2.0 in release mode into a 64-bit executable. The callable library of CPLEX 20.1.0 is used for solving the RMPs and for the MIP-based heuristic. CPLEX's default parameters are retained, except that the number of threads is set to one. The computation time for the matheuristic is limited to 3600 seconds per instance, and the MIP solver is given an additional 600 seconds to solve the RMP of the last branch-and-bound node processed.

In the following, we first analyze the performance of the matheuristic on small instances in Section 2.5.1. In Section 2.5.2, we evaluate the impact of different capacity and cost values of the AWCS, and we study the capacity utilization of the tree in the combined collection system. Finally, we analyze different demand scenarios in Section 2.5.3.

2.5.1 Performance of the matheuristic

To assess the performance of the proposed matheuristic, we compare it with an exact BaP algorithm on small-sized instances some of which can be solved to optimality. The exact BaP algorithm uses the MIP solver CPLEX to compute trees with minimum negative reduced cost using a multi-commodity-flow formulation. The BaP algorithm uses the same settings as the matheuristic, especially the identical computation time limit. To generate instances, we use the *set A* instances proposed by Augerat [15] from the CVRPLIB and make the following additional assumptions: The first customer of each instance is the source o^T of AWCS. Additionally, we set the cost of the tree edges to $d_{ij} = 2c_{ij}$ and the capacity of the tree to $Q^T = |C|/3$ (we always round down the capacity to the next integer value to tighten the models but do not write $\lfloor |C|/3 \rfloor$ for the sake of simplicity).

Size	no.	no.	time					ΔUB		
$\|C\|$	inst.	opt.	BaP	%MCF	%Prim+LS	%SPPRC	MH	min	avg	max
< 50	15	11	2038.6	91	3	5	700.6	−3.09	−0.31	0.36
≥ 50	12	0	3965.5	97	1	2	2318.6	−1.58	−0.38	0.00

Table 2.1: Comparison of the exact BaP algorithm and the matheuristic.

Table 2.1 presents aggregated results where we considered two subsets of the instances depending on their size given by $|C|$. The column entries have the following meaning:

no. inst.: the number of instances in the subset
no. opt.: the number of instances solved to optimality by BaP algorithm
time BaP: the average computation time of BaP; all times are given in seconds

time %MCF: the average percentage of the time it takes to solve the tree pricing problem exactly in the BaP

time %Prim+LS: the average percentage of the time it takes to solve the tree pricing problem heuristically in the BaP

time %SPPRC: the average percentage of the time it takes to solve the route pricing problem in the BaP

time MH: the average computation time of the *matheuristic* (MH)

ΔUB: the minimum/average/maximum difference of the upper bounds of the BaP and the matheuristic in percent, computed as $100 \cdot (UB_{MH} - UB_{BaP})/UB_{BaP}$ where UB_{BaP} and UB_{MH} is the upper bound computed by the BaP algorithm and the matheuristic, respectively

Only eleven of the 27 instances are solved to optimality with the BaP, all with less than 50 customers. The matheuristic finds nine of these optima with an average relative difference of 0.3% for the remaining two. For the other 16 instances, it finds a better solution in ten cases with a maximum relative difference of -3.1%. Note that the matheuristic always computes a better (or equivalent) solution than the BaP algorithm on the instances with more than 50 customers.

The bottleneck of the BaP algorithm is the exact tree generation component (CPLEX): on average, 91% (97%) of the total computation time is consumed by solving the tree pricing problem exactly in the smaller (larger) instances. For comparison, the heuristic and exact solution of the route generation subproblem as well as the heuristic solution of the tree generation pricing problem take together less than 8(3)% of the total computation time in the smaller (larger) instances. Hence, the matheuristic takes significantly less time than the BaP.

2.5.2 Impact of AWCS capacity and cost ratio tree/route

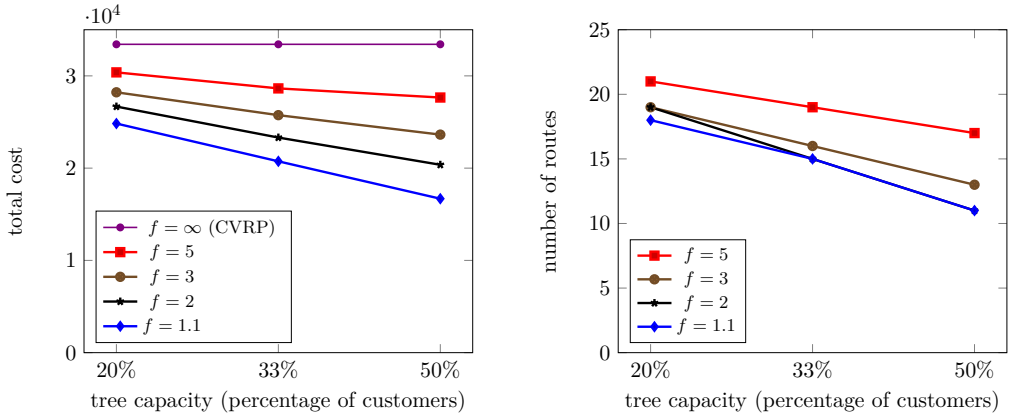
In this section, we conduct experiments that show the impact of two important parameters on the structure and total cost of solutions: (1) the capacity of the AWCS and (2) the cost ratio between tree costs and routing costs. For the latter, we vary the ratio of the costs d_{ij} and c_{ij} associated with the edges in the two subsystems. As in the previous section, we use instances from the CVRPLIB dataset, now 45 larger instances provided by Uchoa et al. [167] with sizes ranging from 100 to 308 nodes.

For the capacity of the AWCS, we analyze three tree capacity scenarios. As before, we express this capacity as a percentage of the customers that could be served by the pneumatic system. The considered percentages are 20%, 33%, and 50%, i.e., $Q^T = |C|/5$, $Q^T = |C|/3$, and $Q^T = |C|/2$.

The proportion of the tree and routing costs is certainly difficult to estimate, especially as a time horizon must be determined. However, the literature provides some perspectives on this cost ratio:

- Nakou et al. [131] calculate with €214 per meter for the AWCS and an average routing cost of €194 per meter in the conventional point-to-point collection system. They assume a working life of about 30 years for the AWCS. This results in a cost ratio of $f = 1.1$.
- Teerioja et al. [162] use a similar argument and arrive at a cost ratio of $f = 5$. The significant cost difference in their case study, Helsinki, Finland, is mainly attributed to investment costs. They highlight that land value plays a key role in waste collection, with their results being particularly relevant to old established areas. They acknowledge that in new residential developments, the cost gap between systems may narrow, as pneumatic systems are easier and cheaper to install in new construction. Interestingly, they also show that the AWCS has 40% lower collection costs.

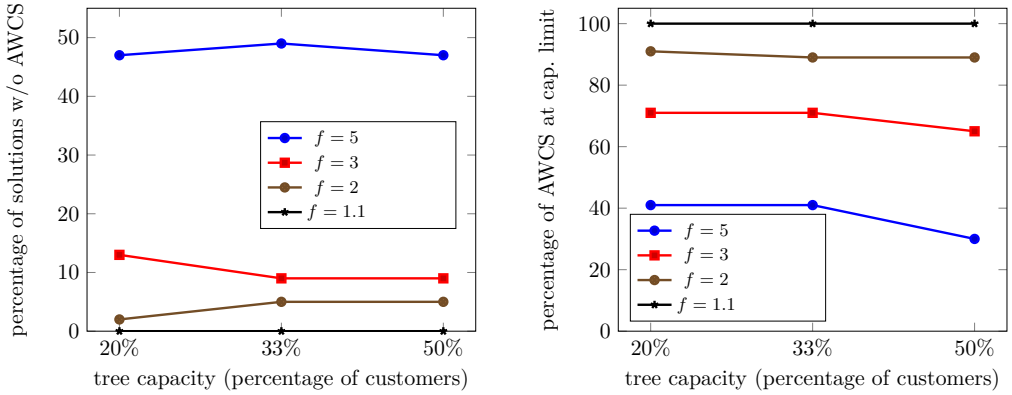
To complete the picture, we add two intermediate cost ratios $f = 2$ and $f = 3$ as well as the extreme ratio $f = \infty$, which represents a pure truck-based point-to-point collection system (labeled with CVRP below). For $f = \infty$, we took the best-known solution values from the CVRP library.



(a) Impact on total cost.

(b) Impact on the number of routes.

Figure 2.3: Impact of the tree capacity Q^T and the cost ratio f.



(a) Percentage of instances in which the AWCS is not used.

(b) Percentage of instances in which the AWCS is at the capacity limit.

Figure 2.4: Utilization of AWCS.

Figure 2.3 visualizes the aggregated result over all 45 instances (each one solved $12 = 3 \cdot 4$ times). Figure 2.3a shows that even in the most pessimistic scenario (cost ratio of $f = 5$), the total cost remains lower than that of a conventional point-to-point collection system (the CVRP solution). Figure 2.3b shows a significant difference in the number of routes when comparing the cost ratios 1.1, 2, and 3 to the most pessimistic scenario $f = 5$, where the latter indicates a greater willingness to use conventional trucks and to assign fewer customers to the AWCS. Detailed results for all instances can be found in Section 2.8.1 of the Appendix.

The averages in Figure 3 do not show that, in several instances, the AWCS is either completely unused or operating at the capacity limit. Therefore, Figure 2.4 shows these metrics for the same twelve settings from the above analysis (three capacities and four cost ratios; CVRP omitted). For cost ratio $f = 1.1$, the AWCS is always used, and its capacity is fully utilized. Note that when the tree capacity is fully relaxed, all customers are served by the AWCS (the solution is an MST), as we observed in additional experiments. From the cost ratio $f = 2$ onwards, the AWCS is utilized to a progressively lesser extent, as shown in Figure 2.4a. While the percentage

of solutions without AWCS remains below 15% for $f = 2$ and $f = 3$, this number rises to nearly half of the instances for an expensive pneumatic system at $f = 5$.

Figure 2.4 shows another interesting point: For moderate cost ratios $f \leq 3$, the percentage of instances where the AWCS operates at full capacity is almost constant. Increasing the capacity Q^T from 20% to 33% and 50% does not have much effect. Our interpretation of this result is as follows: Whether the AWCS is beneficial for a large portion of customers depends on the individual instance, e.g., the demand distribution. We investigate this next.

2.5.3 Impact of demand distribution

The CVRP benchmark set of Uchoa et al. [167] was designed to provide a comprehensive and balanced experimental setting. In particular, the instances have a number of different characteristics that allow us to identify five specific groups that have similar demand distributions and ratios of demands q_i for $i \in C$ and vehicle capacity Q^R.

We compute the following indicator to highlight the impact of demand distributions and capacity to demand ratios have:

tree %cust.: the average percentage of customers served by the AWCS

tree %demand.: the average percentage of the demand covered by the AWCS

total %Δcost: the average percentage increase of the total cost compared to the *baseline* (BL) scenario where BL refers to the cost ratio $f = 1.1$, i.e., the scenario in which installing and operating the AWCS is the cheapest

In total, the benchmark contains 45 instances. Note that the six groups do not span the entire benchmark of all 45 instances, and some groups overlap slightly (see Section 2.8.2 of the Appendix for a list specifying each group). We consider the instances in twelve scenarios (three tree capacities and four cost ratios f).

The aggregated values, which we now discuss separately for each group, are presented in Table 2.2:

SL (Small and Large demands): Six instances in which 70 to 95% of the customer demands are taken from the *uniform discrete distribution* (UD) ranging from 1 to 10 (hereafter written as $UD[1, 10]$). The remaining demands are $UD[50, 100]$.

The AWCS covers a much larger percentage of the demand than the other groups, although the percentage of customers served by AWCS is similar to the other groups. Moreover, it is substantially higher than the percentage of customers covered by the AWCS. For example, for the cost ratio $f = 5$, all instances (*Total*) cover on average between 10% and 19% of the demand, while **SL** covers between 30% and 37%. In comparison, the percentage of customers covered is similar, ranging from 10% to 21%. In addition, this group is less robust to changes in the cost ratio, as it has the largest increase in total costs.

U (Uniform): Six instances with unit demand, i.e., $q_i = 1$ for all $i \in C$.

In comparison, group **U** has the smallest cost increases *total* %Δ*cost*. The percentages of demand and customers covered do not differ from the full set of instances. Thus, unit demand makes the waste collection system more robust to changes in the cost ratio.

SCap (Small Vehicle Capacity): Seven instances with an average demand of approximately 25% of the vehicle capacity Q^R.

The AWCS is fully utilized here for all capacities Q^T and all cost ratios f. For $f = 3$ and $f = 5$, the fraction of the demand and of the customers covered by the AWCS greatly exceeds that of the full set (*Total*), while the total cost increase %Δ*cost* is similar.

LCap (Large Vehicle Capacity): Seven instances where the average demand is only 5% of the vehicle's capacity.

For cost ratios $f = 3$ and 5, the AWCS serves fewer customers and covers a smaller fraction of the demand. In particular, the AWCS remains almost unused for the cost ratio $f = 5$. Interestingly, the total cost increase for larger cost ratios is similar to that of the group **SCap** and the entire set (*Total*).

Group	cost ratio f	tree capacity 20% tree %cust.	tree %dem.	total %Δcost	tree capacity 33% tree %cust.	tree %dem.	total %Δcost	tree capacity 50% tree %cust.	tree %dem.	total %Δcost
SL	1.1	20	60	BL	33	65	BL	50	73	BL
(6)	2	18	48	+14	29	55	+19	43	62	+27
	3	18	43	+26	26	50	+34	29	45	+48
	5	12	30	+43	16	31	+54	21	37	+70
U	1.1	20	20	BL	33	33	BL	50	50	BL
(6)	2	20	20	+5	32	32	+8	47	47	+18
	3	16	16	+9	30	30	+16	44	44	+34
	5	10	10	+13	16	16	+27	21	21	+52
SCap	1.1	20	20	BL	33	33	BL	50	50	BL
(7)	2	20	20	+5	33	33	+9	50	50	+20
	3	20	20	+11	33	33	+21	50	50	+40
	5	20	20	+20	33	33	+39	50	50	+79
LCap	1.1	20	24	BL	33	33	BL	50	51	BL
(7)	2	20	23	+7	32	31	+11	47	46	+24
	3	13	16	+13	22	23	+24	21	23	+40
	5	0	0	+21	5	4	+33	7	6	+59
Q	1.1	20	20	BL	33	33	BL	50	50	BL
(6)	2	20	20	+5	33	33	+7	50	50	+16
	3	18	18	+9	30	30	+15	43	43	+33
	5	15	15	+16	24	24	+30	34	34	+58
Total	1.1	20	23	BL	33	36	BL	50	51	BL
(45)	2	20	22	+7	31	35	+12	47	49	+22
	3	16	17	+13	27	28	+24	36	32	+41
	5	10	10	+22	16	14	+38	21	19	+65

Table 2.2: Impact of varying demand on the percentage of the average number of customers and total demand covered by the AWCS.

Q (Quadrant): Six instances where the demand of a customer depends on the quadrant in which the customer is located. All customers are located on a 1000×1000 grid, and (500,500) is the point of origin. If a customer is located in an even quadrant, its demand is drawn from $UD[1, 50]$, otherwise from $UD[51, 100]$.

This type of demand distribution results in some significantly longer routes in terms of the number of customers served. The AWCS covers more demand compared to *Total* and group **LCap**, consistently for all three capacities and cost ratios $f = 3$ and 5. Moreover, the AWCS covers more customers than the other groups, with the exception of **SCap**, for which the coverage is always complete. Compared to *Total*, the total cost increase $\%\Delta cost$ is slightly lower when the cost ratio f is increased.

A visualization of the results is shown in Section 2.8.2 of the Appendix.

In summary, the AWCS is a valuable addition to the traditional truck-based point-to-point collection system for the groups **SL** and **SCap**, i.e., when some customers have relatively high demand, and the vehicle capacity is small compared to the demand. Even at the highest cost ratio of $f = 5$, which represents the most challenging scenario for the use of the AWCS, its use is cost-effective. This result is encouraging for the introduction of a new AWCS, since the above

scenarios may well represent real-world situations where more waste is produced in certain areas and where the vehicle capacity is inadequate relative to the demand.

2.6 Case study for Vienna, Austria

In the city of Vienna, Austria, residents are required to dispose of certain types of waste, such as plastic and glass, at designated collection points outside their homes, typically on street corners. These collection points closely align with the structure of our model, which focuses on a point-to-point waste collection system rather than a door-to-door service. Given this similarity, Vienna provides a highly relevant environment for evaluating the performance of our model in optimizing waste collection routes and reducing operational inefficiencies.

We chose Vienna as a representative for a case study for the following reasons: Vienna has a diverse urban infrastructure, well-documented data is available, and the city is ongoing efforts to improve transportation, sustainability, and urban planning.

2.6.1 Data

The Viennese open data platform *open government data* (`https://data.wien.gv.at`) allows us to simulate realistic waste collection scenarios and test the applicability of our approach in dealing with large-scale urban waste management challenges. We use the available locations of waste collection points for our experiments (note that suitable demand data is not available in the data platform). Specifically, we consider plastic waste collection points across different districts of Vienna. Distances between collection points (and o^T and o^R, respectively) are computed with tools from the open street map (OSM) project (`https://www.openstreetmap.org/`). To create subsets of instances, we first combine several districts of Vienna, resulting in eight different subsets. Table 2.3 shows which districts are combined and the resulting number of plastic bin locations in each of the eight subsets.

Districts	$\{1,3,4\}$	$\{1,4,6,7,8,9\}$	$\{2,20\}$	$\{4,5,6\}$	$\{15,16\}$	$\{16,17\}$	$\{17,18\}$	$\{18,19\}$
No. loc.	140	174	123	86	148	170	158	212

Table 2.3: Subsets of instances defined by districts of Vienna and number of plastic waste bin locations per subset.

In a second step, we generate the 48 instances of the case study from the eight subsets by varying the following two key assumptions: (a) the locations of the source o^T of the AWCS and the depot o^R of the trucks in the point-to-point collection system and (b) the distribution of demand q_i of the collection points $i \in C$.

The geographical positions of o^T and o^R strongly influence, for each collection point, the attractiveness of the AWCS and point-to-point collection system.

Median: We solve a unit-weight 2-median problem [52] to determine the two best locations for o^T and o^R. The use of unit weights makes the determination of the two locations for o^T and o^R independent from the demand scenario, and it reflects that the capacity of the AWCS limits the number of customers being served by it.

Random: The locations for o^T and o^R coincide with the locations of the first and second customer, which can be interpreted as a random choice.

The second key factor is the amount of waste at each collection point. To address this variability, we create three different demand scenarios:

Uniform demand: The first scenario assumes a uniform demand that is $UD[2,8]$-distributed for all collection points.

HiDem High demand in one district: Also the second scenario starts with the $UD[2, 8]$-distributed demand. Then, one district is selected, and the demand for its collection points is uniformly elevated by a factor of four. For example, in the subset $\{1, 3, 4\}$, which includes 140 collection points, the 24 collection points of district 1 have a demand approximately four times higher than that of the other collection points.

HiDemR High demand randomly distributed: The third scenario is like the second except that collection points with four times higher demand are randomly chosen. To have comparable scenarios, the number of these collection points is the same as in the second scenario. For example, in the subset $\{1, 3, 4\}$, the demand for 24 randomly chosen collection points is approximately four times higher than that of the other 116 collection points.

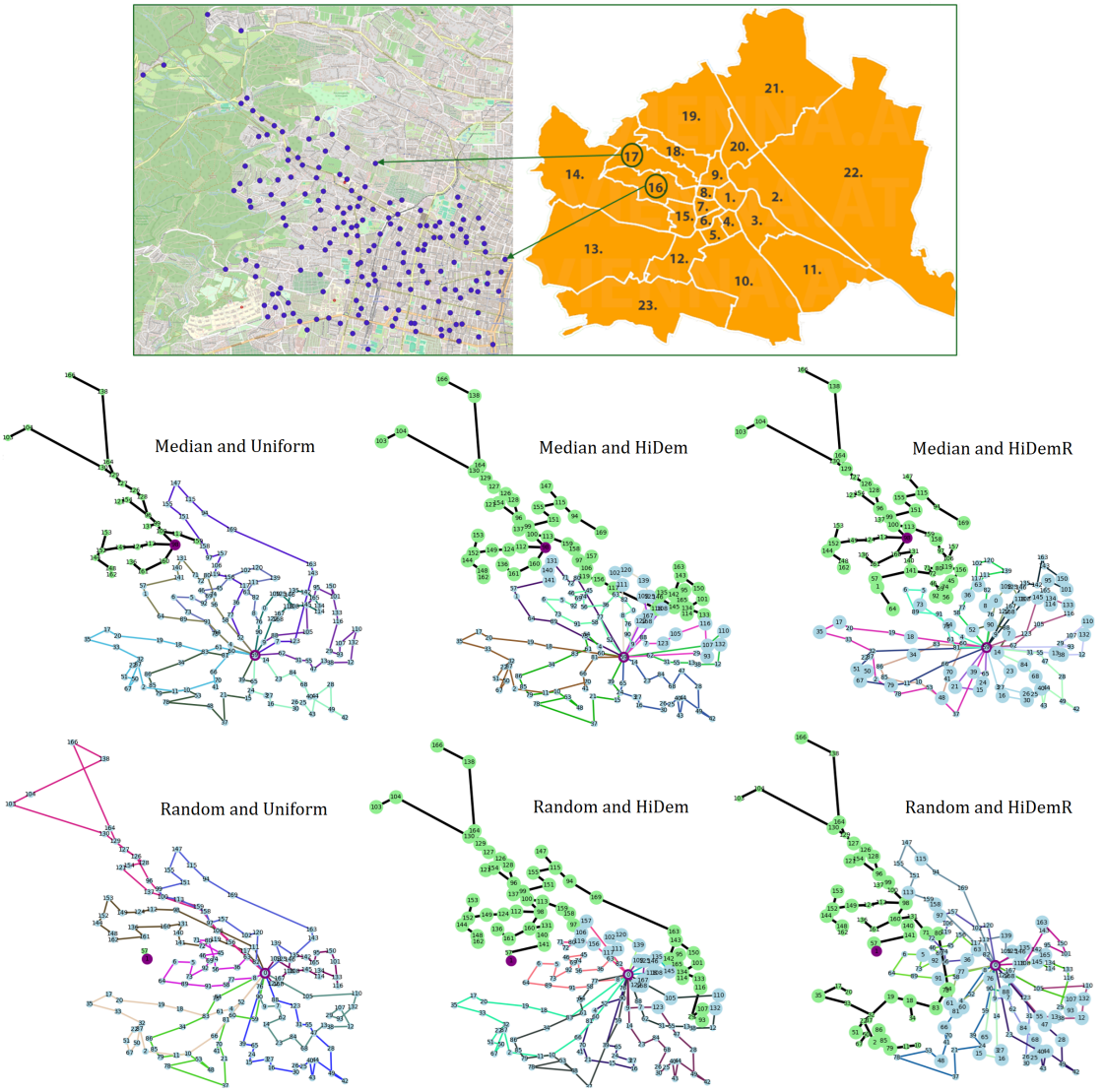


Figure 2.5: Result for the subset $\{16, 17\}$ of the districts. The AWCS source o^T is shown as •, the vehicle depot o^R as ○, customers served by the AWCS as •, and customers served by truck as •. The level of the demand determines the size of the point.

We note that the total demand in the two higher-demand scenarios **HiDem** and **HiDemR** is approximately 40% higher than in the demand scenario **Uniform**.

Finally, for all 48 instances (created from the eight above combinations of districts, two choices for o^T and o^R, and three demand scenarios), we assume that

- all garbage collection vehicles are identical, each with a capacity of $Q^R = 100$;
- the cost ratio is $f = 2$; and
- the AWCS capacity suffices to reach 33% of the customers, i.e., $Q^T = |C|/3$.

2.6.2 Results

Table 2.4a shows aggregated results for the 48 instances and Table 2.4b for the (sixth) subset $\{16, 17\}$ of the districts. The entries *tree %cust.*, *tree %dem.*, and *total cost* have the same meaning as in Table 2.2.

Source/ depot	demand distrib.	tree %cust.	tree %dem.	total cost
Median	**Uniform**	12.7	13.7	82549
	HiDem	32.4	47.06	91918
	HiDemR	31.5	37.1	95957
Random	**Uniform**	11.9	11.7	86576
	HiDem	30.7	54.4	93553
	HiDemR	33.1	44.4	102537

(a) For Vienna as a whole.

Source/ depot	demand distrib.	tree %cust.	tree %dem.	total cost
Median	**Uniform**	17.2	17.7	90249
	HiDem	33.3	56.1	101256
	HiDemR	33.3	37.5	108259
Random	**Uniform**	0.59	0.36	90441
	HiDem	33.3	54.9	102689
	HiDemR	33.3	39.2	110850

(b) For the subset $\{16, 17\}$ of the districts.

Table 2.4: Aggregated results of the Vienna case study.

For Vienna as a whole, the AWCS approaches its full capacity (33% of customers) in the scenarios **HiDem** and **HiDemR**. The number of customers served by the AWCS is consistently higher than in the scenario **Uniform**. In both high-demand scenarios, the percentage of demand covered by the AWCS is substantially higher than the percentage of customers served. Thus, the AWCS serves the customers with the relatively higher demand. Naturally, scenario **Uniform** covers the fewest customers. Furthermore, the smaller total demand in scenario **Uniform** implies the smallest total costs, where optimized source and depot locations (**Median**) obviously lead to smaller costs compared to randomly chosen locations (**Random**). The average cost saving amounts to around 5%.

In addition to the aggregated results for all subsets, we have selected the subset $\{16, 17\}$ of the districts for a more detailed analysis. For this purpose, the upper part of Figure 2.5 shows the geographical locations of the 170 waste collection points of the subset $\{16, 17\}$ and of these two districts in relation to the other 21 municipal districts of Vienna. The lower part visualizes the solutions of all six scenarios (**Median** or **Random**, three demand scenarios). The associated quantitative indicators are summarized in Table 2.4b. We make two key observations:

- **Median** versus **Random**: In the **Uniform** scenario, the total cost of the system for **Median** and **Random** is almost equal. However, they operate differently: In **Median**, the AWCS serves 17.2% of the customers and covers 17.7% of the total demand. In **Random**, the AWCS remains almost unused due to the lack of customers near the source. Connecting customers to o^T is not advantageous for low-demand customers far from the o^T. However, in the two high-demand scenarios **HiDem** and **HiDemR**, the AWCS is fully utilized regardless of the location of the source. Compared to **Random**, **Median** achieves the highest cost savings in demand **HiDemR** scenario (about 2.4%). This underscores the importance of locating the source o^T well, i.e., close to high-demand customers.

- **Median** with either **Uniform** or high-demand (**HiDem**, **HiDemR**): We now focus on different demand scenarios for well-chosen source and depot (**Median**). Significant changes can be observed when moving from the demand scenario **Uniform** to the district-wide high-demand scenario **HiDem**, see Table 2.4b. The demand covered by the AWCS increases by 39 percentage points. This jump translates into an 11% increase in the system's total cost, demonstrating how shifts in demand patterns affect the efficiency of the AWCS. On the other hand, the transition from **Uniform** to **HiDemR** increases the portion of the demand covered by the AWCS and total cost by around 20% and 17%, respectively. This reflects the challenges of introducing an AWCS for areas with randomly distributed demand, which clearly reduces the overall system performance.

The Vienna case study shows that it is very important to identify good potential sites for the installation of the AWCS waste treatment facility (the source o^T), which should be centrally located within a high-demand sub-area. Our matheuristic can be used to evaluate potential networks using the CWCP. Even with a well-suited location for o^T, it is also important to plan with realistic demand scenarios because they have a strong impact on the size (required capacity), structure, and total cost of the AWCS. In addition to these challenges, the study also highlights the potential benefits of an integrated system. We think that the installation of an AWCS is worth considering, particularly in newly developed areas where the city of Vienna can design highly efficient waste collection systems that leverage pneumatic infrastructure powered by renewable energy. The system promises multiple benefits (see discussion in Sections 3.1 and 3.2), including increased operational efficiency, cost savings, enhanced environmental sustainability, and improved public health outcomes.

2.7 Conclusions and outlook

In this paper, we introduced a comprehensive mathematical model for the CWCP, which integrates an AWCS into a traditional point-to-point waste collection system. To solve the CWCP, we applied a BPC-based matheuristic in which the AWCS-related subproblems are solved with a heuristic column generation approach. This holistic solution approach simultaneously determines, for each collection point, whether it is served by the AWCS or by a truck and constructs the network for the AWCS and the routes for the trucks.

We compared the matheuristic with an exact branch-and-price algorithm on small instances and found that it produces near-optimal feasible solutions with the largest deviation of the upper bound below one percent. In the majority of the cases, especially for the larger instances in this group, the matheuristic computed better solutions than the exact branch-and-price given the time limit of one hour per instance. The bottleneck of the exact branch-and-price is the MIP-based exact tree generation solver, which explains why the matheuristic performs so much better. The detailed computational study then used 45 larger CVRP instances [167], ranging in size from 100 to 308 nodes. These were combined with scenarios with different capacity limits for the AWCS, different cost ratios between the two subsystems, and different demand distributions. The matheuristic provides feasible solutions for all resulting CWCP instances, underscoring the applicability of our approach.

Our results indicate that the installation of an AWCS is particularly beneficial when some collection points have a relatively high demand, and the vehicle capacity is small compared to the demand. This result is consistent even for the highest cost ratio of $f = 5$, which represents the most challenging scenario for AWCS deployment. In almost all cases, the use of the AWCS reduces the total cost of the system compared to a traditional point-to-point collection system using trucks only.

Furthermore, the findings of a case study examining the collection of plastic waste in various districts of Vienna were presented. The study demonstrates that the results are highly contingent upon the location of the depot and the AWCS waste treatment facility, as well as the demand

scenario. For example, selecting an appropriate location (computed by solving a median problem) for the AWCS facility results in an average cost saving of approximately five percent in comparison to a random selection. In the majority of high-demand scenarios, the AWCS operates at or near its maximum capacity.

We have identified a number of avenues for further research. One avenue is to extend the fundamental problem setting of the CWCP by incorporating additional constraints that are pertinent in specific real-world contexts: For the point-to-point waste collection system, the incorporation of periodic planning horizons may prove a potentially fruitful avenue of inquiry (see also Section 2.1). For the AWCS, the technical limitations of a pneumatic system may result in tree-generation subproblems that must, e.g., incorporate branch-length constraints for the tree that models the AWCS network. A second avenue for further research would be the consideration of an integrated CWCP that selects an appropriate facility for the AWCS from a set of potential locations. This would fall into a new category of problems in location routing and network design. Addressing these aspects could facilitate the development of more efficient and sustainable waste management solutions for cities worldwide.

Acknowledgement

The authors gratefully acknowledge the computing time granted on the high performance computing cluster MOGON II at Johannes Gutenberg University Mainz (`https://hpc-en.uni-mainz.de/`).

Statements and declarations

Conflicts of interest: The authors have no competing interests to declare that are relevant to the content of this article.

2.8 Appendix

2.8.1 Detailed Results

Table 2.5 contains detailed results for the comparison of the branch-and-price algorithm (BaP) and the matheuristic (MH). For each instance, it is shown whether it was solved to optimality (Opt), the computation time of BaP (in seconds), the computation time of MH (in seconds), the best-known solution value (BKS), and the percentage difference of the upper bounds computed by the BaP and the MH (Δ UB).

Tables 2.6–2.9 show the results for the 45 first instances of [167] from the CVRPLIB and twelve scenarios grouped by tree capacity $Q^T \in \{|C|/5, |C|/3, |C|/2\}$ (denoted as 20%, 30%, and 50%) and cost ratio $f \in \{1.1, 2, 3, 5\}$. For each instance, we present the number of routes (no. of routes), the percentage of customers covered by the tree (tree %cust.), the percentage of demand covered by the tree (tree %dem.), and the best-found solution value (total cost).

Figure 2.6 illustrates the impact of the tree capacity and cost ratio on the total cost and the percentage of customers served by the AWCS. The figure shows that, for all three capacity levels, up to a cost ratio of 3, the tree is almost fully utilized.

Instance	Opt	Time BaP	Time MH	BKS	Δ UB
A-n32-k5	yes	298.8	18.4	710	0.00
A-n33-k5	yes	176.9	15.7	592	0.00
A-n33-k6	yes	301.4	29.7	677	0.00
A-n34-k5	no	3600.0	748.3	703	0.00
A-n36-k5	no	3600.0	3600.0	752	−3.09
A-n37-k5	yes	648.5	160.2	644	0.31
A-n37-k6	yes	2637.5	301.4	811	0.00
A-n38-k5	yes	1098.8	64.1	661	0.00
A-n39-k5	yes	3405.0	569.9	738	0.00
A-n39-k6	yes	136.0	3.6	743	0.00
A-n44-k6	yes	3537.5	411.7	822	0.36
A-n45-k6	yes	1781.9	60.9	855	0.00
A-n45-k7	yes	1972.4	119.2	899	0.00
A-n46-k7	no	3600.0	732.1	797	0.00
A-n48-k7	no	3600.0	3600.0	947	−2.27
A-n53-k7	no	3600.0	3600.0	887	−1.22
A-n54-k7	no	3600.0	1339.2	876	0.00
A-n55-k9	no	3600.0	1401.4	887	−0.78
A-n60-k9	no	3600.0	3600.0	999	−1.58
A-n61-k9	no	3600.0	298.4	807	0.00
A-n62-k8	no	3600.0	3600.0	1024	−0.39
A-n63-k10	no	3600.0	650.3	977	−0.10
A-n63-k9	no	3600.0	1827.4	1230	−0.24
A-n64-k9	no	3600.0	3600.0	1084	−0.09
A-n65-k9	no	3600.0	319.7	954	0.00
A-n69-k9	no	3600.0	3600.0	1010	−0.10
A-n80-k10	no	3600.0	3600.0	1398	0.00

Table 2.5: Detailed results of the exact BaP algorithm (BaP) and the matheuristic (MH).

Instance	tree capacity 20%				tree capacity 33%				tree capacity 50%			
	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost
X-n101-k25	18	20	20	20143	14	33	33	16157	8	50	50	12919
X-n106-k14	11	20	20	20731	9	33	30	17108	7	50	50	13347
X-n110-k13	10	20	20	13061	9	33	30	11879	6	50	50	10690
X-n115-k10	4	20	60	10856	4	33	60	10195	3	50	70	9739
X-n120-k6	5	20	20	11912	5	33	33	11876	3	50	50	9418
X-n125-k30	20	20	20	34242	13	33	33	24635	8	50	50	17486
X-n129-k18	12	20	20	21455	9	33	30	17471	6	50	50	14121
X-n134-k13	11	20	20	10167	9	33	30	9050	6	50	50	7836
X-n139-k10	10	20	20	12678	9	33	30	11813	5	50	50	10629
X-n143-k7	6	20	20	14793	5	33	30	13465	4	50	50	11546
X-n148-k46	34	20	20	28778	26	33	33	22379	18	50	50	16324
X-n153-k22	5	20	80	9905	5	33	80	9047	3	50	80	7925
X-n157-k13	12	20	20	14918	10	33	33	12314	7	50	50	9268
X-n162-k11	10	20	20	13145	8	33	30	12410	6	50	50	11132
X-n167-k10	8	20	20	17862	7	33	30	15706	6	50	50	14004
X-n172-k51	36	20	20	30693	27	33	33	22966	18	50	50	15958
X-n176-k26	7	20	70	20018	6	33	70	17131	5	50	80	15161
X-n181-k23	18	20	20	20575	16	33	30	18012	12	50	50	14425
X-n186-k15	13	20	20	20469	11	33	30	17556	8	50	50	14200
X-n190-k8	7	20	20	16064	5	33	30	12683	4	50	50	10649
X-n195-k51	36	20	20	31275	26	33	33	25292	17	50	50	19277
X-n200-k36	28	20	20	45143	22	33	30	35282	16	50	50	26258
X-n204-k19	15	20	20	16896	13	33	30	15536	10	50	50	13712
X-n209-k16	14	20	20	25492	11	33	30	21870	8	50	50	18030
X-n214-k11	10	20	20	10740	8	33	30	9489	6	50	50	7807
X-n219-k73	58	20	20	86056	49	33	33	67757	37	50	50	47416
X-n223-k34	24	20	20	29324	19	33	30	23543	14	50	50	18580
X-n228-k23	9	20	60	15495	8	33	70	14701	6	50	80	12037
X-n233-k16	12	20	20	16326	10	33	40	16506	7	50	50	14838
X-n237-k14	12	20	20	24805	10	33	33	21731	8	50	50	18324
X-n242-k48	33	20	20	54427	25	33	40	42036	17	50	50	30292
X-n247-k50	18	20	60	16511	8	33	80	12289	9	50	80	13839
X-n251-k28	23	20	20	32476	20	33	30	28221	15	50	50	21645
X-n256-k16	15	20	20	18199	13	33	30	16967	11	50	50	15609
X-n261-k13	11	20	20	24804	9	33	30	21982	7	50	50	18427
X-n266-k58	46	20	20	53231	38	33	33	42001	29	50	50	29661
X-n270-k35	30	20	20	29307	24	33	30	24545	19	50	50	20499
X-n275-k28	23	20	20	17781	19	33	33	14672	15	50	50	12548
X-n280-k17	9	20	60	23220	8	33	60	21796	6	50	70	19135
X-n284-k15	12	20	20	18478	10	33	30	16853	8	50	50	14018
X-n289-k60	44	20	20	57733	34	33	33	42539	23	50	50	31813
X-n294-k50	37	20	20	35452	27	33	30	28459	20	50	50	23845
X-n298-k31	26	20	20	28458	21	33	30	24371	16	50	50	20472
X-n303-k21	18	20	20	20523	15	33	30	18963	11	50	50	16481
X-n308-k13	9	20	50	22898	8	33	50	21968	6	50	60	19557

Table 2.6: Results for cost ratio $f = 1.1$.

Instance	tree capacity 20%				tree capacity 33%				tree capacity 50%			
	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost
X-n101-k25	19	20	20	21426	14	33	33	18298	10	50	50	15564
X-n106-k14	11	20	20	22130	9	33	30	18442	6	50	50	15067
X-n110-k13	10	20	20	14477	9	33	30	14108	6	50	50	14070
X-n115-k10	9	2	10	12555	9	3	10	12781	9	3	10	12513
X-n120-k6	5	20	20	12758	5	20	20	13020	5	26	26	13203
X-n125-k30	20	20	20	34459	15	33	33	25724	9	50	50	19759
X-n129-k18	12	20	20	23565	9	33	30	20495	6	50	50	18097
X-n134-k13	11	19	20	10640	9	33	30	10095	6	50	50	9045
X-n139-k10	11	0	0	13696	11	0	0	13741	11	0	0	13723
X-n143-k7	6	20	20	16522	5	33	30	15283	4	50	50	14832
X-n148-k46	35	20	20	30386	26	33	33	25165	20	50	50	19849
X-n153-k22	5	20	80	11913	5	33	80	11437	4	50	80	10924
X-n157-k13	12	20	20	15334	10	33	33	13029	7	50	50	10682
X-n162-k11	9	20	20	13877	10	20	20	14069	10	22	20	13941
X-n167-k10	8	20	20	19258	7	33	30	18094	6	50	50	17545
X-n172-k51	37	20	20	33745	27	33	33	25646	17	50	50	20113
X-n176-k26	7	20	70	23326	6	33	70	20571	5	50	80	18904
X-n181-k23	18	20	20	21357	15	33	30	18994	12	50	50	16894
X-n186-k15	13	20	20	21842	11	33	30	20291	8	50	50	18218
X-n190-k8	7	20	20	17169	6	33	30	15807	4	50	50	12832
X-n195-k51	37	20	20	33782	28	33	33	28756	19	50	50	24602
X-n200-k36	28	20	20	46028	22	33	30	37018	16	50	50	28407
X-n204-k19	16	20	20	18661	13	33	30	17780	10	50	50	17315
X-n209-k16	14	20	20	27571	12	33	30	25403	9	50	50	22548
X-n214-k11	10	20	20	11365	8	33	30	10425	6	50	40	10222
X-n219-k73	58	20	20	88168	49	33	33	70930	37	50	50	51938
X-n223-k34	26	20	20	32562	20	33	30	27429	14	50	50	23083
X-n228-k23	10	20	60	18074	9	33	60	17943	7	50	70	17152
X-n233-k16	12	20	20	18327	10	33	30	18388	9	50	50	18492
X-n237-k14	13	20	20	28448	10	33	33	25626	8	50	50	23581
X-n242-k48	33	20	20	57362	26	33	40	46884	17	50	50	35111
X-n247-k50	18	20	60	18940	8	33	80	14631	7	50	80	14143
X-n251-k28	23	20	20	34122	20	33	30	30072	15	50	50	25558
X-n256-k16	15	20	20	19292	13	33	30	19200	10	50	50	18747
X-n261-k13	11	20	20	26739	9	33	30	25163	7	50	50	23720
X-n266-k58	47	20	20	56480	39	33	33	45434	29	50	50	34677
X-n270-k35	29	20	20	31390	24	33	30	27613	20	50	50	25756
X-n275-k28	23	20	20	18954	20	33	33	17080	14	50	50	15593
X-n280-k17	10	20	50	27443	8	33	60	26712	7	50	70	26154
X-n284-k15	13	20	20	20584	11	33	30	18333	8	50	50	16711
X-n289-k60	45	20	20	60897	33	33	33	45986	22	50	50	37681
X-n294-k50	37	20	20	38165	28	33	30	35628	21	50	50	30402
X-n298-k31	26	20	20	30320	21	33	30	26629	15	50	50	24025
X-n303-k21	18	20	20	21730	14	33	30	20146	11	50	50	19840
X-n308-k13	9	20	40	24456	9	33	50	24885	7	50	60	25163

Table 2.7: Results for cost ratio $f = 2$.

Instance	tree capacity 20%				tree capacity 33%				tree capacity 50%			
	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost
X-n101-k25	19	20	20	22749	16	33	33	20374	10	50	50	18298
X-n106-k14	11	20	20	23184	9	33	30	19917	7	50	50	17214
X-n110-k13	13	0	0	15111	13	0	0	15165	13	0	0	15187
X-n115-k10	9	2	10	12386	9	2	10	12379	9	2	10	12308
X-n120-k6	6	11	10	14368	6	11	11	14386	6	12	12	14361
X-n125-k30	20	20	20	35554	15	33	33	27134	9	50	50	21985
X-n129-k18	13	20	20	25344	11	33	30	23964	8	50	50	22075
X-n134-k13	11	20	20	11167	9	33	30	10885	6	50	50	10584
X-n139-k10	11	0	0	13807	11	0	0	13706	11	0	0	13705
X-n143-k7	7	0	0	17175	7	0	0	17337	7	4	0	16590
X-n148-k46	35	20	20	32024	28	33	33	27772	20	50	50	23812
X-n153-k22	6	20	70	14042	5	33	80	13305	4	50	80	13497
X-n157-k13	11	20	20	14945	9	33	33	13378	7	50	50	12254
X-n162-k11	10	12	10	14569	10	13	10	14532	10	12	10	14459
X-n167-k10	9	20	20	21360	9	17	20	20634	9	18	20	21042
X-n172-k51	37	20	20	35731	27	33	33	28815	17	50	50	25700
X-n176-k26	8	20	70	27479	7	33	70	24809	5	50	80	23507
X-n181-k23	18	20	20	22330	15	33	30	20609	12	50	50	19446
X-n186-k15	13	20	20	23617	11	33	30	23022	12	27	30	23162
X-n190-k8	7	20	20	17867	6	33	30	16664	4	50	50	14520
X-n195-k51	37	20	20	35974	31	33	33	33119	21	50	50	28850
X-n200-k36	28	20	20	46604	22	33	30	37997	16	50	50	30588
X-n204-k19	18	7	10	20018	13	33	30	19884	17	11	10	19776
X-n209-k16	13	20	20	29832	12	33	30	28961	9	50	50	27688
X-n214-k11	10	20	20	11419	10	20	20	11459	10	20	20	11452
X-n219-k73	58	20	20	90465	49	33	30	74593	37	50	50	56958
X-n223-k34	25	20	20	34535	21	33	30	31551	15	50	50	27584
X-n228-k23	13	20	40	19791	11	33	50	18898	11	50	50	20528
X-n233-k16	15	9	10	19869	16	10	10	19818	16	10	10	19916
X-n237-k14	15	0	0	29688	10	33	33	28666	8	50	50	27897
X-n242-k48	33	20	20	60000	26	33	40	51385	18	50	50	41141
X-n247-k50	18	20	60	20359	9	33	80	17652	8	50	80	16927
X-n251-k28	23	20	20	35449	19	33	30	31958	15	50	50	29297
X-n256-k16	17	6	0	19815	17	8	10	20455	17	8	10	20590
X-n261-k13	14	0	0	29029	10	33	30	30261	14	0	0	29292
X-n266-k58	47	20	20	58810	39	33	33	50114	30	50	50	39870
X-n270-k35	31	20	20	34214	25	33	30	31129	20	46	40	29587
X-n275-k28	24	20	20	20496	20	33	33	18947	15	50	50	18667
X-n280-k17	11	20	40	30343	9	33	50	29851	15	8	20	32193
X-n284-k15	13	20	20	21666	11	33	30	20770	8	50	50	19666
X-n289-k60	45	20	20	63519	34	33	33	50387	22	50	50	43229
X-n294-k50	39	20	20	41634	33	33	30	39811	23	50	50	36751
X-n298-k31	25	20	20	32157	21	33	30	30713	16	50	50	31019
X-n303-k21	19	12	10	22410	19	13	10	22606	19	13	10	22591
X-n308-k13	9	20	40	27010	10	20	40	28630	10	12	30	27753

Table 2.8: Results for cost ratio $f = 3$.

Instance	tree capacity 20%				tree capacity 33%				tree capacity 50%			
	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost	no. of routes	tree %cust.	tree %dem.	total cost
X-n101-k25	22	20	20	24765	16	33	33	23724	11	50	50	23658
X-n106-k14	11	20	20	23841	9	33	30	22238	6	50	50	19857
X-n110-k13	13	0	0	15110	13	0	0	15160	13	0	0	15016
X-n115-k10	9	2	10	12483	9	2	10	12471	9	2	10	12460
X-n120-k6	7	0	0	14301	7	0	0	14942	6	0	0	13725
X-n125-k30	20	20	20	37561	15	33	33	29677	9	50	50	26411
X-n129-k18	13	20	20	28944	11	33	30	28262	8	50	50	29305
X-n134-k13	12	10	10	11478	12	11	10	11539	12	10	10	11466
X-n139-k10	11	0	0	13724	11	0	0	13764	11	0	0	13787
X-n143-k7	8	0	0	17568	7	0	0	17344	7	0	0	17348
X-n148-k46	35	20	20	35114	29	33	33	32445	27	50	50	32546
X-n153-k22	9	20	60	17490	7	33	70	16785	9	20	60	17557
X-n157-k13	11	20	20	15971	9	33	30	14629	8	46	40	14491
X-n162-k11	11	0	0	14575	11	0	0	14421	12	0	0	14510
X-n167-k10	10	2	0	21476	10	2	0	21725	10	2	0	21932
X-n172-k51	38	20	20	39449	27	33	33	34984	21	44	44	33909
X-n176-k26	10	20	60	33513	9	33	60	31594	5	50	80	32906
X-n181-k23	18	20	20	23711	17	25	25	23514	16	30	30	23413
X-n186-k15	17	2	0	25128	15	2	0	24943	15	1	0	25066
X-n190-k8	9	1	0	19401	6	33	30	18988	5	50	40	18749
X-n195-k51	40	20	20	39467	33	33	33	37947	22	50	50	36759
X-n200-k36	28	20	20	48468	23	33	30	41727	17	50	50	35019
X-n204-k19	19	0	0	19983	19	0	0	20228	19	0	0	20037
X-n209-k16	17	0	0	32939	17	1	0	32150	17	0	0	32442
X-n214-k11	12	0	0	11760	12	0	0	11744	13	0	0	12093
X-n219-k73	58	20	20	94630	49	33	30	81678	37	50	50	66790
X-n223-k34	31	13	10	39278	22	33	30	37661	16	50	50	36689
X-n228-k23	15	20	30	22612	16	19	30	22882	16	14	30	23079
X-n233-k16	18	0	0	20498	18	0	0	20374	18	0	0	20414
X-n237-k14	15	0	0	29491	15	0	0	29431	15	0	0	29623
X-n242-k48	35	20	20	65830	28	33	30	58484	19	50	50	52454
X-n247-k50	19	20	60	23851	12	33	70	22008	10	50	80	21492
X-n251-k28	24	20	20	39138	20	33	30	36535	20	33	30	36746
X-n256-k16	18	2	0	20507	17	2	0	19612	18	2	0	20764
X-n261-k13	14	0	0	28870	14	0	0	29242	14	0	0	29077
X-n266-k58	47	20	20	64171	39	33	33	57303	29	50	50	50122
X-n270-k35	37	0	0	36281	37	0	0	36477	37	0	0	36508
X-n275-k28	29	2	2	22433	28	2	2	21758	29	2	2	22312
X-n280-k17	16	6	20	34395	15	5	20	33173	15	7	20	33294
X-n284-k15	16	0	0	22428	16	0	0	22427	16	0	0	22505
X-n289-k60	44	20	20	68374	34	33	33	58075	23	50	50	53452
X-n294-k50	40	20	20	46507	45	12	10	46869	46	10	10	46787
X-n298-k31	33	2	0	35752	32	3	0	35533	33	3	0	35517
X-n303-k21	23	1	0	24002	22	2	0	23339	22	2	0	23268
X-n308-k13	14	1	0	29787	14	1	0	29096	15	1	0	29040

Table 2.9: Results for cost ratio $f = 5$.

2.8.2 Groups of instances and visualization of solutions of different demand scenarios

Table 2.10 shows the assignment of the instances to the five groups **SL**, **U**, **SCap**, **LCap**, and **Q**. Note that the benchmark consists of 45 instances, some of which are not assignable to one or more of the five groups.

Group	Instances
SL 70 to 95% of the customer demands are $UD[1, 10]$, the remaining demands are $UD[50, 100]$.	X-n115-k10 X-n176-k26 X-n228-k23 X-n247-k50 X-n280-k17 X-n308-k13
U Unit demand.	X-n120-k6 X-n157-k13 X-n181-k23 X-n219-k73 X-n237-k14 X-n275-k28
SCap The average demand is approximately 25% of the vehicle capacity.	X-n101-k25 X-n125-k30 X-n148-k46 X-n172-k51 X-n195-k51 X-n266-k58 X-n289-k60
LCap The average demand is approximately 5% of the vehicle's capacity.	X-n120-k6 X-n143-k7 X-n190-k8 X-n214-k11 X-n261-k13 X-n284-k15 X-n308-k13
Q The demand of a customer depends on the quadrant in which the customer is located.	X-n125-k30 X-n134-k13 X-n172-k51 X-n200-k36 X-n233-k16 X-n289-k60
Total All 45 instances	...

Table 2.10: Instances of the CVRPLIB in each group.

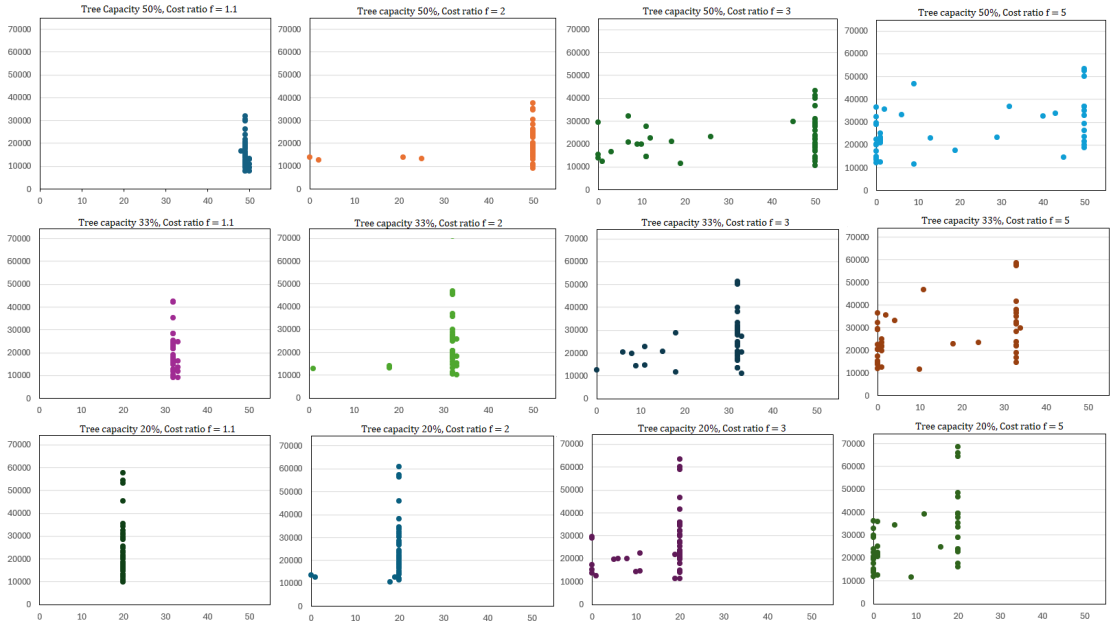


Figure 2.6: Impact of the tree capacity and cost ratio on the total cost and the percentage of customers in the tree. Each data point represents one instance.

Figure 2.7 visualizes the data presented in Tables 2.2 and 2.10.

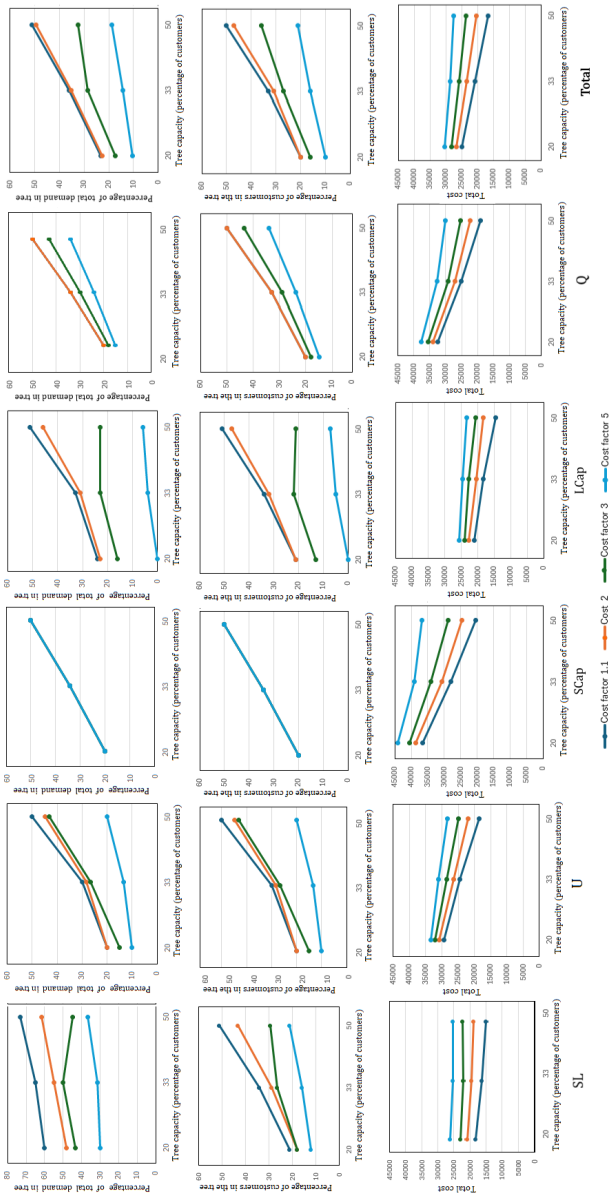


Figure 2.7: Visualization of different demand scenarios.

Chapter 3

An Exact Branch-Price-and-Cut Algorithm for Integrated Waste Collection System: Elementary Shortest Path and Hop Constraint Steiner Tree Problems

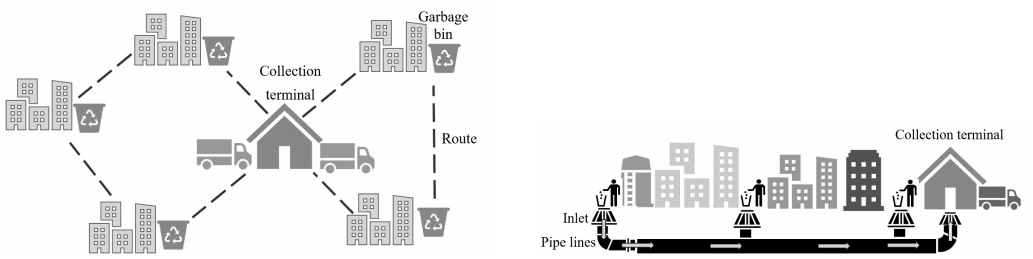
Submitted to: *European Journal of Operational Research*
An Exact Branch-Price-and-Cut Algorithm for Integrated Waste Collection System: Elementary Shortest Path and Hop Constraint Steiner Tree Problems.

Abstract This paper presents an exact solution approach to a waste collection problem, which combines traditional truck-based collection with an automated pneumatic pipe network. The problem jointly determines (i) whether each collection point is assigned to truck service or to the automated system, and (ii) the corresponding truck routes and a cost-minimizing pneumatic tree. The truck component is modeled as a Capacitated Vehicle Routing Problem, while the pneumatic tree is represented by a variant of the Steiner Tree Problem.

Our main contribution is the development and comparison of two decomposition strategies within a branch-price-and-cut framework. In the full decomposition, routing and tree generation are handled independently in two pricing problems. On the contrary, in the partial decomposition, the tree structure is modeled directly in the master problem, and column generation is applied only to the routing part. For the tree-part, we adapt two state-of-the-art formulations, resulting in four distinct branch-price-and-cut algorithms. Computational experiments on benchmark data sets with up to 80 nodes reveal apparent performance differences between the decomposition strategies depending on instance characteristics, demonstrating the substantial impact of modeling choices on solvability, scalability, and overall computational efficiency.

3.1 Introduction

The urban waste collection system is increasingly challenged by the need for efficiency and sustainability. Hybrid systems that combine conventional truck-based waste collection with automated pneumatic networks have emerged as promising solutions for urban waste management. However, designing such systems is computationally challenging, both theoretically, due



(a) Point-to-point waste collection system (b) Automatic waste collection system (AWCS)

Figure 3.1: Principle of a point-to-point waste collection system and an AWCS [53].

to the tight interdependence between routing and network design decisions, and in practice, as demonstrated in real-world studies [53, 131, 162].

The *combined waste collection problem* (CWCP) is defined on an undirected graph where each edge is associated with two transportation costs: one corresponding to a *point-to-point waste collection system*, in which waste is collected from designated points using trucks, and another associated with an *Automated Waste Collection System* (AWCS), in which waste is transported via a pipeline network to a central terminal. The CWCP aims to simultaneously decide whether each collection point should be served by a truck or by the AWCS while also seeking a set of feasible truck routes and at most one feasible tree that together serve all collection points at minimum cost. The overall system is illustrated in Figure 3.1.

A related viewpoint is well established in the literature on location routing problems, where location planning must be made jointly with tour planning [130]. This line of research consistently shows that treating these decisions independently can lead to inferior system designs because of the strong interactions between where service infrastructure is placed and how vehicles are routed. In the context of the CWCP, the same logic applies: the design of the AWCS network and the organization of truck routes mutually influence each other. As highlighted by [149], separating these interdependent decisions often results in solutions that are far from optimal.

Vehicle routing within the point-to-point system is naturally formulated as a *Vehicle Routing Problem* [VRP; 172]. VRPs are particularly suitable for systems where waste is delivered to a set of discrete collection points, whereas *Arc Routing Problem* [ARP; 44] better captures curbside or door-to-door collection in dense urban areas. Waste collection problems at tactical and operational levels are frequently represented using the *Periodic Vehicle Routing Problem* (PVRP) [26], in which each customer may require one or more visits within a given planning horizon. Since our study adopts a strategic perspective in which all service takes place in a single period, we use the classical *Capacitated VRP* (CVRP) to represent truck-based collection. The AWCS component of the CWCP is modeled as a variant of the *Steiner Tree Problem* (STP).

[53] modeled the CWCP for the first time from a strategic perspective and proposed a branch-and-price-based metaheuristic approach with pricing problems for generating vehicle routes and the AWCS tree. In their solution framework, route generation could be solved exactly; however, the AWCS tree pricing problem was computationally too demanding, leading the authors to use only heuristics for generating trees. To overcome this limitation and enhance the practical applicability of the CWCP model, the study at hand extends their work by i) introducing a new practical relevant constraint and ii) solving the resulting problem exactly by *branch-price-and-cut* [BPC, 139] algorithms based on different models and different decomposition strategies. Below, we summarize our main contributions.

Main contributions

- **Modeling enhancement:** Building on the strategic CWCP formulation of [53], we address a key practical shortcoming by introducing a maximum branch-length (hop) constraint into

the AWCS design to capture suction-pressure limits of pneumatic pipeline systems. Hence, the AWCS pricing problem becomes a variant of a *Steiner Tree Problem with Revenues and Hop Constraints* (STPRH), and extends the overall model to the *CWCP with a hop limit*. This modification is well-aligned with real-world AWCS deployments, where pipeline length constraints are essential. For instance, [131] reports that the longest pipeline in Athens is under 1 km, while [162] reports an appropriate range for such systems between 584 m and 1626 m. Similarly, on Roosevelt Island in New York, although the entire AWCS system spans around 3.2 km, the longest individual pipeline branch does not exceed 800 m [103].

- **Methodological framework:** We develop the first exact solution approaches for the CWCP with a hop limit by designing four BPC algorithms derived from two alternative decomposition strategies: (i) a *full decomposition*, in which the routing and tree pricing problems are solved independently within the column generation process, and (ii) a *partial decomposition*, where the STPRH model is embedded directly into the master problem, and column generation is used solely for vehicle routing.
- **STPRH formulations:** The AWCS pricing problem has been identified as the primary computational bottleneck in earlier CWCP work [53]. To deal with this challenge, we implement and analyze two state-of-the-art formulations for the STPRH, originally proposed by [99] and [159]. Unlike the sparse instances typically examined in the STPRH literature, our setting requires solving these models on complete graphs, which considerably increases their computational difficulty. Our experiments provide a comprehensive comparison of the two formulations, under both decomposition strategies, in terms of runtime, scalability, and solution quality.
- **Computational results:** Finally, we perform an extensive computational analysis on benchmark data sets adapted from *Set A* of [15], providing quantitative insights into how the four BPC algorithms differ in efficiency under varying settings. Our analysis compares the performance of all four BPC algorithms, highlighting how decomposition choices influence tractability and solution efficiency.

Paper organization The literature review in Section 3.2 surveys relevant extensions of the STP and reviews existing works comparing different decomposition approaches. Section 3.3 formally defines the CWCP with a hop limit. In Section 3.4, we present the full decomposition approach and the corresponding BPC algorithm. Section 3.5 describes the partial decomposition approach. Computational results are presented and discussed in Section 3.6. We briefly summarize our most important findings in Section 3.7 and point out the challenges ahead for future research.

3.2 Literature Review

The global demand for sustainable urban development has intensified efforts to enhance the efficiency of municipal solid waste collection systems. Numerous studies have examined the comparative performance of different collection strategies, weighing their economic, environmental, and operational implications.

A thorough comparative analysis of AWCSs and conventional point-to-point (truck-based) collection methods is presented by [53], offering a detailed assessment of cost structures, energy consumption, scalability, and life-cycle impacts for both systems. Building on the trade-offs identified between these approaches, the authors introduced the CWCP as a novel framework to better address the complex requirements of urban waste logistics. Overall, the CWCP demonstrated superior performance compared to each individual system across a range of practical scenarios.

Given that the literature on waste collection systems is extensively covered in [53], the remainder of this section focuses on two key areas. First, we review variants of the STP in graphs

that are relevant for our work. Second, we examine works developing and comparing different decomposition approaches for optimization problems .

3.2.1 Related Variants of the Steiner Tree Problem

Many real-world decision problems involve finding a minimum-cost network connecting all or some nodes in a graph. A fundamental example is the STP, which, given a weighted undirected graph $G = (V, E)$ with positive edge costs $c : E \to \mathbf{R}^+$, and a set of terminal nodes $R \subseteq V$ seeks to find a subtree T of G that spans all terminals and minimizes the total cost, i.e., $\sum_{e \in E(T)} c_e$. The STP is NP-hard and serves as the foundation for various practical extensions often modeled via integer linear programs [159]. In the following subsections, we summarize the relevant STP variants presented in the literature and outline their practical relevance, commonly used optimization techniques, and insights from comparative performance studies.

The *Hop-Constrained Steiner Tree Problem* (HSTP) is one of the well-studied variant of the STP extending the problem by a root node r and a so-called hop limit $H \in \mathbf{N}^+$. The task is then to find a minimum-cost subtree T of the graph that spans all terminals such that the number of edges in the path from r to any node $v \in V(T)$ does not exceed H. The problem was first formalized by [82], who introduced hop constraints into multicommodity flow formulations for minimum spanning and Steiner trees as a way to model Quality-of-Service constraints in communication networks.

There are several variants of the HSTP. A well-known special case occurs when $R = V$, which is referred to as the *Hop-constrained Minimum Spanning Tree* (HMST). While the *Minimum Spanning Tree* (MST) can be solved in polynomial time, the HMSTP is NP-hard [98]. The closest problem in the literature to our AWCS pricing problem is the *Steiner Tree Problem with Revenues, Budgets, and Hop Constraints* (STPRBH). This variant generalizes the HSTP by introducing node revenues and an overall budget. The objective is to identify a subtree within a given graph that includes the root node, maximizes the total revenue of the selected nodes, and satisfies two key constraints: the total cost of the edges must remain within the specified budget, and the number of edges between any node and the root must not exceed the hop limit.

The STPRBH originated in telecommunications, where service providers must design cost-efficient networks that meet strict Quality-of-Service requirements. In such networks, the reliability of each link and the maximum number of hops (or links) between a central server and clients are crucial factors. Moreover, clients often contribute different revenue amounts, introducing the need to balance cost minimization with profit maximization. The STPRBH addresses these challenges by identifying a subtree that not only connects key nodes within a given hop limit but also optimizes overall revenue while staying within budget constraints. The STPRH is an NP-hard problem of significant interest in network design, where achieving efficient, cost-effective connectivity under strict performance constraints is critical [119].

The STPRBH was first introduced by [47], who proposed three branch-and-cut models based on Miller–Tucker–Zemlin constraints, one on Dantzig–Fulkerson–Johnson (also known as subtour-elimination) constraints, and one on hop-indexed formulation. These models were effective in solving moderate-size instances (up to 500 nodes and 625 edges), but failed to handle larger instances within practical time limits. To address this, [48] developed a greedy heuristic and a tabu search, which improved solution quality on harder instances with up to 12,500 edges. [112] later explored two exact formulations for the STPRBH based on Miller–Tucker–Zemlin (MTZ) constraints, which they adapted to enforce hop and connectivity restrictions within the tree. Their computational experiments included instances with up to 500 nodes and 625 edges, using hop limits of 3, 5, 6, 9, and 15. [158] introduced branch-and-price algorithms based on directed and undirected path formulations for instances with up to 500 nodes and 625 edges. Additionally, two heuristic approaches for the STPRBH were proposed by Fu and Hao [72, 73]: the Breakout Local Search algorithm [72] and a Dynamic Programming-driven Memetic Search algorithm [73]. They evaluated their methods on large-scale DIMACS benchmark data sets with up to 12,500

edges. The computational results demonstrated that their heuristics produced better feasible solutions compared to those obtained by the method of [47].

The majority of existing STPRBH formulations fall into two categories: *node-oriented* and *edge-oriented* models. The key distinction lies in how hop constraints are imposed. Node-oriented formulations assign each node to a layer (i.e., a depth relative to the root), enabling hop limits to be modeled directly through node depth variables. In contrast, edge-oriented formulations encode hop limits through flow-based constraints on edges [98]. The advantage of node- and edge-oriented formulations lies in their polynomial size, making them directly compatible with standard *Integer linear programming* (ILP) solvers. While edge-oriented formulations tend to have stronger *Linear Programming* (LP) relaxations compared to node-oriented ones, they result in large ILPs, which are only suitable for small graphs. In contrast, node-oriented models use significantly fewer variables, enabling them to handle larger instances. Two node-oriented formulations dominate the current state of the art: the assignment-based model by [159] and the partial-ordering-based model by [99].

[159] developed a branch-and-cut algorithm on a new *layer-assignment-based* formulation of the STPRBH. They introduced a layered graph in which binary variables y_{vh} represent whether node v is at depth h. Their approach significantly outperformed previous exact approaches, successfully solving to optimality 78 of the 86 DIMACS benchmark data sets that had remained unsolved, including those with up to 500 nodes, 12,500 edges, and hop limits from 3 to 25. Their method, which won the STPRBH category in the DIMACS challenge, is widely recognized as a breakthrough in solving large STPRBH instances.

Building upon this foundation, [99] developed a *partial-ordering-based* ILP that offers an alternative yet highly effective way to encode hop constraints. Rather than assigning nodes to specific depths directly, their model uses binary variables to express whether a node's position in the tree is greater or less than a given depth. This leads to a stronger LP relaxation than the layer-assignment model. Remarkably, their method solves all 414 DIMACS benchmark instances to optimality within two hours, including the largest ones that prior approaches had left unsolved. They also introduced a novel preprocessing technique that significantly reduces computation time, especially for large graphs and large hop limits.

A comparative study by [98] indicates that although both models are highly effective, the partial-ordering model generally outperforms the layer-assignment model, both in theory and in practice, solving more instances more quickly. It is worth noting that although subtour elimination constraints are valid inequalities cutting of fractional solutions, only [159] incorporated them through dynamic separation. Neither the work of [99] nor the comparative study [98] included them.

3.2.2 Decomposition Approaches

This section reviews key contributions that compare several different decomposition approaches. [88] compare two Dantzig–Wolfe reformulations for the VRP with time windows and multiple trips, each representing a different decomposition of the routing structure. The *route-based formulation* (SSP) generates full multi-trip routes as columns and yields tighter LP relaxations, but results in highly complex pricing problems. The *trip-based formulation* (SMP), in contrast, decomposes routes into individual trips, which simplifies pricing but requires the master problem to enforce mutual temporal-exclusion constraints. Computational experiments on modified Solomon instances with 25 customers show that, despite providing a weaker relaxation, the trip-based decomposition consistently outperforms the route-based one.

[25] analyzed two decomposition models for a staff scheduling problem: a staff-decomposed formulation and an activity-decomposed one. The staff-based model generates columns representing full schedules for individual staff members, while the activity-based model bundles activity assignments across personnel. Both formulations are embedded in a branch-and-price framework.

Their results show that while the staff-based model offers stronger LP bounds, it also leads to more complex pricing problems and a larger column pool.

[170] focused on *recoverable robust optimization problems*, where an initial solution is chosen under uncertainty and later adjusted in a limited way to balance robustness and flexibility across scenarios. They introduced two primary decomposition strategies: *separate recovery decomposition* (SRD) and *combined recovery decomposition* (CRD). In SRD, first-stage (baseline) and second-stage (recovery) decisions are modeled separately, allowing for parallelizable pricing problems. In contrast, CRD integrates these decisions into single columns, yielding a tighter LP relaxation but increasing the complexity of pricing. Their computational experiments demonstrate that SRD often performs better in practice due to its simplicity, despite the theoretically stronger bounds offered by CRD. Building on this foundation, [165] extended SRD and CRD decomposition strategies to a *two-stage stochastic multiple knapsack problem* (2SMKP). Here, items are selected in the first stage and may be removed in the second stage under different stochastic scenarios. The study mirrors the decomposition structures of [170] but adapts them to a stochastic framework involving multiple knapsacks. Notably, the authors explore several column generation strategies, such as adaptive column addition and pricing frequency control. Their results suggest that SRD, when combined with intelligent column management, often outperforms CRD, particularly in large-scale instances.

[173] examined the integrated shift and task scheduling problem for logistics assistants in hospitals, comparing two decomposition paradigms. In the *two-subproblem* (2SP) approach, task scheduling and shift scheduling are handled in separate pricing problems. Alternatively, the *one-subproblem* (1SP) approach embeds task scheduling within the master problem and uses a single pricing problem for shift generation. While 1SP simplifies the overall structure, 2SP results in a more modular and computationally efficient framework, as it yields stronger relaxations and allows more targeted pricing. Both formulations are solved via column generation, embedded in a branch-and-price framework to handle integrality.

[163] proposed two Dantzig–Wolfe reformulations for the *optimal communication spanning tree problem* (OCSTP): a *path-based* formulation and a *tree-based* formulation. While the tree-based reformulation yields tighter linear relaxation bounds, its pricing problems, formulated as fixed-cost network flow problems, are computationally more intensive compared to the simpler shortest-path problems in the path-based formulation. Consequently, the path-based reformulation, despite providing weaker bounds, leads to superior overall performance. The authors implement combined column-and-row generation techniques and develop efficient BPC algorithms, solving benchmark instances with up to 50 nodes.

These works highlight a common trade-off in decomposition: On the one hand, more decomposition leads to stronger master formulations but increases pricing complexity, on the other hand more decompositions lead to weaker master formulations but usually easier to solve pricing problems. In the majority of previous works the latter approach was more successful leading to better scalable algorithms.

3.3 Problem Description

The CWCP with a hop limit can be seen as a two-level decision problem combining two complementary subsystems: a point-to-point collection system and an AWCS. In this regard, we aim to determine how these two subsystems should be coordinated. In the first level, each collection point is assigned either to the AWCS or to the point-to-point collection system, according to the system constraints. In the second level, a CVRP is solved for the collection points allocated to the trucks, while a minimum-cost tree rooted at a designated depot is constructed for those assigned to the AWCS. Clearly, the first-level and second-level decisions are interdependent, making the CWCP with a hop limit at least as difficult as the CVRP and the STPRH.

A schematic illustration is provided in Figure 3.2, showing a partition of the collection points into the two subsets (green and red points) as well as three routes and a tree covering these two subsets. In addition, it represents an AWCS with a tree hop limit of four.

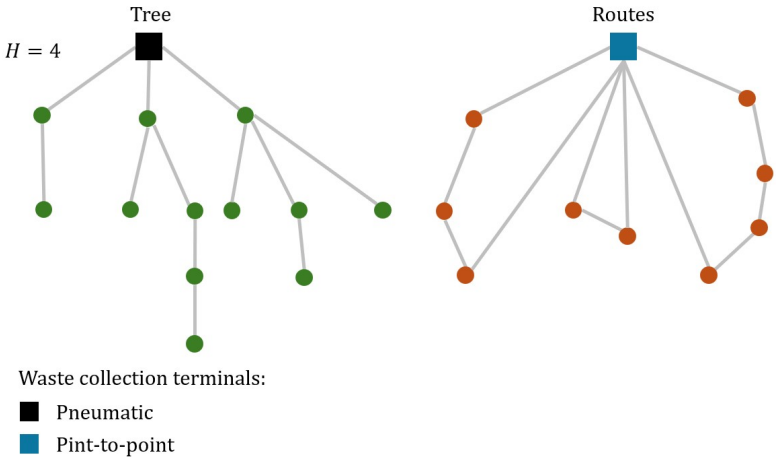


Figure 3.2: Schematic of the CWCP with a hop limit.

Next we formalize the problem. Let C be the set of collection points also called *customers* in the following. For each customer $i \in C$, q_i denotes the quantity of waste that must be collected if served by truck. Note that parameter q_i has no influence on the AWCS as we assume that waste is generated and dealt with continuously over time. A number K of homogeneous trucks, each with a capacity of Q^R, is stationed at the *depot* o^R for point-to-point collection. The AWCS is constrained by a hop limit $H \in \mathbf{N}^+$ and an overall capacity limit Q^T, i.e, only Q^T customers can be served by the AWCS and the number of edges in each branch of the AWCS is limited by H. The *root* of the AWCS is the collection terminal o^T.

The underlying optimization model is defined on an undirected complete graph (V, E) with node set $V = \{o^T, o^R\} \cup C$ and edge set E. Each edge $\{i, j\} \in E$ is associated with two cost values: the routing cost c_{ij} for the trucks and the cost d_{ij} for installing and using the pneumatic pipeline on the edge. Both cost, (c_{ij}) and (d_{ij}), satisfy the triangle inequality. The objective is to construct feasible routes and at most one feasible AWCS such that all customers are covered and costs are minimized.

A feasible AWCS is a cardinality-constrained tree $t = (V', E') \subset (V \setminus \{o^R\}, E)$ rooted at o^T, such that at most Q^T customers belong to V', and every root-to-customer path has at most H edges. Note that the cost d_t of a feasible tree t covering a given set of nodes $V' = \{o^T, i_1, \ldots, i_m\}$ can be obtained by constructing a MST over $\{o^T, i_1, \ldots, i_m\}$ since the graph is complete and the triangle inequality holds for (d_{ij}). A feasible truck route is an ordered sequence $r = (r_0, r_1, \ldots, r_m, r_{m+1})$ starting and ending at the depot o^R, where $r_1, \ldots, r_m$ are distinct customers such that their total waste respects the vehicle capacity, i.e., $\sum_{j=1}^{m} q_{r_j} \leq Q^R$. The cost of a route is $c_r = \sum_{j=0}^{m} c_{r_j, r_{j+1}}$.

3.4 Full Decomposition-Based Column Generation for Tree Generation

In this section, we present a set-partitioning formulation for the CWCP with a hop limit which forms the basis of our BPC algorithm. Similar to the formulation in [53], we employ a *full decomposition* in which tree and route construction is moved to the pricing problem. Let T denote the set of all feasible AWCS trees and R the set of all feasible routes. The model features

two types of variables, i.e., binary variables Υ_t for each feasible tree $t \in T$ and binary variables λ_r for each feasible route $r \in R$. Moreover, parameters a_{ir} and b_{it} indicate whether customer $i \in C$ is served by route $r \in R$ or by tree $t \in T$, respectively.

$$z_{CWCP} = \min \sum_{t \in T} d_t \Upsilon_t + \sum_{r \in R} c_r \lambda_r \tag{3.1a}$$

$$\text{subject to } \sum_{t \in T} b_{it} \Upsilon_t + \sum_{r \in R} a_{ir} \lambda_r = 1 \qquad \forall i \in C \tag{3.1b}$$

$$\sum_{t \in T} \Upsilon_t \leq 1 \tag{3.1c}$$

$$\sum_{r \in R} \lambda_r \leq K \tag{3.1d}$$

$$\Upsilon_t \in \{0,1\} \qquad \forall t \in T \tag{3.1e}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in R \tag{3.1f}$$

The objective (3.1a) minimizes the total cost. Constraints (3.1b) ensure that each customer is served, either by the tree or one of the routes. Constraint (3.1c) enforces that at most one tree is chosen. The number of routes is restricted by Constraint (3.1d). Variable domains are given in (3.1e) and (3.1f).

Due to the exponential number of feasible routes and tree variables, we apply column generation. Column generation considers smaller linear relaxations, known as *restricted master programs* (RMPs), in which the sets of feasible routes and trees (R and T) are replaced by subsets $\bar{R} \subset R$ and $\bar{T} \subset T$. An RMP is solved, variables (columns) with negative reduced cost are identified by solving pricing problems and added to the RMP. This process is repeated until no variables with negative reduced cost exist, which is proven by solving the pricing problems to optimality. Under the full decomposition, two independent pricing problems appear, one generates improving vehicle routes, and the other generates improving AWCS trees. For a given RMP, let $(\pi_i)_{i \in C}$ be the dual prices of the partitioning constraints (3.1b), μ_T be the dual price of the convexity constraint (3.1c), and μ_R be the dual price of the fleet-size constraint (3.1d).

The first pricing problem focuses on generating negative reduced-cost vehicle routes by solving a *shortest-path problem with resource constraints* (SPPRC), while the second targets the construction of negative reduced-cost trees through a cardinality-constrained STPRH. The overall framework is embedded in a BPC scheme, a branch-and-cut approach in which column generation is used at each node of the search tree to solve the linear relaxation of formulation (3.1).

Section 3.4.1 briefly introduces the route-generation procedure as it is quite standard. In Section 3.4.2, we present tree-generation approaches, including two alternative formulations for STPRH with valid inequalities and branching rules, as well as heuristic methods designed to accelerate the overall algorithm. Finally, Section 3.4.3 describes the branching strategies and valid inequalities for the master problem.

3.4.1 Pricing Problem for Route Generation

The objective of the truck-route pricing problem is to identify feasible routes $r \in R$ with negative reduced cost $\bar{c}_r = c_r - \sum_{i \in C} a_{ir}\, \pi_i - \mu_R$. As in most column-generation algorithms for VRPs, the pricing task can be formulated as a *shortest path problem with resource constraints* (SPPRC) [96] that can be solved with a labeling algorithm. Here we briefly summarize the method as the topic has been widely studied in the literature.

Labeling algorithms expand partial paths step by step, starting from an origin to a destination node through the network. Each label represents a partial path and stores the resources accumulated up to the current endpoint. *Resource extension functions* [REFs, 95] are employed to propagate labels over the arcs of the network. Additional dominance criteria are used to eliminate labels that cannot lead to an optimal solution.

For the CWCP with a hop limit, a label $\mathcal{L}$ associated with a partial path $p = (o^R, \ldots, i)$ contains three resources: the reduced cost accumulated along path p (T_i^{rdc}); the total amount of waste collected on path p (T_i^{load}); and a set $T_i^{vis,k}$, which collects the customers $k \in C$ visited along path p.

The initial label $\mathcal{L}_{o^R}$ is defined by the resource values $(0, 0, \emptyset)$. Extending a label $\mathcal{L}_i$ at node i over an arbitrary edge $\{i, j\}$ towards a node j results in a new label $\mathcal{L}_j$ at j. The extension is only conducted if the resulting label $\mathcal{L}_j$ is feasible regarding vehicle capacity and elementary constraints, i.e., $T_i^{load} + q_j \leq Q^R$ and $j \notin T_i^{vis,k}$. For the sake of simplicity, we define $\pi_{o^R} = \mu_R$ and $q_{o^R} = 0$. The new resource values corresponding to the partial path represented by $\mathcal{L}_j$ are then computed using the following REFs:

$$\begin{aligned} T_j^{rdc} &= T_i^{rdc} + c_{ij} - (\pi_i + \pi_j)/2 \\ T_j^{load} &= T_i^{load} + q_j \\ T_j^{vis,k} &= T_i^{vis,k} \cup \{j\} \end{aligned}$$

Since all resources are non-decreasing and are bounded only from above, the standard dominance rule can be applied: A label $\mathcal{L}_{i,1} = (T_{i,1}^{rdc}, T_{i,1}^{load}, T_{i,1}^{vis,k})$ is dominated and therefore discarded if there exists a label $\mathcal{L}_{i,2} = (T_{i,2}^{rdc}, T_{i,2}^{load}, T_{i,2}^{vis,k})$ which ends at the same node i and has not larger resource consumption, i.e., $T_{i,2}^{rdc} \leq T_{i,1}^{rdc}$, $T_{i,2}^{load} \leq T_{i,1}^{load}$, and $T_{i,2}^{vis,k} \subseteq T_{i,1}^{vis,k}$.

To further speed up the pricing procedure, we incorporate the *ng*-path relaxation [18] and adopt a partial pricing strategy employing reduced networks. Moreover, bidirectional labeling is now widely regarded as a standard technique for solving SPPRCs [145, 164]. Due to the symmetric network in the CWCP with a hop limit, we can use an implicit bidirectional labeling scheme of [29] in which forward labels can be directly interpreted as backward labels. In our implementation, the resource T^{load} is chosen as the monotone resource. Consequently, a label is extended only if it satisfies the half-way condition $T^{load} \leq Q^R/2$. At all nodes $i \in C \cup \{o^R\}$, pairs of forward labels $\mathcal{L}_{i,1} = (T_{i,1}^{rdc}, T_{i,1}^{load}, T_{i,1}^{vis,k})$ and $\mathcal{L}_{i,2} = (T_{i,2}^{rdc}, T_{i,2}^{load}, T_{i,2}^{vis,k})$ are merged to feasible routes with negative reduce costs if (i) $T_{i,1}^{load} + T_{i,2}^{load} - q_i \leq Q^R$, (ii) $T_{i,1}^{vis,k} \cap T_{i,2}^{vis,k} \subseteq \{i\}$, and (iii) $T_{i,1}^{rdc} + T_{i,2}^{rdc} < 0$.

3.4.2 Pricing Problem for Tree Generation

The objective of the tree-generation pricing problem is to identify feasible trees $t \in T$ with negative reduced cost $\bar{d}_t = d_t - \sum_{i \in C} b_{it}\pi_i - \mu_T$. The pricing problem can be modelled as a cardinality-constrained STPRH. To solve the STPRH exactly, we build on two formulations inspired by the STPRBH literature: The layer-assignment-based formulation [159] and the partial-ordering-based formulation [99]. Both formulations rely on the core concepts of node depth and layered representations, using node variables to encode feasible positions within the layered graph and thereby achieving strong LP relaxations. Rather than modelling the problem directly on G, a layered graph is constructed in which, for each layer $1 \leq h \leq H$, a duplicate of the nodes of G is created. Connections are then added between nodes in consecutive layers whenever the corresponding nodes are adjacent in G. In a tree rooted at o^T, the depth of a node corresponds to its distance, measured in the number of edges, from the root node, while the layer represents the collection of nodes at a given depth. Imposing a hop limit H therefore restricts the tree to at most H layers, ensuring that every selected customer remains within a feasible distance of the root node. The layer-assignment model [159] explicitly assigns each node to a unique depth layer, whereas the partial-ordering model [99] enforces relative depth relations without fixing exact layers. In both formulations, a directed tree is constructed and, hence, the underlying graph (V, E) is transformed into a directed graph $G = (V, A)$ by replacing each undirected edge $\{i, j\} \in E$ by two anti-parallel arcs (i, j) and $(j, i) \in A$, both assigned the same cost.

Layer-Assignment Formulation

The layer-assignment formulation employs three types of decision variables to describe a directed tree: (i) binary variables u_i for all $i \in C$ indicating whether node $i \in C$ is included in the tree; (ii) binary variables x_{ij} specifying whether arc $(i,j) \in A$ belongs to the tree; and (iii) binary variables y_{ih} denoting whether node $i \in C$ is assigned to layer $h \in \{1, 2, \ldots, H\}$.

$$z_{Ful}^{LA} = \min \sum_{(i,j)\in A} d_{ij}x_{ij} - \sum_{i\in C} \pi_i u_i - \mu_T \tag{3.2a}$$

$$\text{subject to} \sum_{(i,j)\in A} x_{ij} = u_j \qquad \forall j \in C \tag{3.2b}$$

$$\sum_{h=1}^{H} y_{ih} = u_i \qquad \forall i \in C \tag{3.2c}$$

$$x_{o^T i} = y_{i1} \qquad \forall (o^T, i) \in A \tag{3.2d}$$

$$y_{iH} + y_{ih-1} + x_{ij} \leq u_i + y_{jh} \qquad \forall (i,j) \in A,\ \forall\ 2 \leq h \leq H,\ i \neq o^T \tag{3.2e}$$

$$x_{ij} + x_{ji} \leq u_j \qquad \forall i, j \in C \tag{3.2f}$$

$$\sum_{(i,j)\in A} x_{ij} \leq Q^T \tag{3.2g}$$

$$\sum_{(o^T,i)\in A} x_{o^T i} \geq 1 \tag{3.2h}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{3.2i}$$

$$y_{ih} \in \{0,1\} \qquad \forall i \in C,\ 1 \leq h \leq H \tag{3.2j}$$

$$u_i \in \{0,1\} \qquad \forall i \in C \tag{3.2k}$$

The objective (3.2a) minimizes the reduced cost of the constructed tree. Constraints (3.2b) enforce that each included customer node is entered by exactly one arc, preserving the tree structure. Constraints (3.2c) ensure that if a customer node i is selected, it must be assigned to exactly one layer h. Constraints (3.2d) link the root node o^T to each selected customer node on the first layer. The hop-link constraints (3.2e) ensure that if a node i lies on layer $h-1$ ($2 \leq h \leq H$) and arc (i,j) is included in the solution, then node j must lie on layer h. Constraints (3.2f) impose subtour elimination for sets of size two. Constraint (3.2g) ensures that the tree's capacity constraint is respected. Constraint (3.2h) requires the root node o^T to have at least one outgoing arc. Finally, constraints (3.2i)–(3.2k) are integrality constraints that ensure binary decisions for arc selection, layer assignments, and customer inclusion.

Partial-Ordering Formulation

In the partial-ordering formulation, the tree is represented in a manner conceptually similar to the layer-assignment model (3.2). Both formulations use the same decision variables u_i and x_{ij}, but differ in how the layer variables y_{ih} are defined. In the partial-ordering-based formulation, binary variables y_{ih} indicate if node $i \in C \cup \{o^T\}$ lies above layer $h \in \{0, 1, \ldots, H\}$ (i.e., its depth is greater than h).

$$z_{Ful}^{PO} = \min \sum_{(i,j)\in A} d_{ij}x_{ij} - \sum_{i\in C} \pi_i u_i - \mu_T \tag{3.3a}$$

$$\text{subject to} \sum_{(i,j)\in A} x_{ij} = u_j \qquad \forall j \in C \tag{3.3b}$$

$$y_{o^T 0} = 0 \tag{3.3c}$$
$$y_{i0} = 1 \qquad \forall i \in C \tag{3.3d}$$
$$y_{iH} = 0 \qquad \forall i \in C \tag{3.3e}$$
$$y_{ih} - y_{ih+1} \geq 0 \qquad \forall i \in C \cup o^T,\ \forall\ 0 \leq h \leq H-1 \tag{3.3f}$$
$$y_{j0} - x_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{3.3g}$$
$$1 - y_{ih-1} + y_{jh} \geq x_{ij} \qquad \forall (i,j) \in A,\ \forall\ 1 \leq h \leq H \tag{3.3h}$$
$$x_{jk} \leq \sum_{(i,j) \in A : i \neq k} x_{ij} \qquad \forall\ (j,k) \in A : j \in C \tag{3.3i}$$
$$\sum_{(i,j) \in A\ \ i \neq o^T} x_{ij} \geq 1 - y_{iH-1} \qquad \forall\ i \in C \tag{3.3j}$$
$$\sum_{(i,j) \in A} x_{ij} \leq Q^T \tag{3.3k}$$
$$\sum_{(o^T,i) \in A} x_{o^T i} \geq 1 \tag{3.3l}$$
$$x_{ij} \in \{0,1\} \qquad \forall\ (i,j) \in A \tag{3.3m}$$
$$y_{ih} \in \{0,1\} \qquad \forall\ i \in C \cup \{o^T\}, \forall\ 0 \leq h \leq H \tag{3.3n}$$
$$u_i \in \{0,1\} \qquad \forall i \in C \tag{3.3o}$$

The objective (3.3a) minimizes the reduced cost of the constructed tree. Constraints (3.3b) ensure that every selected customer node is entered by exactly one arc, thereby maintaining the tree structure. Constraints (3.3c)—(3.3f) manage the node positions within the tree: Constraint (3.3c) fixes the root node o^T at position 0, Constraints (3.3d) ensures all customer nodes appear in positions after the root node o^T, Constraints (3.3e) restrict the tree layer's depth to a maximum of H, and Constraints (3.3f) enforce the consistency of position indicators. Constraints (3.3g) and (3.3h) enforce structural consistency within the solution. Specifically, they prevent the inclusion of arcs connecting to nodes that are not yet part of the tree, and they ensure positional coherence between connected nodes across consecutive layers. Constraints (3.3i) avoid subtours of size two and enforces connectivity. A customer node that is a leaf with no outgoing arcs is positioned in the last layer with the help of Constraints (3.3j). The tree's capacity constraint is enforced by Constraint (3.3k). Constraint (3.3l) requires the root node o^T to connect to at least one other node. Finally, the variable domains are defined in constraints (3.3m)–(3.3o) for arc selection, layer assignments, and customer selection.

Valid Inequalities and Branching for the STPRH

To strengthen the linear relaxation of the STPRH and speed-up the exact solution of the tree-generation pricing problem, we integrate families of valid inequalities known from the literature as well as tailored branching strategies. These enhancements are essential for the STPRH, since the structural requirements of acyclicity, connectivity, and hop feasibility are otherwise only weakly captured in the linear relaxation.

Depending on whether the STPRH is formulated using the layer-assignment or the partial-ordering approach, we consider three classes of valid inequalities. Whenever the solution of a branch-and-bound node is fractional and violates one of these constraints, we separate a cut in the pricing problem until depth 10 of the branch-and-bound tree.

Subtour Elimination Constraints: Although integer solutions of both models are cycle-free, fractional solutions might have cycles. The classical subtour elimination constraints prevent cycles among selected nodes by ensuring that, for every subset $S \subseteq C$ with $|S| \geq 3$, the number

of arcs inside the subset is bounded.

$$\sum_{\substack{(i,j)\in A\\ i,j\in S}} x_{ij} \leq |S| - 1 \tag{SEC}$$

Lifted Subtour Elimination Constraints: These constraints form a strengthened variant of the classical SEC for problems where the visit to all nodes is not mandatory. For each subset $S \subseteq C$ with $|S| \geq 3$, and for each node $v \in S$, they enforce that the amount of flow entering S is at least as large at the value u_v.

$$\sum_{(i,j)\in A: i\notin S, j\in S} x_{ij} \geq u_v \tag{SEC$_\text{u}$}$$

Node-Arc-Cut Constraints: These constraints, introduced by [159], are only valid for the layer-assignment model (3.2). They enforce that hop assignments are propagated consistently along arcs, ensuring that layer variables reflect valid parent–child relationships under the hop limit. Let $\mathcal{R}$ be the family of binary functions $\mathcal{R} = \{0,1\}^{|A|}$, then for each $q \in \mathcal{R}$, $j \in V$, and $2 \leq h \leq H$, the inequalities

$$\sum_{\substack{(i,j)\in A\\ i\neq o^T}} (q_{ij} \cdot x_{ij} + (1 - q_{ij}) \cdot y_{ih-1}) \geq y_{jh} \tag{NACut}$$

These inequalities guarantee that if node j is assigned to layer h ($y_{jh} = 1$), then at least one incoming arc originates from a node assigned to the previous layer $h-1$ and is included in the tree. Thus, they preserve connectivity and enforce consistency across layers within the layer-assignment model (3.2). When separating these cuts, we choose q_{ij} such that the left-hand side is as small as possible with respect to the current values of the variables x_{ij} and y_{ih-1}.

Branching Priority: We use CPLEX-priorities for branching according to the importance of the decision variables. Node-selection variables u_i receive the highest priority, since they determine customer inclusion. Layer-assignment variables y_{ih} follow, with priority increasing for deeper layers due to their stronger influence on the hop limit h. Arc variables x_{ij} are assigned the lowest priority, reflecting their subordinate role in tree structure once node and layer decisions are fixed.

Greedy Heuristics and Local Search

Before solving the tree-pricing problem exactly, we first attempt to identify negative reduced-cost columns using fast heuristics. We use a modified version of Prim's algorithm to heuristically solve the tree-generation pricing problem. We start with the singleton set $S = \{o^T\}$ and iteratively expands S by adding a new customer $j \in C$. The next node j is chosen by selecting an arc $\{i, j\} \in A$ with $i \in S$ and $j \notin S$ that respect the hop limit and has minimum *modified costs* $p_{ij} = d_{ij} - \pi_j$. Note that p_{ij} is asymmetric in i and j, and we therefore work with arcs $(i, j) \in A$ rather than undirected edges. To ensure feasibility, we terminate the procedure once the tree reaches the capacity bound Q^T. The reduced cost of the resulting tree is computed as the sum of the modified costs of its arcs minus μ_T.

Since an optimal tree may contain fewer than Q^T customers, we also examine all intermediate trees whose total modified cost is less than μ_T, i.e., the current tree has a negative reduced cost. Because the heuristic is extremely fast, we apply it for every customer $i \in C$, initializing the set as $S = \{o^T, i\}$ so that the first selected arc is $(o^T, i) \in A$.

We apply a local search procedure to trees generated by the Prim-based heuristic whenever the constructed tree does not already exhibit a negative reduced cost. We use three different

neighborhoods in a Variable Neighborhood Descent fashion, namely *leaf-drop*, *leaf-add*, and *leaf-swap* [87]. Each move yields a feasible tree since we ensure that connectivity, hop limit, and overall capacity constraints are respected.

Let $t = (V', E')$ be the current tree. The leaf-drop operation removes a customer from V' that is a leaf of the tree. Formally, a node $i \in V'$ is a leaf of the tree if it has no outgoing arcs in E'. For each such leaf $i \in V'$, we compute the savings obtained by deleting its incident arc $(\text{parent}(i), i) \in E'$. If there is a leaf with a positive saving, we remove the leaf with the largest positive saving and the corresponding arc from t. The leaf-add operation introduces a new customer into the tree. Specifically, a candidate node $k \in C \setminus V'$ is selected and attached as a child of some parent $j \in V'$. The parent must already belong to the tree and adding k must satisfy both the hop limit and the overall capacity limit. If a feasible pair (j, k) yields a positive saving, we select the one that minimizes the additional edge cost d_{jk}. The leaf-swap neighborhood simultaneously removes one customer from the tree and introduces a new one. More precisely, a leaf node $i \in V'$ is selected for removal, and a candidate customer $k \in C \setminus V'$ is chosen to be inserted. The new customer k is attached to some parent $j \in V'$, provided that adding k does not violate the hop limit. The improvement of such a move is measured as the cost difference between the removed edge $(\text{parent}(i), i)$ and the newly added edge (j, k), adjusted by the corresponding dual terms. Among all feasible triplets (i, j, k), we select the one yielding the largest savings, if any such improving move is available.

3.4.3 Branching and Valid Inequalities

In this section, we introduce branching rules used to ensure integrality of solutions to the RMP and describe valid inequalities to strengthen the linear relaxation.

First, to finally compute integer solutions for Formulation (3.1), we explain how branching decisions are applied to fractional solutions of the RMP to enforce integrality. Let $(\Upsilon_t^*, \lambda_r^*)$ be a fractional primal solution of the current RMP. We first branch on the total number of routes if $K^* = \sum_{r \in R} \lambda_r^*$ is fractional. Two children nodes result from the constraints $\sum_{r \in R} \lambda_r \leq \lfloor K^* \rfloor$ and $\sum_{r \in R} \lambda_r \geq \lceil K^* \rceil$. While the first constraint simply replaces the fleet-size constraint (3.1d) of the RMP, the second one introduces a new constraint in the RMP with a non-positive dual price, which can be handled in the same manner as μ_R.

If the number of routes is integral, we branch on edges enforcing either that the edge is used by the tree or a route, or not used at all. To this end, we compute, for each edge $\{i, j\} \in E$, the value

$$e_{ij}^* = \sum_{t \in T} g_{ijt} \Upsilon_t^* + \sum_{r \in R} h_{ijr} \lambda_r^*,$$

where binary indicators g_{ijt} and h_{ijr} have the value 1, if and only if route r or tree $t \in T$ contains the edge (or one of its corresponding anti-parallel arcs), respectively. We select an edge with a value e_{ij}^* closest to 0.5. The zero-branch can be enforced directly on the underlying network by removing the edge $\{i, j\}$ and the arcs (i, j) and (j, i) from the respective graphs. Trees and routes incompatible with this branching decision are temporarily removed from the RMP. The one-branch introduces a new constraint $\sum_{r \in R} g_{ijr} \lambda_r + \sum_{t \in T} h_{ijt} \Upsilon_t \geq 1$ into the RMP. Its dual price can be directly incorporated into the modified costs in the tree-generation procedures (Sections 3.4.2) and into the reduced cost of arcs in the labeling algorithms (Section 3.4.1).

Subset Row Inequalities: In addition, we adapt *Subset Row Inequalities*(SRIs), which were originally introduced by [101] for the *vehicle routing problem with time windows*. As proposed by [101], we restrict ourselves to SRIs defined on subsets $S \subseteq C$ with three customers, i.e., $|S| = 3$, because they can be separated by straightforward enumeration. The corresponding SRI is:

$$\sum_{r \in R} \left\lfloor \frac{g_r^S}{2} \right\rfloor \lambda_r \leq 1, \tag{SRI}$$

where g_r^S counts the number of customers from S served by route r. SRIs comprise a family of non-robust cuts meaning that for each SRI with a positive dual value an additional binary attribute has to be added in the labeling algorithm and dealt with in the dominance. Therefore, we restrict the number of SRIs to 30 in our computational study.

3.5 Partial Decomposition-Based Column Generation for Tree Generation

In this section, we present an alternative decomposition strategy in which all tree-related constraints are incorporated directly into the master problem. As a result, the tree is constructed within the master problem, while column generation is applied exclusively to the route variables. The primary motivation is eliminating the need to repeatedly solve a complex tree pricing problem during column generation. We first present two adapted master problems in Section 3.5.1, and then describe the corresponding branching rules and valid cutting planes in Section 3.5.2.

3.5.1 Adapted Master Problem

In this section, analogous to the tree-generation pricing problem in Section 3.4.2, we present two mathematical models for the STPRH embedded in the RMP: the layer-assignment formulation and the partial-ordering formulation. Each model leads to a combined master problem that integrates the route-selection decisions while explicitly capturing the STPRH structure. To this end, we adapt the original master formulation (3.1) by retaining the binary route variables λ_r for each feasible route $r \in R$, while removing the tree variables Υ_t. Instead, the tree is represented through the following variables: (i) binary node variables u_i for $i \in C$ indicating whether node $i \in V$ belongs to the tree; (ii) binary arc variables x_{ij} indicating whether arc $(i,j) \in A$ is part of the tree; and (iii) binary layer variables y_{ih} specifying the position of node $i \in C$ with respect to layer h. As before, the precise definition of y_{ih} depends on the chosen formulation. The resulting combined master problems are given below.

Layer-assignment-based adapted master problem. Incorporating constraints (3.2b)–(3.2g) from model (3.2), where a binary layer variable $y_{ih} = 1$ indicates that node i is assigned to layer h, yields the combined master problem under the layer-assignment-based formulation.

$$z_{Par}^{LA} = \min \sum_{(i,j)\in A} d_{ij}x_{ij} + \sum_{r\in R} c_r\lambda_r \tag{3.4a}$$

$$\text{subject to } u_i + \sum_{r\in R} a_{ir}\lambda_r = 1 \qquad \forall i \in C \tag{3.4b}$$

$$\sum_{r\in R} \lambda_r \le K \tag{3.4c}$$

$$(3.2b) \text{ to } (3.2g)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{3.4d}$$

$$y_{ih} \in \{0,1\} \qquad \forall i \in C,\ 1 \le h \le H \tag{3.4e}$$

$$u_i \in \{0,1\} \qquad \forall i \in C \tag{3.4f}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in R \tag{3.4g}$$

The objective (3.4a) minimizes the total cost. Constraints (3.4b) ensure that each customer is either served by the tree or one of the routes. Constraint (3.4c) is the fleet-size constraint. Finally, the domain of the variables are stated in constraints (3.4d)–(3.4g). Note that Constraint (3.2h) is omitted as we allow that all customers are served by truck.

Partial-ordering-based adapted master problem. Analogously, integrating constraints (3.3b)–(3.3k) from model (3.3), where a binary layer variable $y_{ih} = 1$ specifies that node i lies above layer h, yields the combined master problem under the partial-ordering-based formulation.

$$z_{Par}^{PO} = \min \sum_{(i,j)\in A} d_{ij}x_{ij} + \sum_{r\in R} c_r\lambda_r \tag{3.5a}$$

$$\text{subject to} \quad u_i + \sum_{r\in R} a_{ir}\lambda_r = 1 \qquad \forall i \in C \tag{3.5b}$$

$$\sum_{r\in R} \lambda_r \leq K \tag{3.5c}$$

$$(3.3b) \text{ to } (3.3k)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{3.5d}$$

$$y_{ih} \in \{0,1\} \qquad \forall i \in C,\ 0 \leq h \leq H \tag{3.5e}$$

$$u_i \in \{0,1\} \qquad \forall i \in C \tag{3.5f}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in R \tag{3.5g}$$

The objective function (3.5a) seeks to minimize the overall cost. Constraints (3.5b) guarantee that every customer is assigned to either the tree or one of the vehicle routes. Constraint (3.5c) enforces the limitation on the fleet size. As above, we do not enforce an outgoing arc from the root, i.e., we omit Constraint (3.3l). The variable domains are defined in Constraints (3.5d)–(3.5g).

The column generation for route variables in both adapted master problems follows the same framework described in Section 3.4.1. In the RMP, the set of all feasible routes R is replaced by a restricted subset $\bar{R} \subset R$ and binary conditions (3.5d)–(3.5g) are relaxed. The RMP is solved iteratively: at each iteration, route variables (columns) with negative reduced cost are identified and added to the RMP. This procedure repeats until no further route variables with negative reduced cost can be found. Afterwards, branching is usually required to obtain integer solutions.

3.5.2 Valid Inequalities and Branching

To further strengthen the linear relaxation of the adapted master problems in the partial decomposition, we incorporate several families of valid inequalities, namely the SEC and the SEC_u, and the SRI. These cuts are presented and discussed in detail in Section 3.4.2 and Section 3.4.3. Due to the weaker master problem, we allow for up to 300 SRIs. Additionally, we do not limit the number of separated SECs as they have no impact on the pricing problem.

Branching strategies: To finally obtain integer solutions, we apply a similar branching strategy as described in Section 3.4.3. Let $(\lambda_r^*, u_i^*, x_{ij}^*, y_{ih}^*)$ be a fractional solution to the RMP (3.4) or (3.5). Branching on the total number of routes works exactly as described in Section 3.4.3. Regarding branching on edges, the only difference is the computation of the value b_{ij}^*. We compute, for each edge $\{i,j\} \in E$,

$$b_{ij}^* = x_{ij}^* + \sum_{r\in R} h_{ijr}\lambda_r^*,$$

where indicators h_{ijr} have the value 1, if and only if the route r contains the edge $\{i,j\}$.

In addition, we need a branching rule on the binary layer variables y_{ih}. Two child nodes are then created: one enforcing $y_{ih} = 1$ and the other enforcing $y_{ih} = 0$. Note that both decisions have no impact on the route-generation pricing problem. Branching follows a strict order: first on the number of vehicles λ_r, then on the edge variables x_{ij}, and afterwards on the layer variables y_{ih}.

Note that we do not branch on the node-selection variables u_i as the pretest has indicated this is not beneficial.

Moreover, our results indicate that nodes located on layers farther from the root have a larger impact on the solution structure than those closer to the root. Consequently, we prioritize branching on y_{ih} variables corresponding to larger hop-limit values h. If there are several fractional variables on the same largest layer, we choose one with a value y^*_{ih} closest to 0.5.

For both formulations, massive branching is required to solve the problem to integer optimality. We therefore decided to use *strong branching* as follows: We use a strictly hierarchical branching, i.e., i) we first branch on the number of vehicles, ii) if the number of vehicles is integer and b^*_{ij} is fractional for some $(i, j) \in A$ then we choose up to six most fractional candidates in the current solution, iii) if also all arcs are binary, we choose six candidates y_{ih} with fractional values according to the largest layer position h. In cases ii) and iii), we perform a rough evaluation of each candidate by solving the current RMP twice, adding the constraint corresponding to each child node, and using only the pricing heuristics to generate additional columns. The resulting improvements in the lower bounds are usually overestimated. However, this evaluation strategy is faster than a full evaluation and returns better decisions than simply choosing the most fractional candidates to branch on. The candidate to branch on is then chosen according to the product rule [2]. As a branch-and-bound node-selection rule, we apply a best-bound-first strategy.

3.6 Computational results

This section presents the computational results obtained for the four BPC algorithms developed in this study. The algorithms correspond to two decomposition strategies, full and partial, and, within each, two formulations of the STPRH, namely the layer-assignment-based and the partial-ordering-based models. The following abbreviations are used to refer to these algorithms:

FulLA : The layer-assignment-based formulation in full decomposition
FulPO : The partial-ordering-based formulation in full decomposition
ParLA : The layer-assignment-based formulation in partial decomposition
ParPO : The partial-ordering-based formulation in partial decomposition

All experiments are conducted on the high-performance computing cluster (HPC3) at the University of Vienna. Each computing node is equipped with two AMD EPYC 7452 32-Core processors (2.3 GHz) and 256 GB of RAM. The system runs `CentOS 7 (Core)` and uses `SLURM 20.11.8` as the job scheduler. The performance of a single thread on this cluster is slightly lower than that of a modern desktop processor; therefore, all results are based on single-threaded execution to ensure comparability. All algorithms are implemented in `C++` and compiled with `GCC 11.4.0` in release mode (64-bit). The callable library of `CPLEX 22.11.0` is used to (re-)optimize RMP for all four algorithms and to solve the tree-generation pricing problem exactly in both full decomposition algorithms. CPLEX's default parameters are retained, except that the number of threads is set to one. We used a hard CPU time limit of 7200 seconds for all the runs. If an instance cannot be solved to optimality in the time limit, we use CPLEX to solve the RMP with the current pool of columns to optimality, granting a time limit of 600 additional seconds.

Instance generation. To generate benchmark instances, we adapt the classical *Set A* of Augerat [15] from the CVRPLIB. For the CWCP with a hop limit, the first customer of each instance is designated as the AWCS root o^T. The cost of a tree edge is defined as $d_{ij} = 2 * c_{ij}$, reflecting that AWCS installation is generally more expensive than truck routing. The tree capacity is set to $Q^T = \lfloor |C|/2 \rfloor$. We generate one instance for each hop limit value $H \in \{3, 5, 6, 9, 12, 15\}$ capturing a representative range of operational network depths.

In the following, we analyze the linear relaxation results of the four BPC algorithms in Section 3.6.1. Section 3.6.2 reports integer results when cutting and branching is applied. Section 3.6.3 examines the impact of multiple SEC variants on all models. Finally, Section 3.6.4

investigates the impact of three key instance parameters: the hop limit, the tree capacity, and the cost ratio between tree and routing costs.

3.6.1 Linear relaxation results

This subsection analyzes the performance of the four BPC algorithms, FulLA, FulPO, ParLA, and ParPO, at the linear relaxation level. The analysis aims to evaluate the tightness of the LP bounds and the computational effort required to solve the root-node relaxations for different algorithms and hop limit values H.

In this section and in Section 3.6.2, we present results only over those instances in which all four algorithms were able to solve the root node successfully for all hop limit values, i.e., 20 out of 27 instances (comprising up to 69 nodes). Table 3.1 summarizes the relevant results under the following header titles.

H : Hop limit used for the AWCS tree
 # sol : Number of instances (out of all 27) solved to LP-optimality
Time root : Minimum/average (avg)/maximum computation time (in seconds) required to solve the root node
Gap root : Minimum (min)/geometric mean (gm)/maximum (max) relative difference between the largest lower bound at the root node, obtained from the FulLA and FulPO at the minimum hop limit of 3 (denoted as $LBroot_3^{SEP}$), and the lower bounds corresponding to each hop limit and algorithm ($LBroot_h^A$), computed as $\frac{LBroot_h^A}{LBroot_3^{SEP}}$

Table 3.1: Comparison of the LP relaxation performance of all four BPC algorithms.

	FulLA				FulPO				ParLA				ParPO			
H	**#sol**	**Time root**			**#sol**	**Time root**			**#sol**	**Time root**			**#sol**	**Time root**		
		min	**avg**	**max**		**min**	**avg**	**max**		**min**	**avg**	**max**		**min**	**avg**	**max**
3	26	0.9	38	336.4	25	0.6	74	1009.5	27	0.1	1.3	5.8	27	0.3	1.4	5.7
5	22	1.4	358.7	2093	23	1.03	351.5	3467	27	0.2	2.3	13.5	27	0.21	1.8	8.3
6	21	0.0	665.7	4967	22	0.0	270.3	2458	27	0.2	2.9	13.8	27	0.2	2.2	8.7
9	21	1.8	450.0	4525	24	1.1	119.7	1231	27	0.4	5.2	31.3	27	0.4	2.6	11.1
12	23	2.13	346.6	3200	26	1.2	64	370.1	27	0.6	7.7	36.6	27	0.4	4.1	21.3
15	22	1.99	582.8	6747	27	1.2	38.5	194	27	0.7	12.4	71.8	27	0.6	4.6	24.1

	Gap root (FulLA / FulPO)			**Gap root (ParLA)**			**Gap root (ParPO)**		
	min	**gm**	**max**	**min**	**gm**	**max**	**min**	**gm**	**max**
3	1.00	1.00	1.00	0.79	0.89	0.99	0.79	0.89	0.99
5	0.93	0.98	1.00	0.78	0.88	0.99	0.78	0.88	0.99
6	0.91	0.97	1.00	0.78	0.88	0.99	0.78	0.88	0.99
9	0.87	0.96	1.00	0.78	0.88	0.99	0.78	0.88	0.99
12	0.86	0.95	1.00	0.78	0.88	0.99	0.78	0.88	0.99
15	0.85	0.95	1.00	0.78	0.88	0.99	0.78	0.88	0.99

Table 3.1 compares the LP relaxation performance of the four BPC algorithms across all hop limits. Numerical results on the tightness of the LP relaxation clearly show the superiority of the full decompositions (FulLA and FulPO) over their partial counterparts for every hop limit. Between the two full versions, FulPO consistently outperforms FulLA, achieving lower average root-node computation times and solving a larger number of root LPs successfully. Note that the bounds of both full decomposition algorithms are the same, as the same master problem formulation is solved.

Both ParLA and ParPO exhibit weaker LP relaxations, with geometric mean lower-bound ratios typically 7–11% smaller than those of the full decompositions. The minimum differences range 6–21%, while the maximum differences reach 1% across all hop limits. There were slight differences in the root gap values of ParPO and ParLA, not visible in Table 3.1 due to rounding,

indicating a slight dominance of ParPO. Nevertheless, they solve the root LP substantially faster, requiring only a few seconds on average compared with several hundreds or even thousands of seconds for the full decompositions. The partial decomposition algorithms can solve the linear relaxations of all 27 instances, whereas the full decomposition methods occasionally fail to solve instances with more than 46 nodes within the 7200-second time limit.

Increasing the hop limit H slightly deteriorates the LP relaxation for FulLA and FulPO, whose geometric-mean ratios decrease from 1.00 at $H = 3$ to about 0.95 at $H = 15$. This moderate (5%) decline indicates that their LP bounds remain tight even for deeper network structures. In contrast, the effect of H on the LP tightness of ParLA and ParPO is negligible, as their geometric-mean ratios remain nearly constant across all hop limits. This indicates that the hop limit has only a minor impact on optimal LP solutions when partial decomposition is used.

Finally, when comparing the two STPRH formulations, the partial-ordering formulation (FulPO and ParPO) clearly dominates the layer-assignment formulation (FulLA and ParLA) across all quality metrics. Overall, the relative preference between FulPO and ParPO depends on the decision-maker's priorities, whether to emphasize stronger LP bounds or faster root-level computation, particularly for larger problem instances.

3.6.2 Integer results

Subsequently, this section extends the evaluation to overall algorithmic performance, focusing on the *Integer Programming* (IP) results. The analysis examines each algorithm's ability to close optimality gaps, achieve proven optimality, and quantifies pricing effort as well as branching and cutting. Table 3.2 reports the main performance indicators for each algorithm and hop limit. Recall that averages are computed only over those 20 instances where all algorithms were able to solve the linear relaxation for all hop limits.

 # opt : Number of instances (out of all 27) solved to proven optimality by the BPC algorithm
GAP : Minimum/average/maximum optimality gap at the end of optimization, computed as $100 \cdot (UB - LB)/UB$ where UB and LB are the best upper and lower bounds found by the algorithm. The reported average includes 0.0 gap values for all instances that are solved to proven optimality.
Time : Average computation time of the BPC algorithm in seconds
%Pricing : Average percentage of the total computation time spent in the pricing component of the BPC algorithm, including both heuristic and exact pricing solver
 # B&B : Average number of branch-and-bound nodes solved in the RMP
 # Cuts : Average number of generated cuts in the RMP

Table 3.2: Comparison of the IP performance of all four BPC algorithms.

	H	#Opt	GAP			Time	%Pricing	#B&B	#Cuts
			min	avg	max				
FulLA	3	19	0.19	0.09	1.1	1917	98.5	57	71
	5	15	0.10	0.33	2.79	2594	99.7	21	48
	6	14	0.55	0.39	3.52	2909	99.8	11	45
	9	13	0.12	0.39	2.10	2930	99.8	13	33
	12	13	0.09	0.42	3.38	3223	99.8	23	31
	15	15	1.32	0.45	2.58	2993	99.7	24	29
FulPO	3	18	0.19	0.20	1.33	2156	99.1	38	70
	5	15	0.10	0.29	3.17	2361	99.6	27	49
	6	15	0.09	0.27	3.03	2272	99.7	17	48
	9	15	0.27	0.25	1.69	2158	99.3	26	37
	12	18	0.73	0.22	1.99	1802	99.0	40	38
	15	20	0.63	0.11	1.50	1699	98.7	50	38
ParLA	3	6	0.13	2.96	13.46	5392	17.3	2667	239
	5	6	0.23	1.65	7.31	5315	11.7	1525	322
	6	9	0.27	0.92	4.27	4266	8.8	974	246
	9	14	0.41	0.34	2.26	2510	5.9	344	141
	12	16	0.13	0.11	0.83	2475	5.0	337	154
	15	18	0.20	0.10	0.93	2233	3.8	207	62
ParPO	3	5	0.83	3.45	13.51	5782	16.3	2708	157
	5	7	0.38	1.59	6.86	5048	12.5	1332	143
	6	10	0.14	0.89	4.17	4147	10.1	845	89
	9	14	0.41	0.26	1.80	2529	6.3	332	68
	12	17	0.21	0.11	1.24	2124	5.2	249	63
	15	18	0.14	0.08	1.04	2226	4.2	226	62

Table 3.2 shows that the full decomposition algorithms outperform their partial counterparts in terms of instances solved to proven optimality. Note that the differences are significantly smaller for larger hop limit values. The results regarding the gaps are more differentiated. For small hop limits ($H = 3, 5, 6$) the full decomposition algorithms (FulLA, FulPO) produce significantly smaller average optimality gaps than the partial decomposition algorithms: e.g., at $H = 3$ the average gaps are 0.09 (FulLA) and 0.20 (FulPO) versus 2.96 (ParLA) and 3.45 (ParPO). However, this advantage shrinks as H increases. For $H = 9$, the gaps become comparable (FulLA: 0.39, FulPO: 0.25, ParLA: 0.34, ParPO: 0.26), and for larger hop limits ($H = 12, 15$) the partial decomposition algorithms actually achieve better average gaps than the full decomposition ones (e.g., at $H = 15$ the averages are FulLA: 0.45, FulPO: 0.11, ParLA: 0.10, ParPO: 0.08). In short, full decomposition dominates for low H, while partial decomposition is competitive and even superior for larger H values, despite the fact that less instances are solved to proven optimality.

The distribution of computational effort further distinguishes the two types of decomposition. The full decomposition variants spend about 99% of their total runtime in pricing independently of the hop limit, reflecting the high cost of repeatedly solving the tree-generation pricing problem exactly. Conversely, the partial variants allocate less than 18% of total computation time to pricing, shifting the effort toward solving the master problem, where a substantially larger number of branch-and-bound nodes are explored and a larger number of cuts are generated by the BPCs based on partial decomposition. Moreover, as the number of y_{ih}-variables in the RMP grows when the hop limit increases, the percentage of computation time spent in pricing further reduces to around 4% for $H = 15$ as more time is needed to solve the RMP.

Consistent with the LP analysis, among the two STPRH formulations, the partial-ordering-based model again demonstrates superior overall performance. Both FulPO and ParPO outperform their layer-assignment counterparts in terms of bound tightness, number of optimal solutions, and total computation time.

To give a holistic view of the results, Table 3.3 presents averaged results for the seven instances for which at least one of the full decomposition algorithms, and in some cases both, failed to solve the root relaxation within 7200 seconds for at least one hop limit value. Since the full decomposition variants do not produce a valid lower bound on these instances, the gap cannot be computed for these variants. Therefore, Table 3.3 does not report gaps; instead, columns for the average upper bound (**avg. UB**) and the average lower bound (**avg. LB**) are reported. The remaining columns are the same as in Table 3.2.

Table 3.3: Comparison of the IP performance of all four BPC algorithms on 7 instances unsolved at the root for at least one algorithm and hop limit value.

	H	**#Opt**	**avg UB**	**avg LB**	**Time**	**%Pricing**	**#B&B**	**#Cuts**
FulLA	3	2	1239		5493	99.4	28	65
	5	0	1197		7204	99.9	0	4
	6	0	1167		7203	99.9	2	25
	9	0	1134		7203	99.9	0	0
	12	0	1115		7205	99.9	1	5
	15	0	1103		7203	99.9	1	8
FulPO	3	2	1245		5768	99.8	8	57
	5	0	1193		7203	99.9	1	16
	6	0	1170		7203	99.9	1	8
	9	0	1125		7204	99.8	2	17
	12	1	1105		6539	99.7	7	40
	15	2	1093		5821	99.6	32	66
ParLA	3	0	1243	1099	7428	14.1	1169	160
	5	0	1203	1083	7817	8.7	628	135
	6	0	1186	1082	7813	6.5	501	124
	9	0	1136	1078	7719	6.2	385	114
	12	1	1121	1078	7540	4.9	240	105
	15	2	1097	1076	6521	3.8	166	98
ParPO	3	0	1249	1089	7483	11.4	929	150
	5	0	1184	1082	7741	7.9	567	131
	6	0	1162	1080	7783	6.9	474	123
	9	0	1118	1079	7539	6.7	381	113
	12	1	1097	1077	7284	5.2	284	109
	15	2	1087	1076	5871	4.4	171	102

Table 3.3 shows that FulPO performs slightly better than FulLA, solving one instance to optimality at $H = 12$ and two instances at $H = 15$. Compared to Table 3.2, both FulLA and FulPO remain clearly limited by the expensive tree-pricing problem and can solve substantially fewer branch-and-bound nodes.

In contrast, both partial decomposition algorithms (ParLA and ParPO) successfully produce valid lower bounds for all seven instances and also obtain integer-optimal solutions for several cases. Similar to FulPO, they solve up to two instances optimally at $H = 12$ and $H = 15$.

Moreover, increasing H has a noticeably positive impact on both ParLA and ParPO: the gap between **avg. UB** and **avg. LB** narrows consistently. The **avg. LB** in ParLA is similar to ParPO with slightly better values. Except for $H = 3$, ParPO achieves the best **avg. UB** among all four algorithms.

3.6.3 Evaluation of SECs

In this section, we evaluate the impact of incorporating SEC variants across both decompositions. In our implementation, SECs are used for eliminating cycles in fractional solutions, thereby strengthening the linear relaxation and accelerating convergence to an optimal integer solution. For the full decomposition, SECs are applied within the tree-generation pricing problem. In the partial decomposition, where the tree is modeled directly within the master problem, SECs are embedded at the master level. It is worth noting that SECs were not used in the solution methodology of Jabrayilov and Mutzel [99] for the STPRBH. For each configuration, two variants are tested: one incorporating and one omitting the respective SECs (SEC and SEC_u) in each algorithm. We report the number of optimal solutions (#B&B) for all 27 datasets and all four algorithms with and without incorporating SECs, across all hop limits.

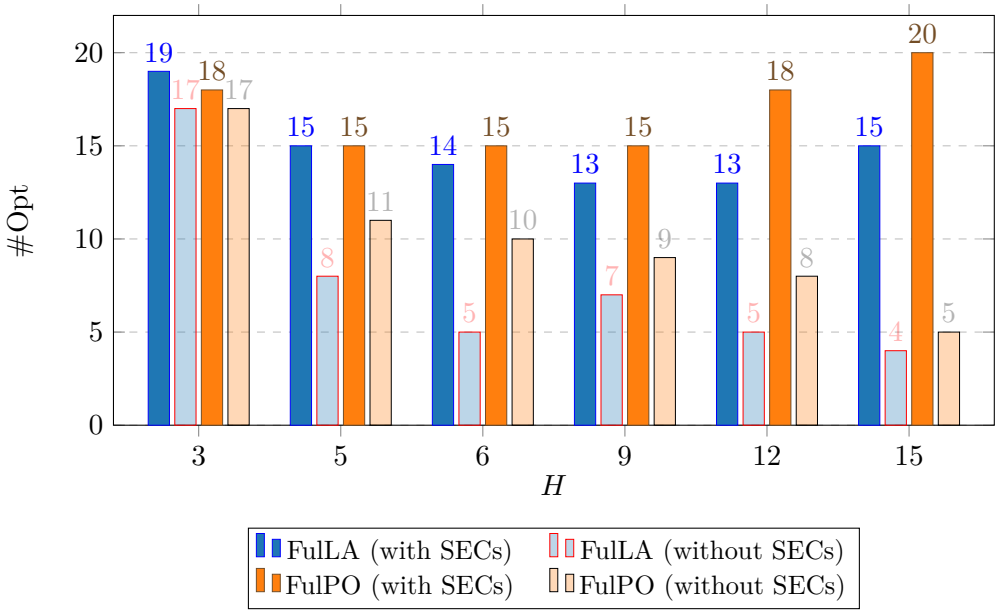


Figure 3.3: Impact of considering SECs for FulLA and FulPO.

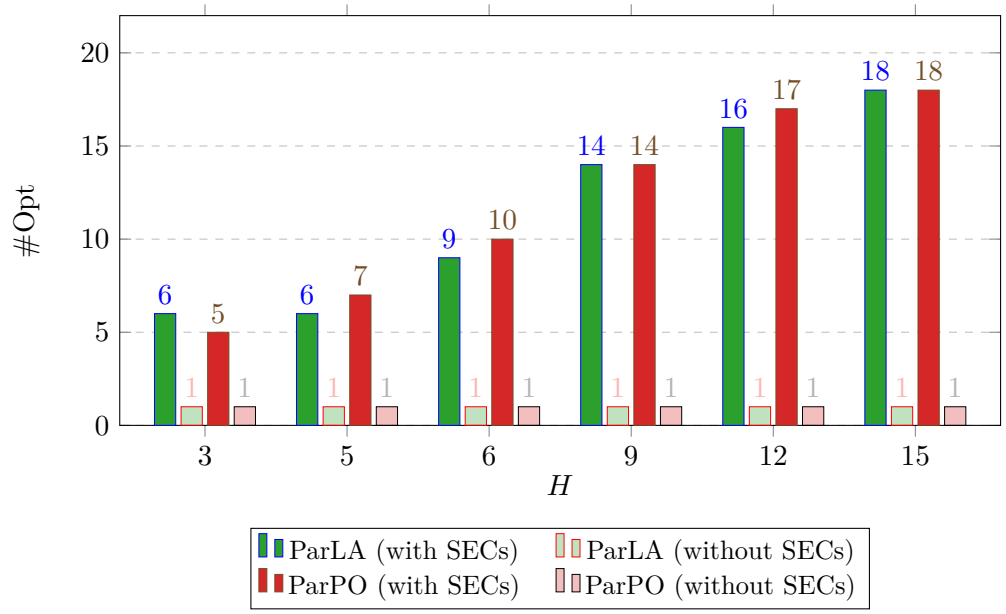


Figure 3.4: Impact of considering SECs for ParLA and ParPO.

Figures 3.3 and 3.4 illustrate that the incorporation of SECs significantly enhances the performance of all four algorithms, though the extent of improvement differs considerably between the partial and full decompositions. For ParLA and ParPO, adding SECs increases the number of proven optimal solutions from one without SECs to more than ten on average when SECs are applied. The full variants, FulLA and FulPO, also experience performance gains, albeit more moderate and incremental in nature. This positive effect is significantly stronger for larger hop limits.

3.6.4 Impact of Hop Limit, AWCS Capacity, and Cost Ratio

In the final series of experiments, we investigate how three key parameters affect the algorithms, solution structure, and total system cost of the CWCP with a hop limit: (i) the hop limit H, (ii) the tree capacity Q^T (the capacity of the AWCS), and (iii) the cost ratio f between tree costs and routing costs.

We consider two representative hop limits, $H \in \{6, 12\}$, corresponding to moderate and relatively deep AWCS structures. For the capacity of the AWCS, we analyze two tree-capacity scenarios. As before, we express this capacity as a percentage of the customers that the pneumatic system could serve. The considered percentages are 33% and 50%, i.e., $Q^T = \lfloor |C|/3 \rfloor$ and $Q^T = \lfloor |C|/2 \rfloor$. Finally, for the cost ratio f, we vary the ratio of the costs d_{ij} and c_{ij} by setting $d_{ij} = \mathrm{f} * c_{ij}$. Following the cost structures discussed in DehghanChenary et al. [53], we use three cost ratios $f \in \{1.1, 2, 3\}$. This results in $12 = 3 \times 2 \times 2$ scenarios.

For all scenarios with 50% tree capacity, Tables 3.4 present aggregated results over all 15 instances where all four algorithms successfully solved the root-node under all scenarios. Similarly, Table 3.5 reports aggregated results for the 12 instances for which at least one of the full decomposition algorithms, and in some cases both, failed to solve the root relaxation within 7200 seconds for at least one scenario. As results are quite similar for the two tree-capacity scenarios, we report the results for capacity 33% only in the Appendix 3.8 (Tables 3.7 and 3.8). The tables have the same columns as their counterparts in Section 3.6.2.

Table 3.4: Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 50%. Results are aggregated over those 15 instances for which the LP-Relaxation could be solved for all scenarios and all algorithms.

	H	f	**#Opt**	**GAP**			**Time**	**%Pricing**	**#B&B**	**#Cuts**
				min	**avg**	**max**				
FulLA	6	1.1	7	0.27	1.21	4.14	4482	99.8	24	18
	6	2	14	0.27	0.11	1.37	1532	99.8	17	40
	6	3	19	0.34	0.02	0.34	1308	94.7	100	66
	12	1.1	8	0.29	0.94	4.43	4155	99.4	77	18
	12	2	14	1.51	0.33	3.38	2033	99.7	33	27
	12	3	21	0.76	0.05	0.76	1148	94.4	81	60
FulPO	6	1.1	9	0.13	1.01	5.00	4017	99.6	46	19
	6	2	15	1.57	0.10	1.57	1239	99.8	22	40
	6	3	20	0.08	0.01	0.08	1097	93.0	113	66
	12	1.1	19	0.40	0.38	2.72	2490	98.0	165	21
	12	2	19	0.30	0.11	1.37	1121	98.8	67	29
	12	3	23	0.25	0.02	0.25	1086	93.0	119	62
ParLA	6	1.1	3	0.36	1.96	4.10	5979	10.2	1489	108
	6	2	9	0.33	0.69	5.68	3463	8.3	748	166
	6	3	17	0.59	0.12	1.14	1404	15.7	315	182
	12	1.1	10	0.16	0.31	1.46	4212	7.5	500	78
	12	2	15	0.13	0.12	1.31	2113	4.8	262	128
	12	3	19	0.12	0.04	0.42	1418	10.8	191	37
ParPO	6	1.1	4	0.87	1.80	3.60	5571	10.1	1936	126
	6	2	10	0.27	0.47	3.63	3130	10.2	885	79
	6	3	18	0.25	0.04	0.28	1378	13.6	320	44
	12	1.1	11	0.15	0.18	1.14	3748	9.1	648	82
	12	2	16	0.14	0.06	0.81	1258	4.8	221	52
	12	3	21	0.25	0.02	0.25	1116	8.7	194	40

Table 3.5: Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 50% on 12 instances unsolved at the root for at least one of the full decomposition algorithms in at least one scenario.

	H	f	**#Opt**	**avg UB**	**avg LB**	**Time**	**%Pricing**	**#B&B**	**#Cuts**
FulLA	6	1.1	0	878		7202	99.9	0	0
	6	2	1	1113		6775	99.9	3	30
	6	3	5	1230		4710	99.5	20	111
	12	1.1	1	827		6862	99.9	2	10
	12	2	2	1064		6638	99.9	4	22
	12	3	7	1210		4077	99.2	37	80
FulPO	6	1.1	0	867		7202	99.9	1	3
	6	2	2	1108		6128	99.9	3	36
	6	3	6	1228		4277	99.1	37	110
	12	1.1	4	823		5685	99.8	9	24
	12	2	6	1057		4701	99.7	12	52
	12	3	9	1207		2438	98.4	46	89
ParLA	6	1.1	0	851	810	7726	7.9	566	130
	6	2	0	1115	1045	7665	8.1	434	120
	6	3	4	1231	1203	5251	13.8	330	157
	12	1.1	2	817	807	6814	4.9	261	205
	12	2	3	1068	1040	6606	4.1	141	96
	12	3	6	1234	1201	4828	6.7	127	64
ParPO	6	1.1	0	850	810	7620	9.3	688	406
	6	2	0	1094	1045	7538	8.2	553	123
	6	3	5	1225	1204	5056	13.2	418	77
	12	1.1	2	814	807	6331	5.1	249	110
	12	2	3	1054	1041	6475	4.5	253	104
	12	3	7	1206	1201	4574	10.9	283	545

Comparing the two modelling approaches, the partial-ordering-based formulation consistently outperforms the layer-assignment-based formulation in both decomposition strategies. Regarding the number of proven optimal solutions, FulPO outperforms all other algorithms, followed closely by FulLA, whereas both partial decomposition algorithms solve fewer instances on average. Nevertheless, despite the smaller number of optimality, ParPO maintains an extremely tight gap between the **avg. UB** and **avg. LB**, even in scenarios where the full decomposition algorithms find more proven optimal solutions. This effect is even more pronounced on the larger instances reported in Table 3.5, where the full decomposition algorithms frequently fail to produce a valid lower bound, while both partial variants consistently generate tight gaps which even decrease as either the hop limit or the cost ratio increases.

More precisely, increasing the hop limit clearly benefits the partial decomposition algorithms: at $H = 12$, both ParLA and ParPO achieve substantially smaller average gaps, and more optimal solutions than at $H = 6$, and in most scenarios their gaps fall below those obtained by the full decomposition algorithms. In contrast, larger hop limits do not yield a consistent reduction in gaps in FulLA and FulPO. In all algorithms, higher cost ratios increase the number of optimally solved instances and reduces the average gap. Finally, as Tables 3.7 and 3.8 in the Appendix 3.8 show, smaller AWCS capacities (33% vs. 50%) are slightly easier for nearly all scenarios and algorithms (except FulPO). Overall, instances for which the optimal solution is expected to contain a smaller tree, whether due to higher cost ratios, larger hop limits, or reduced tree capacity, tend to be computationally less demanding. In the next section, we investigate whether these parameter settings indeed lead to smaller tree structures in close-to-optimal solutions.

In addition to the previous metrics, we compute the following indicators to assess the AWCS structure and total system cost. Table 3.6 summarizes the resulting tree structures and total system costs for all 27 instances.

tree %cust.: the average percentage of customers served by the AWCS

tree %demand.: the average percentage of the waste covered by the AWCS

total %Δcost: the average percentage increase of the total cost compared to the *baseline* (BL) scenario, where BL refers to the cost ratio $f = 1.1$ and capacity 50%, i.e., the scenario in which the overall system is the cheapest

Although the differences among algorithms in terms of total cost are negligible, indicating that the algorithmic formulation primarily affects computational efficiency rather than the resulting solution structure, ParPO consistently identified the best solution in more than 95% of instances across all settings and scenarios. Therefore, the results reported in Table 3.6 correspond to those obtained using the ParPO algorithm.

Table 3.6: Impact of hop limit (H), tree capacity (Q^T), and cost ratio (f) on AWCS utilization and total cost.

		Capacity 33%			**Capacity 50%**		
H	f	**%cust.**	**%demand.**	**%Δcost**	**%cust.**	**%demand.**	**%Δcost**
6	1.1	33	33	+14	49	49	+3
6	2	25	26	+35	35	34	+32
6	3	9	10	+45	10	10	+45
12	1.1	33	33	+12	49	48	BL
12	2	28	29	+32	39	40	+29
12	3	10	10	+44	11	12	+43

Table 3.6 shows that both the share of total waste and the number of customers served by the AWCS decline sharply as the cost ratio f increases, from full utilization of the tree capacity at $f = 1.1$ to about $20 - 30\%$ at $f = 3$. This is not surprising as increasing the cost factor from 1.1 to 3 means that the cost of the AWCS rise by around 270%. While, in our results, the corresponding total system cost rises only by roughly 31–43%.

Larger tree capacities Q^T lead to significantly larger trees and slightly reduce the overall system cost. The impact becomes smaller when the cost ratio f is increased. A similar behavior can be observed when the hop limit is increased from 6 to 12.

DehghanChenary et al. [53] highlight the role of demand distribution in shaping AWCS structures. In our instances, demand is nearly uniform, most customers generate a medium amount of waste, with only a few having notably larger demands. Therefore, the percentage of total demand allocated to the AWCS follows almost the same trend as the percentage of customers served.

3.7 Conclusions and outlook

This study presents an extension of the CWCP by incorporating a more realistic operational constraint: a hop limit on each branch of the AWCS pipeline network. By explicitly modeling this hop limit, the problem accounts for practical physical restrictions, such as pressure loss in pneumatic transport systems, effectively transforming the tree-design component into the STPRH. To solve the CWCP with a hop limit exactly, we used the branch-and-price paradigm. This comprehensive solution approach simultaneously determines, for each collection point, whether it is served by the AWCS or by truck, while constructing both the AWCS network and the corresponding truck routes. Moreover, we developed two decomposition strategies: (i) a full

decomposition, in which the routing and AWCS tree-generation are treated separately and independently within the pricing problems, and (ii) a partial decomposition, where the AWCS tree structure is modeled explicitly in the master problem and column generation is applied only to the routing component. In addition, for the STPRH component, we incorporated and adapted two state-of-the-art formulations originally proposed by Jabrayilov and Mutzel [99] (partial ordering) and Sinnl and Ljubić [159] (layer assignment). Combining the two decomposition approaches with these two formulations yields four distinct branch-price-and-cut algorithms.

Computational experiments on benchmark data sets derived from the classical *Set A* of Augerat [15] were conducted to compare all algorithms, revealing several key insights: First, the choice of decomposition dominates computational behavior. As expected, the full decomposition strategies provide significantly tighter bounds than the partial decomposition. Within the full decomposition, the partial-ordering formulation (FulPO) achieves the best balance between relaxation strength and computational effort. Moreover, FulPO is able to solve the largest number of instances to optimality. However, solving the pricing problem to optimality is computationally expensive, and both full decomposition variants fail to solve the root node for larger data sets. By contrast, partial decomposition is far more scalable: it always produces valid lower bounds in our experiments and reliably returns high-quality integer solutions on the largest data sets.

Second, the partial-ordering (PO) formulation consistently outperforms the layer-assignment (LA) formulation across all metrics and both decomposition schemes. Consequently, the combination of partial decomposition and partial-ordering (ParPO) attains the best practical performance: ParPO finds the best-known upper bounds in over 95% of all instances, is robust on large data sets where full decomposition times out when solving the LP, and often achieves small or even negligible optimality gaps.

Third, the relative advantage of full versus partial decomposition depends on instance characteristics. For small-to-moderate data sets (up to roughly 46 nodes in our experiments) and shallow trees (small hop limit), FulPO is preferable when tight LP bounds and the most significant number of proving optimality are required. As the instance size or expected tree size grows, ParPO becomes the practical default. Notably, several parameter changes, including larger hop limits, higher tree-to-route cost ratios, and smaller tree capacity, consistently make instances easier to solve. In such scenarios, all four algorithms show improved performance and solve a greater number of data sets to proven optimality.

Finally, several implementation and modeling insights have practical importance. The inclusion of Subtour Elimination Constraints (SECs) substantially improves efficiency across all algorithms, particularly for the partial decomposition variants.

Outlook. Future research could extend this work in several directions. On the methodological side, integrating advanced acceleration techniques, such as dynamic cut selection and dual stabilization, could further enhance computational scalability. Alternative schemes, including column-and-row generation approaches, may also improve efficiency for large-scale data sets. Finally, from a modeling perspective, incorporating stochastic or dynamic demand, time-dependent routing costs, or multi-source AWCS networks would broaden the applicability of the CWCP.

3.8 Appendix

Tables 3.7 and 3.8 in the following complement the analysis in Section 3.6.4 by reporting the detailed results for the scenarios with a tree capacity of 33%, i.e., $Q^T = \lfloor |C|/3 \rfloor$. These tables parallel the presentation of Tables 3.4 and 3.5 and confirm that the same qualitative algorithmic behavior holds across both capacity settings.

theoremstyledefinition

Table 3.7: Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 33%. Results are aggregated over those 15 instances for which the LP-Relaxation could be solved for all scenarios and all algorithms.

	H	f	**#Opt**	**GAP**			**Time**	**%Pricing**	**#B&B**	**#Cuts**
				min	**avg**	**max**				
FulLA	6	1.1	9	0.37	0.84	5.61	4850	99.4	109	35
	6	2	15	1.97	0.13	1.97	807	99.7	19	45
	6	3	20	0.59	0.04	0.59	1083	95.1	79	63
	12	1.1	10	0.12	0.67	3.88	4211	99.4	83	37
	12	2	15	2.65	0.18	2.65	1285	99.5	35	43
	12	3	20	0.68	0.05	0.68	1530	94.5	128	66
FulPO	6	1.1	9	0.15	0.30	1.34	4230	98.8	176	35
	6	2	17	1.41	0.09	1.41	678	99.5	25	53
	6	3	20	0.17	0.01	0.17	884	94.6	85	64
	12	1.1	12	0.14	0.51	4.82	3424	97.9	226	38
	12	2	19	1.80	0.12	1.80	639	98.0	39	45
	12	3	22	0.17	0.01	0.17	1083	94.1	107	69
ParLA	6	1.1	4	0.28	0.77	2.46	5747	10.2	1335	176
	6	2	12	0.77	0.38	3.85	2750	7.9	546	72
	6	3	17	0.09	0.06	0.85	1432	15.4	289	89
	12	1.1	8	0.15	0.51	3.01	4509	6.4	529	93
	12	2	14	0.12	0.25	3.59	2609	4.4	230	58
	12	3	20	0.59	0.04	0.59	1593	9.7	183	39
ParPO	6	1.1	4	0.14	0.75	2.13	5585	10.8	2026	486
	6	2	12	0.77	0.42	3.85	2356	9.6	650	74
	6	3	18	0.19	0.04	0.34	1336	12.3	323	41
	12	1.1	11	0.14	0.41	1.81	4167	7.3	816	98
	12	2	16	2.39	0.16	2.39	1669	5.2	231	60
	12	3	21	0.25	0.02	0.25	1274	7.3	209	41

Table 3.8: Impact of hop limit (H) and cost ratio (f) on the performance of BPC algorithms with tree capacity 33% on 12 instances unsolved at the root for at least one of the full decomposition algorithms in at least one scenario.

	H	f	**#Opt**	**avg UB**	**avg LB**	**Time**	**%Pricing**	**#B&B**	**#Cuts**
FulLA	6	1.1	2	952		7010	99.8	4	39
	6	2	1	1124		6804	99.8	6	47
	6	3	6	1226		4679	98.9	67	92
	12	1.1	3	928		6240	99.8	11	50
	12	2	1	1091		7163	99.7	22	52
	12	3	6	1215		4402	98.9	46	88
FulPO	6	1.1	1	950		6796	99.8	9	45
	6	2	3	1117		6247	99.8	16	66
	6	3	6	1226		4221	98.8	75	92
	12	1.1	3	924		5549	99.4	43	61
	12	2	5	1091		5167	99.1	52	72
	12	3	8	1212		3093	98.1	54	93
ParLA	6	1.1	0	943	917	7594	7.2	481	132
	6	2	0	1124	1080	7615	10.1	437	138
	6	3	4	1227	1206	5324	13.4	299	116
	12	1.1	1	925	914	7259	4.6	233	171
	12	2	1	1097	1076	7507	4.4	177	99
	12	3	6	1235	1204	5221	6.1	133	75
ParPO	6	1.1	0	941	917	7520	8.3	616	164
	6	2	0	1118	1079	7537	10.9	672	226
	6	3	5	1224	1207	4991	14.1	408	77
	12	1.1	3	921	915	6237	5.7	270	120
	12	2	2	1087	1077	6875	5.8	280	108
	12	3	7	1211	1204	4465	7.7	159	139

Chapter 4

TSCDA: A dynamic two-stage community discovery approach

Published in: *Social Network Analysis and Mining*
TSCDA: A dynamic two-stage community discovery approach.
A. Ferdowsi & M.DehghanChenary. © 2022 The Authors.
10.1007/s13278-022-00874-z

Abstract In this paper, we introduce a new approach for detecting community structures in networks. The approach is subject to modifying one of the connectivity-based community quality functions based on considering the impact that each community's most influential node has on the other vertices. Utilizing the proposed quality measure, we devise an algorithm that aims to detect high-quality communities of a given network based on two stages: finding a promising initial solution using a greedy method and then refining the solutions in a local search manner.

The algorithm's performance has been evaluated on various standard real-world networks and artificial graphs. The quality of the results has been reported and compared with those obtained by several state-of-the-art algorithms. As it turns out, the proposed approach is competitive with the other well-known techniques in the literature and significantly outperforms them.

4.1 Introduction

One of the fundamental topics in analyzing networks that has recently attracted attention is partitioning nodes into so-called communities consisting of highly interactive vertices. In this sense, a network can be modeled as a graph $G = (V, E)$ with the set of vertices V and edges E. Despite there is no universally agreed-upon definition of a community in the literature, this work takes one of the most common ones into consideration: a community is a subset of vertices $C \subseteq V$ with a high density of edges between nodes inside the subset and a low density of edges connecting this subset to the others. Accordingly, the community detection problem is the problem of finding a partitioning of the nodes into communities.

Such partitioning indeed enables us to survey many underlying properties of networks and has numerous applications in a wide range of areas including WEB [10], social media/network analysis [102], [175], [4], biological networks [75], signal processing [16], image segmentation [116], pattern recognition [71], data clustering [123], knowledge-based systems [161], recommender systems [5], and etc.

There are several variants of the problem depending on whether: the network is weighted or unweighted, the network is directed or indirected, the desired number of communities is given or not, and communities have overlap or they are completely separated; e.g., see [80], [41], [32], [155], [55], [133]. For each of the problem types mentioned, a large number of algorithms have been

Table 4.1: Several connectivity-based quality functions for measuring the quality of a community C. $|E_C^{out}|$, $|E_C^{in}|$, $|C|$ and $deg(i)$ respectively represent the number of external connections of C, number of internal connections of C, the number of vertices belonging to C, and the degree of node i.

	Function	Definition
IC	Internal density [142]	$\frac{\lvert E_C^{in}\rvert}{\lvert C\rvert(\lvert C\rvert-1)/2}$
	Edge inside [142]	$\lvert E_C^{in}\rvert$
	Average degree [142]	$\frac{2\lvert E_C^{in}\rvert}{\lvert C\rvert}$
EC	Expansion [142]	$\lvert E_C^{out}\rvert$
	Cut Retio [177]	$\frac{\lvert E_C^{out}\rvert}{\lvert C\rvert(\lvert V\rvert-\lvert C\rvert)}$
I/EC	Conductance [156]	$\frac{\lvert E_C^{out}\rvert}{2\lvert E_C^{in}\rvert+\lvert E_C^{out}\rvert}$
	Normalized Cut [156]	$\frac{\lvert E_C^{out}\rvert}{2\lvert E_C^{in}\rvert+\lvert E_C^{out}\rvert}+\frac{\lvert E_C^{out}\rvert}{2(\lvert E\rvert-\lvert E_C^{in}\rvert)+\lvert E_C^{out}\rvert}$
	Out Degree Fraction [67]	$\max_{i\in C}\frac{\lvert(i,j)\in E : j\notin C\rvert}{deg(i)}$
	Flake-ODF [67]	$\frac{\lvert i\in C,\lvert(i,j)\in E:j\in C\rvert<deg(i)/2\rvert}{\lvert C\rvert}$

presented, and worth noting that although there is a range of non-optimization-based algorithms such as the Label Propagation algorithm [143], the majority of community detection approaches follow a simple optimization strategy: they first define or pick a specific metric, called quality function, for determining the goodness of communities. Then they come up with an optimization algorithm to detect communities with respect to optimizing the quality function.

The rest of this section provides a brief overview regarding some of the most well-known quality functions, state-of-the-art community discovery algorithms, their strengths, and shortcomings. Then, it concisely states the proposed algorithm's highlights.

4.1.1 Related work

Over recent years, numerous approaches have been developed for community discovery, such as [17], [22], [35], [117], etc. Therefore, due to the wide range of works and limited space, we only report the approaches from the literature that are part of our consideration. For detailed reviews, we refer the interested reader to [68] and [70].

There are plenty of quality measures in the literature to qualify the goodness and accuracy of partitioning. At a glance, according to Chakraborty et al. [38], community quality functions can be classified into four broad categories: internal-connectivity-based metrics (IC), external-connectivity-based metrics (EC), metrics based on both internal and external connectivity (I/EC), and metrics based on networks' topology. Table 6.1 lists some of the well-known connectivity-based quality functions. For instance, the function *Average degree* obtains a community's quality based on two factors: the number of edges with both end nodes inside the community as well as the number of nodes within the community. From this quality function's perspective, a community that consists of more nodes and edges is more qualified.

On the other hand, among the metrics based on the topology of the network, *Modularity*, introduced by Newman [134], is one of the most used and best-known quality functions. For a community C, Modularity is defined as the number of edges within C minus the expected number of such edges. In addition to all this, there are plenty of other community score functions such as FOMD, Cohesiveness, Clustering coefficient, Spart, Permanence, etc. We refer the reader to [38], [51] for a comprehensive survey about quality functions for community analysis.

Given a quality function, it is natural to think of proposing a systematic technique for exploring communities by optimizing the quality measure. In this regard, several optimization techniques, such as integer programming approaches, can follow the global optimization procedure [3]. Nevertheless, while these methods can obtain relatively accurate results, they usually fail to encounter extensive networks since optimizing the quality functions often falls into the category of difficult computational problems [151]. Accordingly, while dealing with immense networks, which we usually face in the real world, one acceptable approach would be to rely on heuristic techniques from the very beginning [40]. Indeed, many heuristic (optimization-based) algorithms tackle the problem through one or a combination of the following approaches:

- Bottom-up: starting with singleton communities and merging communities.
- Top-down: starting with considering one big community and splitting clusters.
- Local search: starting with an initial partition and migrating nodes.

For instance, Blondel et al. [28] proposed a bottom-up heuristic greedy Modularity maximization algorithm named *Louvain*. This algorithm initially tries to discover small communities by locally optimizing modularity on all nodes. Afterward, it groups each obtained small community into one node and restarts the procedure. *Infomap* algorithm [146] is another bottom-up algorithm and uses the same optimization procedure as in the Louvain algorithm but tries to optimize a different quality function, an entropy-based function called *Map Equation*. Infomap concentrates on the information needed to condense the motion of the random walker. More precisely, the basic idea behind this algorithm is to use community partitions of the graph as a Huffman code [30] that compresses the information about a random walker exploring a network. This approach involves associating codes of different lengths to each node, depending on a random path's ergodic node visit frequencies. Huffman coding is an efficient way to transmit periodically or discontinuously detach nodes that comprise the random path we assume to encode. *Fast Greedy* algorithm, which was developed by Newman [133], uses a greedy hierarchical bottom-up approach to directly optimizing a Modularity score. *Girvan-Newman (GN)* algorithm is, on the other hand, a top-down hierarchical method based on betweenness metrics [80]. This is while, the spectral graph partitioning approach discovers communities of a network with respect to solving relaxed versions of the *Normalized cut* minimization problem. This approach uses information from the eigenvalues (spectrum) of special matrices associated with a graph and uses the corresponding eigenvectors to detect communities; see [174], [74]. On the local search side, Daniel Aloise et al. [8] has proposed a technique for maximizing the modularity. They applied the idea of the Variable Neighborhood Search (VNS), a metaheuristic framework for building heuristics that aims to solve combinatorial and global optimization problems, to the Modularity maximization problem.

To summarize, for giving a better vision to some recent progress in community detection algorithms, we refer the reader to [40], [110], [91], [171].

The crucial point is that the growing trend in developing various quality functions can indicate the existence of deficiency in all of these functions' applicability. For example, Modularity, as one of the mostly-used measures in this area, suffers from several known drawbacks such as the resolution limit (i.e., failure to detect communities smaller than a specific scale) [69], and the high degeneracy (i.e., leading to various partitions with equally high Modularity score) [81]. As a result, we see a large number of Modularity's extended versions in the literature, provided to cover the limitations. Two of the most successful versions are *Motif Modularity*, introduced by Arenas et al. [11] and *Max-Min Modularity*, proposed by Chen et al. [41]. Authors of the paper [41] figured out that Modularity's downside comes from the fact that it just relies on the existing links, and so massive information is lost due to the lack of noticing the missing links. Furthermore, they observed that not all the missing links are a sign of disconnectivity or lack of relationship, but rather, absent edges can be grouped into two categories: *related* and *unrelated* pair of nodes. They supposed that a network's specialist could recognize the unrelated

absent edges. Therefore, one can think of simultaneously maximizing the Modularity over the main graph G and minimizing the Modularity over a so-called complementary graph consisting of unrelated absent edges of G. While Max-Min Modularity managed to perfectly address the Modularity'drawback by taking non-adjacent unrelated vertices into consideration, it still lacks a systematic way to detect such pairs of nodes. Ferdowsi and Khanteymoori in 2021 [64] were succeeded in proposing an accurate analytical technique for this problem. Nevertheless, despite their method's very promising results, it still has some troubles while dealing with very massive networks due to the computation complexity.

On the other hand, one disadvantage of many connectivity-based quality functions is that they usually only focus on one or two basic topological factors and do not simultaneously consider other attributes or properties of the network. Moreover, possible combinations of such properties have not been built on a smart and logical platform, resulting in facing non-accurate results. Furthermore, another argument is that many optimization algorithms, regardless of the quality function used, have deficiencies in their nature. For instance, many approaches start with relatively inaccurate initial communities, making it difficult, impossible, or even very time-consuming to reach the final high-quality communities [178], [153].

In fact, in this paper, we establish a wise connection between some substantial topological characteristics of graphs to determine the most authoritative nodes that direct us to propose a novel prominent quality function. Besides, we also devise a greedy local search algorithm that does not follow a simple strategy for optimizing the quality function but rather also tries to modify the authoritative nodes as well as the quality function evolutionarily. This new consideration leads to the interactive quality function.

4.1.2 Main contribution

This paper aims to discover a set of non-overlapping communities $\mathbb{C} = \{C_1, \ldots, C_k\}$ for a given unweighted undirected network, where the number of communities, k, will be automatically estimated by the method. To do so, we first modify one of the fundamental connectivity-based quality functions. In this fashion, not only do we take the network connectivity into account, but we also investigate the role that networks' influential nodes can play in establishing communities. More precisely, by carefully simultaneously utilizing the three crucial concepts in graphs, *connectivity level*, *degree*, and *between-nodes distance*, we propose *i*) a systematic technique to determine the most influential nodes in the graph, *ii*) an encouraging quality function, and *iii*) a mechanism to obtain a logical between-communities distance. Furthermore, with all these properties combined, we also propose an efficient optimization-based algorithm to detect high-quality communities with respect to optimizing the modified quality measure. The algorithm greedily establishes high-quality initial communities based on recognizing the influential vertices that are constrained to be placed at a systematically predetermined distance from each other. It then iteratively updates the obtained communities to reach final communities with the best possible quality value in a local search manner. The update procedure is based on a local search technique that seeks the best way to achieve better communities by merging (small) communities, splitting (large) communities, and substituting nodes between neighbor communities. Considerably notable regarding the updating stage is that it is not isolated from the first stage. Instead, it operates the first stage dynamically to break down the large low-quality communities perfectly.

It is worth pointing out that, at a glance, it can be inferred that the proposed work has a relative overlap with the methods using connectivity-based quality measures and also with those in which a local search optimization technique is employed. However, considerably important is that to the best of our knowledge, this study is fundamentally distinguishable from all pre-existing methodologies, both from the perspective of the quality measure and the local search optimization technique. Therefore, although this issue makes this research has significant novelty, it also makes it hard to draw a meaningful relationship between this approach and other community detection

methods. We thereby determined to focus more on the approaches with the most similarities to our method and examine the proposed technique against some optimization methods that resemble our algorithm more or less. Furthermore, we picked the Modularity and one of its successful extensions as the most potent rival to our algorithm on the quality measure side.

In this regard, for the evaluation part, we apply our proposed algorithm and also five state-of-the-art heuristic community detection algorithms on several well-known real-world networks as well a famous category of randomly generated graphs whose optimal community structures are available. We then compare the obtained communities with each other. The results demonstrate the superiority of our proposed algorithm over the other under-study algorithms.

The rest of the paper is organized as follows. First, we introduce the community quality function and the community mining algorithm in Section 2, and in Section 3, we report the experimental results.

4.2 TSCDA: Two-Stage Community Detection Algorithm

In what follows, we first set up some notations and preliminaries, which we use throughout the paper. After that, we recommend our quality function and then, we propose our community detection algorithm.

4.2.1 Background and notations

Let $G = (V, E)$ be a graph with $n = |V|$ nodes, $m = |E|$ edges, and the adjacency matrix A. Let $\mathbb{C} = \{C_1, \ldots, C_k\}$ denotes a set of k disjoint subsets of V. We refer to each $C_l \in \mathbb{C}$ as a community of G. The followings are more notations:

- For each $i \in V$, let $deg(i)$ shows the degree of i, i.e., $deg(i) = \sum_{j=1}^{n} A_{i,j}$.
- Let $N_G(i) = \{j \in V \mid (i, j) \in E\}$ denotes the set of neighbors of the i.
- For each $C \in \mathbb{C}$, we define: $E_C^{in} = \{(u, v) \in E \mid u, v \in C\}$ and $E_C^{out} = \{(u, v) \in E \mid u \in C, v \notin C\}$ to respectively be the set of *inner* and *outer edges* of C. To be more specific, E_C^{in} indicates the number of edges whose both end nodes belong to C, while E_C^{out} expresses the number of edges whose just one end node falls into C.
- A vertex $i \in V$ is said to be $\mathbb{C}-$*homeless* if it does not belong to any $C \in \mathbb{C}$. The set of all $\mathbb{C}-$*homeless* vertices is denoted by $H_{\mathbb{C}}$. Note that $H_{\mathbb{C}} = V$ when $\mathbb{C} = \emptyset$.
- For every vertex $i \in V$, let $C(i) \in \mathbb{C}$ be the community that contains i.
- For every $i \in V$, we let $deg_{\mathbb{C}}^{(in)}(i) = \Sigma_{j \in C(i)} A_{i,j}$ and $deg_{\mathbb{C}}^{(out)}(i) = \Sigma_{j \in H_{\mathbb{C}}} A_{i,j}$ to be the $\mathbb{C}-$*inner degree* and $\mathbb{C}-$*outer degree* of i, respectively. Note that $deg_{\mathbb{C}}^{(out)}(i)$ is the number of edges connecting node i to the homeless vertices. In other words, the inner degree of a node i is the number of vertices adjacent to i that belong to the same community, which contains i. This is while the outer degree of i is the number of vertices adjacent to i that do not belong to any communities.
- Distance between any two vertices $i, j \in V$ is defined by the length of the shortest path connecting these two nodes in G, and is denoted by $dist(i, j)$. The diameter of G is the longest shortest path length in the graph and denoted by $diameter(G)$.
- Two communities C and C' are said to be *neighbors* if there exists an edge $(i, j) \in E$ connecting $i \in C$ to $j \in C'$. In this case, we refer to both i and j as *boundary vertices* of C respectively C'. Let $BV(C)$ denotes the set of all boundary vertices of C.

4.2.2 The quality function

Without any doubt, the degree of a vertex can usually be thought of as one of the authority factors of the vertex for controlling/affecting/biasing its surrounding nodes. In fact, high degree vertices can impose many features on and pass various messages to their nearby vertices. They can even influence and bias their surrounding nodes' attributes and behaviors [86], [179], [106], [140], [34]. To be more specific, a large number of (complex and social) networks usually contain a proportionally small number of influential high-degree vertices e.g., influencers in a social media or essential and top-selling commodities in a co-purchasing network, which are surrounded and followed by a vast number of lower degree vertices; see [138], [58], [19], [43]. In such networks, each vertex is either connected directly to or is usually only a few links away from a high-degree node. Consequently, high-quality communities are likely expected to be formed around high-degree vertices. Indeed, we have observed that high-performance communities typically contain one or a few numbers of high-degree vertices, along with a large number of their nearby surrounded lower-degree vertices.

Based on this, we came to the following definition of an *influential node* of a community:

Definition 1 (Influential node) *Given a community $C \in \mathbb{C}$, a vertex $i \in C$ with the maximum $\mathbb{C}-$inner degree is said to be the influential node of C. Let us denote this node by $m(C)$.*

Note that in our consideration, every community $C \in \mathbb{C}$ is allowed to have only one influential node; if there is more than one vertex in C with the same maximum inner degree, one of them will be selected arbitrarily. Based on the above definition, we also define the following notations.

Definition 2 *Distance between a vertex i and a community C is defined by the length of the shortest path from i to the influential node of C, i.e., $dist(i, m(C))$. In addition, $dist(m(C), m(C'))$ represents the distance between two communities C and C'.*

Definition 3 (Radius of a community) *We let $r(C) = \max_{i \in C}\{dist(i, m(C)\}$ to denote the* radius *of a community C. In other words, $r(C)$ is the maximum distance, in terms of the length of the shortest path, that C's influential node has from other vertices of C.*

Interestingly, it turned out that an influential node's effect on other vertices is correlated to the distance between them. Important insight could be gained by [9], [64]. It means that, for example, a very famous person in a social network can have a more significant impact on individuals who follow him/her directly (distance= 1) rather than those who know him/her through several intermediaries. (distance $>$ 1) (see [186]). Therefore, we conjectured that a community in which the overall distance between the influential node and the other vertices is small (i.e., its radius r being small) is likely to be a more interactive and homogeneous community.

Accordingly, we recommend the following measure for obtaining the quality of a community $C \in \mathbb{C}$:

$$q(C) = \frac{|E_C^{out}| + r(C)}{|E_C^{in}|}. \tag{4.1}$$

We call $q(C)$ the *quality value* of the community C. Besides, let $q_G^{\mathbb{C}}$ indicates the quality of the partition $\mathbb{C}$ of G and be defined as follows:

$$q_G^{\mathbb{C}} = \sum_{\forall C \in \mathbb{C}} q(C). \tag{4.2}$$

Note that the lower the value of $q(C)$, the higher the quality of the community C is. Basically, the proposed community quality measure determines the quality of a community $C \in \mathbb{C}$ by minimizing the sum of the following two terms: $\frac{|E_C^{out}|}{|E_C^{in}|}$ and $\frac{r(C)}{|E_C^{in}|}$. Recall the definition introduced

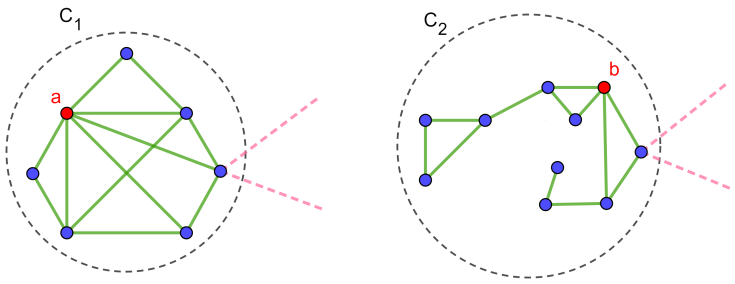


Figure 4.1: Two communities C_1 and C_2 with influential vertices a and b, respectively. C_2 is a worse community since it has more disconnected node pairs than C_1. $q_1 = \frac{3}{12}$ and $q_2 = \frac{5}{12}$ demonstrate the superiority of C_1 over C_2.

in Section 4.1 for a community. Based on that, it is natural to assume that a high-quality community has a large number of outer edges and a small number of inner edges simultaneously. Therefore, obtaining a small (close to zero) value for $\frac{|E_C^{out}|}{|E_C^{in}|}$ can indeed lead C to become a compact and isolated community. The latter term, on the other hand, indicates that a community with a large number of inner edges and a small radius tends to be more coherent and dense.

For a better understanding, we draw your attention to the following case. Assume that C_1 and C_2, shown in Figure 4.1, are two communities of a network with influential vertices a and b, respectively. Consider, for example, the three quality functions $q'_C = \frac{|E_C^{out}|}{|E_C^{in}|}$, conductance $\frac{|E_C^{out}|}{2|E_C^{in}|+|E_C^{out}|}$, and Modularity. Be noted that the score of each of these quality functions for C_1 and C_2 are equal, and therefore based on their viewpoint, C_1 and C_2 have the same quality. This is while it is apparent that C_2 is a much worse community due to its more absence links than C_1. Nevertheless, if we consider our proposed quality function, we have $q_1 = \frac{3}{12}$ and $q_2 = \frac{5}{12}$, which truly shows that C_1 is a better community.

4.2.3 The algorithm

In this section, we are going to depict our Two-Stage Community Detection Algorithm (TSCDA). For a given simple graph $G = (V, E)$, the algorithm tries to find a set of communities $\mathbb{C} = \{C_1, \ldots, C_k\}$ for which $q_G^{\mathbb{C}}$ is (approximately) minimized. TSCDA consists of two phases: *Creating Initial Communities* and *Updating Communities*, which we describe as follows:

Stage 1: Creating Initial Communities (CIC): Algorithm 1 elaborates on the PseudoCode of this stage. This phase aims at establishing a set of initial communities by detecting promising influential vertices. The basic idea is that each discovered influential node can set up its own community by absorbing the nearby vertices. Consequently, the quality of initial communities is strongly likely to rely on the set of initial discovered influential nodes. This issue emphasizes the need of a systematic way of detecting initial influential nodes.

Recall the definition of the influential nodes (Definition 1). It might be counterintuitive that only relying on the degree of vertices would be sufficient for obtaining promising initial communities. While the considered definition for the influential node and its usage toward minimizing the proposed quality function (Equation 6.1) is perfectly capable of strengthening internal structures of communities, it is unlikely to have enough potential for separating communities. More specifically, it can be easily turned out that as crucial as the influential nodes' degree and the proximity of each communities' vertices to these nodes, so is the separation of the communities. In this regard, we conjectured that an additional provision for setting initial influential nodes could help establish better distinct communities. An important intuition could be gained by noting that since the ultimate goal is to find communities as isolated from each other as possible, starting

with initial communities that are far enough apart can likely lead to a significant step in the right direction and propel to finding final accurate communities. This is in accordance with the fact that discovering high-quality communities relies considerably on recognizing initial influential nodes, which are at a proper distance from each other. As a result, we introduce a parameter α and employ the notion of a $\alpha-$*far vertex* (explained below) to supervise the distance between influential vertices.

Definition 4 ($\alpha-$far vertex) *Let $S \subset V$. For any $\alpha \in \{2, 3, \ldots, diameter(G)\}$, vertex $i \in V - S$ is said to be α-far from S if $\min_{s \in S}\{dist(i, s)\} \geq \alpha$. Note that every vertex $i \in V$ is α-far from S when $S = \emptyset$.*

Utilizing the interpretation of an $\alpha-$far vertex, for a given parameter $\alpha \in \{2, 3, \ldots, diameter(G)\}$, the CIC stage tries to guess a corresponding set of initial influential nodes $M(\alpha)$. Then, it determines a set $\mathbb{C}(\alpha)$ consisting of communities, each of which is associated with an influential vertex. The following steps describes the procedure.

- Given a $\alpha \in \{2, 3, \ldots, diameter(G)\}$.
- Initialize $M(\alpha) = \mathbb{C}(\alpha) = \emptyset$.
- While there is an $\alpha-$far vertex from $M(\alpha)$, repeat the following operation:
  - Among all $\alpha-$far vertices from $M(\alpha)$, select a $\mathbb{C}(\alpha)-$homeless node i with a maximum $\mathbb{C}(\alpha)-$outer degree [1] as a new influential vertex and assign it into both of the sets $M(\alpha)$ and $\mathbb{C}(\alpha)$.
- Assign every $\mathbb{C}(\alpha)-$homeless node to its closest community.
- Update the communities' influential nodes as well as the set $M(\alpha)$, based on Definition 1, if necessary.

Now, be advised that by repeating the above procedure for every $\alpha \in \{2, 3, \ldots, diameter(G)\}$, exactly $diameter(G) - 1$ sets of communities, $\mathbb{C}(2)$, ..., $\mathbb{C}(diameter(G))$, with the corresponding quality values $q_G^{\mathbb{C}(2)}$, ..., $q_G^{\mathbb{C}(diameter(G))}$ will be obtained. Let the best-obtained set $\mathbb{C}(\alpha)$, the one with the least $q_G^{\mathbb{C}(\alpha)}$, be referred to as the set of initial communities. More specifically, this set is exactly what the CIC stage (Algorithm 1) returns. Please note that considerably attractive is the repeatable procedure that enables us to find an appropriate between-communities distance with a good approximation.

It is also worth noting that obtaining $\mathbb{C}(\alpha)$ for different values of α does not unacceptably increase the execution time since almost many (social and complex) networks are sparse and have a relatively small diameter compared to their number of vertices; See [154], [10], [43], [19], [135].

Stage 2: Updating Communities (UP): Algorithm 2 provides the PseudoCode of this stage. This phase basically updates the initial communities by using a local search technique based on iteratively moving to neighbor solutions. In a more precise explanation, it iteratively selects a community in such a way that the probability of choosing a community C with a worse quality value is more than a community with a better quality value. After that, it greedily improves the communities based on one of the three functions *Merge*, *Split*, and *Swap* at a time. These functions operate as follows:

- *Merge* function checks if merging C into any of its neighbor communities can improve the quality value.

[1] $\mathbb{C}(\alpha)-$outer degree because the community is not formed yet, so a high degree candidate node must be considered.

Algorithm 1 Creating Initial Communities (CIC)

Require: Graph G
Ensure: Set of initial communities $\mathbb{C}$ and the corresponding set of influential nodes
for $\alpha = 2$ **to** $diameter(G)$ **do**
  $\mathbb{C}(\alpha) \leftarrow \emptyset$ ▷ set of communities
  $M(\alpha) \leftarrow \emptyset$ ▷ set of influential vertices
  $k \leftarrow 0$
  $H_{\mathbb{C}(\alpha)} \leftarrow V$
  condition $\leftarrow$ TRUE
  while *condition* = TRUE **do**
    $F \leftarrow \{i \in V \mid i \text{ is } \alpha\text{-far from } M(\alpha)\}$
    $I \leftarrow \{i \in V \mid i \in \text{argmax}(deg^{(out)}_{\mathbb{C}(\alpha)}(j) : j \in F)\}$
    if $I \neq \emptyset$ **then**
      $k \leftarrow k + 1$
      Choose $i \in I$ randomly
      $C \leftarrow \{i\}$
      $m(C) \leftarrow i$
      $M(\alpha) \leftarrow M(\alpha) \cup m(C)$
      $\mathbb{C}(\alpha) \leftarrow \mathbb{C}(\alpha) \cup C$
      Update $H_{\mathbb{C}(\alpha)}$
    else
      condition $\leftarrow$ FALSE
  for every $i \in H_{\mathbb{C}(\alpha)}$ **do**
    $l \leftarrow \text{argmin}\left(dist(i, m(C_u)) : u \in \{1, 2, \ldots, k\}\right)$
    $C_l \leftarrow C_l \cup i$
    Update $H_{\mathbb{C}(\alpha)}$ and $\mathbb{C}(\alpha)$
    Update $m(C_l)$ if necessary
  $q(\alpha) \leftarrow q_G^{\mathbb{C}(\alpha)}$
  remember the quality value $q(\alpha)$, the corresponding set of communities $\mathbb{C}(\alpha)$, and also $M(\alpha)$
return a set of communities $\mathbb{C}$ and the corresponding set of influential nodes M, associated to the minimum obtained quality value $q(\alpha)$, $\forall \alpha \in \{2, \ldots, diameter(G)\}$

- *Split* function checks if C can be divided into more than one community, given that the quality value of the obtained division is better than $q(C)$, the quality value of C. To do so, the CIC stage (Algorithm 1) will be applied to the induced sub-graph G' of G formed from the subset C of V.
- *Swap* function applies the best possible swaps between nodes in C, and nodes belong to C's neighbor communities.

After each iteration, the best function that leads to obtaining the most significant improvement in the quality value will be selected. The communities and the influential vertices will be updated afterward. The procedure terminates when no improving local search move exists.

It is apparent that the better the obtained initial communities are, the less number of reforms is needed to be done by the UC stage. In the next section, we will discuss this argument in detail. More specifically, we will show how these two stages, individually respectively together, lead to obtaining high-quality communities.

As a final note regarding TSCDA, please recall that combining Definition 1 and Definition 4 led to a systematic intelligence procedure for recognizing the initial influential nodes in the CIC stage. The method, which was based on monitoring the distance between influential nodes, enabled us to find the best possible between-communities distance. Considerably attractive is the Split function in the UC stage, which triggers this procedure in each iteration since it calls the CIC procedure over an induced subgraph of G, resulting in that this methodical design is dynamically distributed throughout the algorithm.

Algorithm 2 Updating Communities (UC)

Require: Graph G, set of initial communities $\mathbb{C}$, set of influential vertices M
Ensure: Set of final communities $\mathbb{C}$
$q \leftarrow q_G^{\mathbb{C}}$
$q_{\text{temp}} \leftarrow 0$
while $q \leq (1+\epsilon)\, q_{\text{temp}}$ **do** ▷ Small constant ϵ guarantees polynomial running time; see [13], [85].
    $q_{\text{temp}} \leftarrow q$
    select a community $C \in \mathbb{C}$ with probability $\frac{q(C)}{q_G^{\mathbb{C}}}$
    $(\mathbb{C}, M) \leftarrow \text{BestMovement}(C, \mathbb{C}, M)$
    $q \leftarrow q_G^{\mathbb{C}}$
return $\mathbb{C}$

Function declarations:
function BestMovement$(C, \mathbb{C}, M)$
    $(\mathbb{C}_{\text{temp\_M}}, M_{\text{temp\_M}}) \leftarrow \text{Merge}(C, \mathbb{C}, M)$
    $(\mathbb{C}_{\text{temp\_S}}, M_{\text{temp\_S}}) \leftarrow \text{Split}(C, \mathbb{C}, M)$
    $(\mathbb{C}_{\text{temp\_Sw}}, M_{\text{temp\_Sw}}) \leftarrow \text{Swap}(C, \mathbb{C}, M)$
    retrieve the set of communities $\mathbb{C}$ and its set of influential vertices M, corresponding to the minimum obtained quality value
    return $(\mathbb{C}, M)$
function Merge$(C, \mathbb{C}, M)$
    for every $C' \in \mathbb{C} \setminus C$ such that C' is a neighbor community of C **do**
        merge C and C'
        update and remember $\mathbb{C}$ and also the set of influential vertices
    retrieve the set of communities $\mathbb{C}$ and its set of influential vertices M, corresponding to the minimum obtained quality value
    return $(\mathbb{C}, M)$
function Split$(C, \mathbb{C}, M)$
    $G' \leftarrow$ induced sub-graph of G formed by the subset C of vertices
    $(\mathbb{C}', M') \leftarrow \text{CIC}(G')$
    if $|\mathbb{C}'| > |\mathbb{C}|$ **and** $q_{G'}^{\mathbb{C}} \leq q(C)$ **then**
        $\mathbb{C} \leftarrow (\mathbb{C} \setminus C) \cup \mathbb{C}'$
        update M
    return $(\mathbb{C}, M)$
function Swap$(C, \mathbb{C}, M)$
    for every community $C' \in \mathbb{C}$ that is neighbor of C **do**
        for every pair of nodes $(i, j) \in BV(C, C')$ **do**
            swap the communities of i and j if it leads to a decrease in the quality value
    return $(\mathbb{C}, M)$

4.3 Experiments

Aside from optimizing the quality function considered, the ultimate target of a community detection algorithm is to detect high-quality communities, which means finding communities that are as similar as possible to the optimal community structures, of course, if such structures are available. To be more precise, according to [38], [160], obtaining communities that leading to a near-optimal value of quality functions might bear no similarity to the optimal community structures. Thus even by discovering communities with an optimal quality function, other verification methods are needed to confirm the efficiency of the communities. In this part, we aim to evaluate our algorithm's performance against several most successful community detection algorithms. To make a consistent condition for the comparisons, we try to estimate the similarity between each algorithm's discovered communities and the optimal community structures. Therefore, in this work, we take networks into account whose optimal community structures (i.e., *ground truth*) are available in the literature. We consider six well-known real-world datasets as well as a commonly-used class of synthetic graphs, and to determine the

Table 4.2: Real-world networks

Network	n	m	number of communities	diam
Karate club [184]	34	78	2	5
Dolphin network [120]	62	159	2	8
American college football [80]	105	613	12	4
Email-Eu-core [113]	986	16064	42	7
Amazon network [Mkhitaryan et al.], [182]	334863	925872	49732	44
LiveJournal social network [157], [182]	3997962	34681189	287512	17

similarity between each algorithm's obtained communities and the ground truth, we use two different evaluation metrics (explained in Section 4.3.2).

All tests are conducted on a computer system with a processor Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz, 2712 Mhz, 2 Core(s), 4 Logical Processor(s), 8 GB of Rams, and Win10 OS. Moreover, algorithms are implemented by the C++ programming language.

4.3.1 Datasets

The examined real-world networks in this paper have various properties and characteristics and are listed in Table 6.2. In addition, as synthetic benchmarks, we use the *Lancichinetti Fortunato Radicchi* (LFR) artificial network [111] that highly resembles real-world social networks and has a prior known and built-in community structure as well. The generating process of such artificial graphs is based on several user-defined parameters and is fully explained in [111]. These networks consider power-law distributions for both the degree and the community size with exponents γ and β, respectively. In addition, the mixing parameter μ specifies that every vertex of the graph should share a fraction of $1-\mu$ of its links with the other nodes of its community and a fraction of μ with the other nodes of the network. μ can control the noise, and the more the value of μ, the worse community structures emerge (i.e., communities become less distinct). Therefore, the higher the value of μ, the more trouble a community detection algorithm faces regarding finding the optimal communities [60]. In this work, we use four different LFR graphs with four different mixing parameters μ ($\mu = 0.1$, 0.3, 0.6, and 0.9), each of which has 1000 nodes and 5 communities. All the other parameters are set to the default values explained in [111].

4.3.2 Evaluation metrics

As mentioned earlier, once the communities are discovered, the next task is most likely to estimate the detected structures. The evaluation becomes more reliable if the ground truth is available. It is natural to conjecture that if an algorithm can detect a community structure with a high resemblance to the ground truth, it is trustworthy enough to be used further for other networks for which the underlying optimal community structures might not be available. Consequently, various metrics, such as F-measure [12] and Rand index [92], have been developed to correspond between the detected communities and the ground truth. Nevertheless, each of them has some shortcomings. For instance, it turned out that F-measure does not reveal the mutual information among communities, while the Rand index suffers from having biases that might lead to low-quality results. As a result, most of the research works considerably rely on two robust performance metrics named *Adjusted Rand Index (ARI)* and the *Normalized Mutual Information (NMI)*. In the following, we introduce these two quantitative metrics for validating the output of a community detection algorithm. ARl, which can be determined easily, measures the extent to which the two partitions agree with one another by evaluating the relationship between pairs of nodes. However, NMI is more complicated to compute but can quantify the performance of different community structures with high computational complexity.

Suppose that for a given network G, $\mathbb{C} = \{C_1, \ldots, C_k\}$ and $\mathbb{C}' = \{C'_1, \ldots, C'_{k'}\}$ be respectively a set of communities obtained by an algorithm $\mathcal{A}$ and the ground truth.

Normalized Mutual Information (NMI) [50] is indeed a well-known clustering comparison metric, though it can be used to evaluate the similarity between two sets of communities. More precisely, the NMI value corresponding to algorithm $\mathcal{A}$ can be written as

$$NMI = \frac{-2\sum_{x=1}^{|\mathbb{C}|}\sum_{y=1}^{|\mathbb{C}'|}\frac{|C_x \cap C'_y|}{n} log(\frac{n|C_x \cap C'_y|)}{|C_x||C'_y|}}{\sum_{x=1}^{|\mathbb{C}|}\frac{C_x}{n} log(\frac{C_x}{n}) + \sum_{y=1}^{|\mathbb{C}'|}\frac{C'_y}{n} log(\frac{C'_y}{n})} \tag{4.3}$$

If the detected communities are identical to the ground truth, the NMI takes its maximum value one, while in the case where the two sets totally disagree, the NMI score is zero. Generally, the higher the value of NMI, the better community structures have been found.

Adjusted rand index (ARI) [183], corresponding to the algorithm $\mathcal{A}$, measure the similarity between $\mathbb{C}$ and $\mathbb{C}'$ as follows

$$ARI = \frac{2(a \times d - b \times c)}{(a+b) \times (b+d) \times (a+c) \times (c+d)} \tag{4.4}$$

where a, b, c and d are respectively the number of vertex pairs that are in the same community in both $\mathbb{C}'$ and $\mathbb{C}$, in the same community in $\mathbb{C}'$ but not in $\mathbb{C}$, in the same community in $\mathbb{C}$ but not in $\mathbb{C}'$, and in different communities in both $\mathbb{C}'$ and $\mathbb{C}$. Like the NMI measure, the value of ARI varies between 0 and 1, and the higher its value is, the more similarity is between the communities obtained by algorithm $\mathcal{A}$ and the grand truth of G.

4.3.3 Comparison algorithms

Recall that one of the most widely used quality functions for community detection is Modularity. The reliance of many of the community discovery methods on this quality function and their success in many social and biological networks led us to measure our algorithm's performance against four state-of-the-art Modularity maximization algorithms, the Distributed Louvain (*DLouvain*) [78], Variable Neighborhood Search (VNS) [8], Simulated Annealing (SA) [180], and FastGreedy [133] algorithms. We have also compared our algorithm with one of the very successful extension of Modularity maximization problem, Max-Min Modularity[2] [41]. Besides, we also consider two other high-efficient algorithms: Infomap [146], and the Improved Label Propagation [181]. These two algorithms are respectively based on (i) optimizing a well-known information-theoretical measure (description code length quality function), which returns high-quality communities for big networks with a relatively low level of noise, and (ii) introducing a new similarity measure and using the minimum distance and local centrality to establish initial communities. The algorithm then employs the label influence to update the community structures based on the new similarity measure.

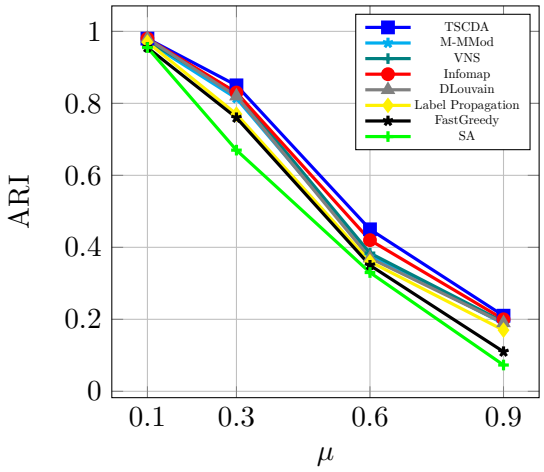
It's worth mentioning that all these algorithms are among the best high-efficient and powerful community detection algorithms for social and complex networks [160].

4.3.4 Experiment setup and results

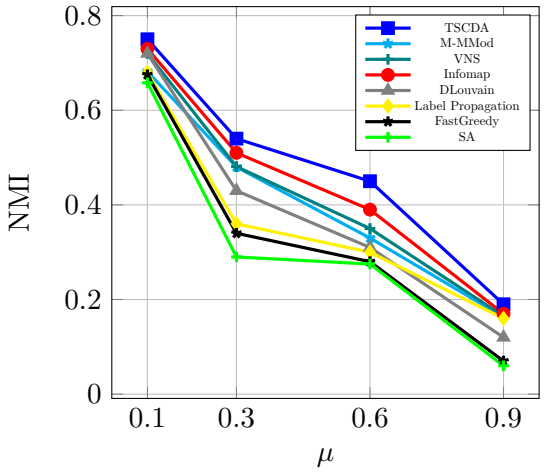
For arranging the comparisons over real-world networks, we compare the performance of all introduced algorithms by examining the average ARI and NMI ranks over all datasets. Figure

[2]Throughout the experiments regarding the Max-Min Modularity, two non-adjacent nodes, i, j are referred to be as related if there is an intermediate node k such that $(i,k) \in E$ and $(k,j) \in E$. Moreover, i and j are considered unrelated pairs if there is no such vertex k.

4.3 presents the results. It is apparent that TSCDA significantly outperforms other algorithms in the sense that the communities it discovered are much more similar to the ground truths than those obtained by the other methods. In addition, Figure 4.2 illustrates the ARI and NMI values obtained by applying TSCDA and other comparable algorithms to the LFR benchmarks explained in Section 4.3.1. As we can see, although the performance of all the algorithms decreases with increasing μ, TSCDA surpasses other techniques in all cases.



(a) ARI



(b) NMI

Figure 4.2: ARI (a) and NMI (b) performance rank of each algorithm applying to the LFR synthetic graphs, generated by considering four different mixing parameters μ.

Although the above experiments explain the high performance of TSCDA compared to the other algorithms we study, another substantial and crucial issue is to see how accurately each stage of TSCDA works. In other words, it needs to be traced that how authoritative the obtained initial communities are and how many reforms must be made during the second stage.

Turning our attention toward the first phase of TSCDA, there are several crucial arguments. The first and foremost is regarding the quality of the initial influential nodes and the initial communities that both considerably depend on the novel distance characteristic considered. Hence, it seems necessary to investigate how the applied distance factor helps to improve the

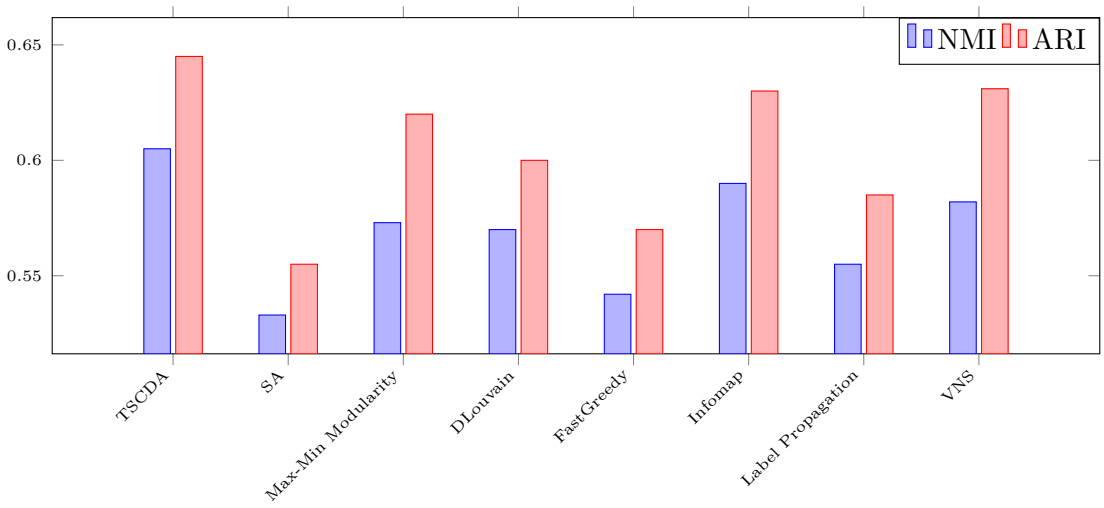
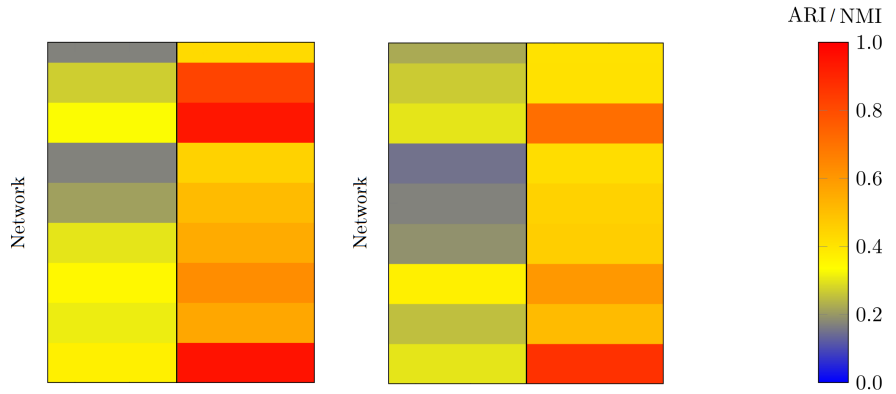


Figure 4.3: Average normalized ARI and NMI performance rank of different algorithms applying to the real-world networks, presented in Table 6.2. The average NMI and ARI values obtained by TSCDA are respectively equal to 0.605 and 0.645.

proposed method's performance. In other words, we need to verify the role of considering the concept of the $\alpha-$far vertex (Definition 4). For doing this, the first phase of the algorithm without considering this criterion was applied to the networks, i.e., the dependence on the distance factor was eliminated from the algorithm, resulting in obtaining influential nodes only by relying on the order of nodes' degree (Definition 1), not by recognizing the proper between-communities distance. Then the first phase was run over the networks. Afterward, we compared the results with the ones obtained by the main algorithm of the CIC stage (Algorithm 1). Fig. 4.4 (a), (b) reports the results in terms of two heatmap diagrams, respectively, for ARI and NMI performance metrics. The left column of each figure shows the corresponding performance measure's value for the case in which the distance factor is completely ignored, while the right column shows the results obtained by the CIC stage's main algorithm. Moreover, each row of the diagrams represents a network. The results show the significant increment in the value of ARI and MNI, when the distance measure is also applied to the TSCDA.

Furthermore, Table 6.3 provides a comprehensive comparison between the two stages of TSCDA. In addition to the ARI and NMI values,obtained by each stage, the following information can also be depicted by the table: the best values for α founded by the CIP stage, the number of iterations done by the UC stage, the number of communities identified by each of the stages, the quality value acquired by each stage, and finally, the amount of execution time elapsed by each stage. By noticing the ARI and NMI values corresponding to the CIC stage, one can infer that the discovered initial communities, in turn, are very high-quality communities that do not require many modifications done by the UP stage. However, as can be seen, the reforms made by the second phase have led to meaningful improvements in communities. In addition, considerably promising is that the number of final communities in 6 of the 10 networks is quite correctly obtained, and in the remaining 4 networks, this number found is very close to the actual number of communities.

We conclude this section by noting that the proposed algorithm outperforms the other understudy algorithms in the provided experiments. More specifically, the values of the ARI and NMI obtained by TSCDA is respectively, on average, five and eight percent higher than that of the other algorithms, and according to [160], this small amount of superiority on this scale can lead to significant refinement in the obtained community structures. Furthermore, results in Table



(a) ARI heatmap (b) NMI heatmap

Figure 4.4: Two heatmap diagrams show the performance of the discovered initial communities (CIC stage) in terms of (*a*) ARI and (*b*) NMI metrics. Left columns specify the results for when the distance factor is entirely ignored. The right columns show the results obtained by taking the distance factor into account. Be noted that each row represents a network.

Table 4.3: Comparison results of different stages of TSCDA and also the total time elapsed in the algorithm.

	First stage (CIC)						Second stage (UC)					
Network	α	ARI	NMI	q_G^c	number of communities	elapsed time (min)	ARI	NMI	q_G^c	number of iterations	number of communities	elapsed time (min)
Karate	2	0.95	0.87	0.93	2	0.08	1	1	0.705	1	2	0.02
Dolphin	5	0.56	0.51	2.4	12	0.095	0.6	0.53	1.8	3	2	0.065
Football college	3	0.63	0.6	1.69	5	0.196	0.71	0.66	0.827	3	12	0.054
Email	4	0.55	0.46	2.12	54	1.82	0.57	0.51	1.4	8	42	1.28
Amazon	15	0.51	0.448	1.17	42216	76	0.54	0.49	0.72	11	49146	67
Journal	12	0.45	0.419	1.09	258362	157	0.47	0.43	0.65	9	287371	137
LFR ($\mu = 0.1$)	13	0.94	0.71	0.97	5	0.44	0.98	0.75	0.65	6	5	0.67
LFR ($\mu = 0.3$)	11	0.818	0.41	1.2	9	1.1	0.85	0.54	0.89	15	5	1.6
LFR ($\mu = 0.6$)	6	0.427	0.407	2.21	4	1.3	0.45	0.45	1.7	19	4	2.08
LFR ($\mu = 0.9$)	2	0.19	0.07	4.9	7	1.29	0.21	0.19	3.06	25	6	2.85

6.3 clearly show the reliability of initial communities obtained by the CIC stage. It means that we can confidently trust the communities achieved by this stage in extraordinarily massive and dense networks, for which it takes a long time to reform the communities by the UC stage and even apply other community detection algorithms. Therefore, when the network is big enough and the computational time is crucial, while the resulting partitioning quality is not, we can only apply the CIC stage and safely skip the second stage. As a result, we were also succeeded in introducing an almost accurate (but fast) alternative technique to guess relatively high-quality communities in enormous networks.

As a final remark, we would also like to argue that despite our proposed algorithm's better performance than other comparable algorithms, we still see that in most cases, optimal communities are not obtained. In addition, the communities found even by the state-of-the-art algorithms in many networks are far from reality. We conjecture that one reason is due to the lack of dynamic knowledge such as emotions, personality, attitudes, or mood. Almost all the algorithms in this field somehow overlook the significant roles that factors like momentary emotions, mood, reaction, etc., might be having on joining or leaving a community. Important insight regarding this issue can be reached in [20], where the authors were able to show that almost any classification algorithms fail to correctly predict the musical preference of an individual by just

relying on basic features such as age, sex, and education. It turned out that instant emotion can highly change the preference dynamically. This is precisely the problem that current community detection algorithms also face. Therefore, it clearly shows that the upcoming algorithms have to be considerably improved even to consider the hidden and instantaneous features of the inputs.

4.4 Conclusion

This paper presented TSCDA, a community detection algorithm for discovering communities of a given graph. TSCDA used an optimization technique to find communities with respect to optimizing a modified community quality function. The modified function measures communities' quality based on considering both the compactness and the cohesiveness of communities. The proposed algorithm first establishes appropriate initial communities by recognizing proper influential vertices, which are at a fair distance from each other. Next, it uses a local search technique to achieve final communities with approximately minimum quality values. We propounded six real-world networks and also a category of randomly generated synthetic networks to evaluate TSCDA's performance against several well-known community detection algorithms. The experimental results show that the ARI and NMI values obtained by our algorithm were approximately, respectively %five and %eight better than the ARI and NMI values obtained by the other algorithms. This superiority in the value of these variables shows that the communities obtained by TSCDA are notably better than the communities obtained by the other algorithms.

Chapter 5

Toward an Optimal Solution to the Network Partitioning Problem

Published in: *18th Conference on Computer Science and Intelligence Systems (FedCSIS)*
Toward an Optimal Solution to the Network Partitioning Problem.
A. Ferdowsi & M.DehghanChenary.
10.15439/2023F2832

Abstract This paper delves into the realm of community detection in network science and graph theory with the overarching objective of unraveling the underlying structures between nodes within a network. In this pursuit, we put forth a novel and comprehensive approach to ascertain the optimal solution to maximizing the renowned community quality metric known as Max-Min Modularity. Through a series of experiments encompassing diverse case studies, we substantiate the efficacy and validity of our proposed approach, further bolstering its credibility.

5.1 Introduction and related work

In complex networks, nodes usually divide into several subsets sharing common characteristics and relationships, forming *communities*. The discovery and analysis of these structures hold paramount importance in computer science, particularly within the network domain and graph theory. Identifying cohesive and interconnected groups of nodes enables a deeper understanding of complex systems, facilitating insights into structural patterns, functional modules, and underlying relationships. The ability to detect network communities owes immense value and applications in networked systems, such as Social Network Analysis [102, 175], Biological Networks [14], Cosmological Networks [109], WEB analysis [10], Distributed Computing [93], Signal Processing [166], and Data Clustering [123]. It enables us to uncover hierarchical structures, predict missing links, and enhance network resilience.

More concretely, a network can be represented as a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. A community within the network can be seen as a subset of vertices $C \subseteq V$, characterized by a dense connection of edges among the nodes within the subset and a sparse connection of edges with other subsets; see Fig. 5.1. In this regard, the community detection problem can be defined as partitioning V into a set of communities $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$ that often entails optimizing a specific *quality measure* that quantifies the excellence of a community. A wide array of quality measures has been proposed in the literature, encompassing both *connectivity-based* and *topology-based* metrics [38].

For example, in [63], a dynamic connectivity-based metric is introduced to assess the quality of a community C by computing the ratio between the sum of the radius of C and the number of edges exiting C, divided by the number of edges with both endpoints belonging to C. The radius, a measure showcasing the size and compactness of the function, plays a crucial role in

Figure 5.1: Illustration of a network and its communities as subsets of vertices with densely connected nodes within each subset and sparser connections to nodes outside the batch.

this evaluation. The authors of [63] also devised a two-stage heuristic algorithm to identify high-quality communities by minimizing the proposed metric. The initial stage of the algorithm, which is particularly crucial for our research, intelligently identifies an initial set of remarkably high-quality communities. This will be followed by a revising phase aimed at refining and enhancing the quality of the communities. The contributions in [63] properly highlight the significance of connectivity-based metrics in assessing community quality.

On the other hand, topological metrics have also gained considerable attention in the field of community detection. Notably, *Modularity*, introduced by Newman [134], stands out as one of the most widely recognized and extensively utilized measures in this regard. For a network G containing n vertices and m edges, the Modularity (Q) of a given partitioning $\mathbf{C}$ is mathematically defined as follows:

$$Q(\mathbf{C}) = \frac{1}{2m} \sum_{i,j \in V} [a_{ij} - \frac{d_i d_j}{2m}] \sigma(i,j) \tag{5.1}$$

where $A = (a_{ij})$ is the adjacency matrix of G with a_{ij} sets to one if an edge exists between node i and node j, and zero otherwise. d_i represents the degree of node i and is defined as the sum of all entries in the i-th row of the adjacency matrix. Moreover, $\sigma(i,j)$ is one if i and j are in the same community and zero otherwise.

Simply put, Modularity quantifies the number of edges within a community minus the expected number of such edges leading to the fact that communities with higher Modularity values have better quality. Therefore, maximizing Modularity results in identifying high-quality communities within a network.

Nevertheless, despite the widespread use of Modularity, it has been known to have certain limitations (see [59, 38] for more details). Notably, Modularity only takes into account the existing edges of the network, meaning it solely evaluates the goodness of a community based on its fit with the observed edges, while it fails to consider disconnected nodes (absent edges) within the same community. This is indeed a drawback since the disconnection of nodes does not inherently imply an absence of underlying relations between them. To overcome this limitation, an extension of Modularity called *Max-Min Modularity* [41] has been developed, which improves the accuracy of the measure by penalizing Modularity when disconnected nodes are present in the same community. In Max-Min Modularity, an additional zero-one relation matrix $U = (u_{ij})$ is introduced, which defines the relationship between pairs of disconnected nodes in the network. The value of u_{ij} is one if disconnected nodes i and j are *related* and zero otherwise. This extension

acknowledges the significance of indirect connections between disconnected nodes by penalizing the Modularity measure only when unrelated nodes coexist within a community. In a more abstract sense, consider a complemented graph $G' = (V, E')$, where E' contains an edge between every pair of disconnected nodes in G that are unrelated. In other words, an edge exists between nodes i and j in G' if there is no such edge in G, and u_{ij} is zero. Let $A' = (a'_{ij})$ be the adjacency matrix of G', and d'_i be the degree of node i in G'. Additionally, let m' be the number of edges in G'. The Max-Min Modularity (Q_{MM}) of a given partition $\mathbf{C}$ of V is defined as follows:

$$Q_{MM}(\mathbf{C}) = \sum_{i,j \in V} [\frac{1}{2m}(a_{ij} - \frac{d_i d_j}{2m}) - \frac{1}{2m'}(a'_{ij} - \frac{d'_i d'_j}{2m'})]\sigma(i,j) \tag{5.2}$$

We refer to the problem of partitioning a network with respect to maximizing the Max-Min Modularity as the *Max-Min Modularity Maximization* problem.

Chen et al. [41] proposed a hierarchical clustering algorithm, similar to what Newman [134] had offered for the classical Modularity Maximization Problem, that approximately greedily optimizes Max-Min Modularity. In addition to the suboptimal precision of the final community detection results obtained through the heuristic approach, the primary drawback of their method lies in its reliance on a user-defined relation matrix rather than a systematic approach. This dependency on subjective input introduces the potential for misinterpretations, biases, and erroneous human decisions, which can lead to significant issues. Relying on a user-defined matrix not only increases the likelihood of errors but also lacks the robustness and objectivity provided by a systematic and automated procedure.

Furthermore, since it is known that solving the Max-Min Modularity Maximization problem is computationally challenging, as it is proven to be NP-hard, most of the approaches for solving it rely on heuristic methods. [41]. There exist, of course, methods focusing more on exact techniques, such as the approach presented in [64], in which the authors successfully formulated the Max-Min Modularity Maximization problem as an *integer programming* model and proposed an equivalent sub-problem that simplified the overall formulation. This streamlined approach facilitated the efficient solving of the model's linear relaxation and provided a systematic means of defining the relation matrix. They further employed a local search strategy to convert the fractional solutions to integer ones that led to obtaining a set of communities. Nevertheless, despite their outstanding breakthrough in the modeling and the solution approach, the error caused by the rounding approach prevents us from obtaining an optimal solution, which is indeed crucial in scenarios where precise analysis is required. This underscores the need for alternative methods that can provide more accurate results.

Main contribution:
(1) Building upon the integer programming modeling discussed in [64], we present an alternative integer model for the Max-Min Modularity Maximization problem that offers a significant reduction in the number of variables and constraints. This streamlined model enhances computational efficiency while maintaining the same optimization capabilities. The equivalence of the proposed model and the original formulation proved in Theorem 1, affirms that both models yield the same set of optimal solutions.
(2) Inspired by the prominent algorithm proposed in [63], we devise a methodology to generate an initial feasible solution for the formulated model. By employing a row generation technique in combination with the branch and bound method and leveraging the powerful CPLEX[1] solver, we efficiently and optimally solve the model. This comprehensive approach enables the detection of a set of (near) optimal communities whose efficacy and effectiveness become apparent in the experiments done.

[1]CPLEX is a powerful optimization software package developed by IBM that employs advanced algorithms to solve linear programming, mixed-integer programming, and quadratic programming problems efficiently.

The structure of this paper unfolds as follows: In Section 5.2, we delve into modeling an effective integer programming formulation for the Max-Min Modularity Maximization problem, followed by an insightful theoretical exploration of how to simplify the model. Subsequently, Section 5.3 outlines our devised approach for acquiring an intelligent initial feasible solution and employing a row/column generation technique to solve the model optimally. Finally, Section 5.4 is dedicated to showcasing the experimental results, providing a comprehensive analysis of the outcomes.

5.2 Mathematical modeling

Let the binary variable x_{ij} indicate if nodes i and j belong to the same community or not; the value of x_{ij} is zero if nodes i and j belong to the same community, and one otherwise. Let $I_{all} = \{(i,j) \in V^2 \mid i < j\}$; and $q_{ij} = a_{ij} - \frac{d_i d_j}{2m}$, for each $(i,j) \in I_{all}$. As described in [56], the Modularity Maximization problem can be formulated in terms of the following integer linear program.

$$\max \quad \frac{1}{m} \sum_{(i,j)\in I_{all}} q_{ij}(1 - x_{ij}) \tag{IP-M}$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \qquad \forall i < j < k \tag{5.3}$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0 \qquad \forall i < j < k \tag{5.4}$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \qquad \forall i < j < k \tag{5.5}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in I_{all} \tag{5.6}$$

Constraints (5.3)-(5.5) guarantee that if i and j are in the same community and j and k are in the same community, then so are i and k. We refer to the relaxation of (IP-M), obtained by replacing the constraints $x_{ij} \in \{0,1\}$ by $x_{ij} \in [0,1]$, as (LP-M).

Now to turn our attention to the Max-Min Modularity Maximization problem, we first recap the systematic and precise approach provided in [64] for defining the relation matrix governed by an optimal fractional solution x^* to the linear programming problem (LP-M). Considerably crucial is that x^* can be efficiently obtained in polynomial time using various algorithms such as the row and column generation algorithm introduced by [126]. It is also important to note that x^* gives rise to a metric known as the LP distance on the graph G. In this context, x^*_{ij} can be interpreted as the "distance" between nodes i and j, and notably, the constraints (5.3)-(5.5) guarantee the fulfillment of the triangle inequality for any nodes $i, j, k \in V$ in the induced metric. Evidently, the larger the LP distance between two nodes, the less related those nodes are. This observation, along with the fact that the Modularity Maximization problem can be effectively formulated for weighted graphs as demonstrated by [132], serves as motivation to define the relation matrix and the corresponding complemented (weighted) graph G' utilizing the LP distance rather than using user knowledge.

In this framework, we define the relation matrix $A' = (a'_{ij})$ (and consequently G', with (a'_{ij}) representing the weight of the edge between nodes i and j in G'), as follows:

$$a'_{ij} = \begin{cases} x^*_{ij} & \text{if } a_{ij} = 0 \text{ and } j > i \\ x^*_{ji} & \text{if } a_{ij} = 0 \text{ and } i > j \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

Consequently, given a matrix $A' = (a'_{ij})$, the Max-Min Modularity Maximization problem can be formulated as the following integer programming problem. Let $c_{ij} = \frac{q_{ij}}{m} - \frac{q'_{ij}}{m'}$, where $q'_{ij} = a'_{ij} - \frac{d'_i d'_j}{2m'}$, $d'_i = \sum_{l=1}^{n} a'_{il}$, and $m' = \sum_{(i,j)\in I_{all}} a'_{ij}$ for each $(i,j) \in I_{all}$.

$$\max \sum_{(i,j)\in I_{all}} c_{ij}(1 - x_{ij}) \tag{IP-MM}$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \qquad \forall i < j < k \tag{5.8}$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0 \qquad \forall i < j < k \tag{5.9}$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \qquad \forall i < j < k \tag{5.10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in I_{all} \tag{5.11}$$

The very first thing to note, however, is that solving (IP-MM) falls in the class of NP-hard problems, making it challenging to be optimally solved. Consequently, we came to investigate whether it is possible to simplify (IP-MM) while maintaining the same set of optimal solutions. To achieve this objective, we have obtained the subsequent conceptual insights derived from the notion of row generation: The following lemma demonstrates that the optimal solution remains unaffected when only focusing on variables x_{ij} with $c_{ij} > 0$, and the subsequent theorem establishes that the optimal solution to (IPs-MM) without constraints involving x_{ij} where $c_{ij} \leq 0$ is equivalent to the optimal solution to (IP-MM).

Lemma 1 *If a binary variable x_{ij} satisfies the constraints* (6.5)*,* (6.6)*, and* (6.7)*, then it is sufficient to consider only the variables x_{ij} for which $c_{ij} > 0$ in the objective function* (IP-MM)*.*

Proof. Suppose we have a binary variable x_{ij} that satisfies the constraints (6.5), (6.6), and (6.7). We will show that if $c_{ij} \leq 0$, then x_{ij} will not affect the optimal solution of the objective function (IP-MM). Let us consider the term $c_{ij}(1 - x_{ij})$ in the objective function (IP-MM). If $c_{ij} \leq 0$, then regardless of the value of x_{ij} (0 or 1), the term $c_{ij}(1 - x_{ij})$ will be non-positive. Thus, including x_{ij} in the objective function with $c_{ij} \leq 0$ does not contribute to maximizing the objective. On the other hand, if $c_{ij} > 0$, including x_{ij} in the objective function can potentially increase the objective value by setting x_{ij} to 0 (i.e., nodes i and j belong to the same community) since $c_{ij}(1-0) = c_{ij}$. Therefore, it is sufficient to consider only the variables x_{ij} for which $c_{ij} > 0$ in the objective function (IP-MM). □

Theorem 1 *For the given integer programming problem, if we exclude the constraints involving variables x_{ij} where $c_{ij} \leq 0$, the optimal solution of the modified problem remains the same as the original problem.*

Proof. Let us assume that we have an optimal solution to the original (IP-MM) model, which satisfies all the constraints including those involving variables x_{ij} where $c_{ij} \leq 0$. We will show that by excluding these constraints, we can still obtain the same optimal solution. If x_{ij} satisfies the constraints (6.5), (6.6), and (6.7), its value will not change when we exclude the constraints involving $c_{ij} \leq 0$. The reason is that these constraints do not impose any restrictions on x_{ij}; they only provide additional information. Removing them does not alter the feasible region. Furthermore, since $c_{ij} \leq 0$, excluding these constraints means that the corresponding term $c_{ij}(1 - x_{ij})$ is non-positive and does not contribute to the objective function. Therefore, the objective value remains unchanged. Consequently, the optimal solution for the modified problem without constraints involving x_{ij} where $c_{ij} \leq 0$ is the same as the optimal solution for the original problem. This concludes the proof of the theorem. □

Having these considered, one can simplify (IP-MM) by considering only the variables x_{ij} where $c_{ij} > 0$ in the objective function. We can express this modified model as follows:

$$\max \sum_{(i,j)\in I_{pos}} c_{ij}(1 - x_{ij}), \tag{5.12}$$

where I_{pos} is the set of all pairs $(i,j) \in I_{all}$ for which $c_{ij} > 0$. We still need to ensure that the constraints (6.5), (6.6), and (6.7) hold for the selected variables. To achieve this, we introduce new binary variables y_{ijk} for all $i < j < k$ such that:

$$y_{ijk} = \begin{cases} 1 & \text{if } x_{ij} + x_{jk} - x_{ik} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

By introducing these variables, we can replace the constraints (6.5), (6.6), and (6.7) with the following constraint:

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \qquad \forall i < j < k \text{ s.t. } (i,j) \in I_{pos} \tag{5.14}$$

Finally, we include the binary variable definitions:

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in I_{pos} \tag{5.15}$$

$$y_{ijk} \in \{0,1\} \qquad \forall i < j < k \text{ s.t. } (i,j) \in I_{pos} \tag{5.16}$$

Gathering all together, we get an equivalent sparse model for (IP-MM) as follows:

$$\max \sum_{(i,j) \in I_{pos}} c_{ij}(1 - x_{ij}) \tag{IPs-MM}$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \qquad \forall i < j < k \text{ s.t. } (i,j) \in I_{pos} \tag{5.17}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in I_{pos} \tag{5.18}$$

$$y_{ijk} \in \{0,1\} \qquad \forall i < j < k \text{ s.t. } (i,j) \in I_{pos} \tag{5.19}$$

Highly encouraging is that (IPs-MM) has considerably fewer constraints and variables compared to the original (IP-MM) model but preserves the same set of optimal solutions.

5.3 Solution Approach

Despite (IPs-MM) providing us with a considerably simpler integer modeling than (IP-MM), it could still remain unlikely to obtain the optimal solution, particularly when dealing with average to large-scale networks. Undoubtedly, one possible way to speed up the branch and bound technique using the CPLEX solver is to feed it with a reasonably good initial solution. Starting with a smart choice among the feasible solution space has turned out to improve performance immensely. In this vein, we came to take advantage of the heuristic two-stage community detection algorithm introduced in [63]. Considerably relevant to our research is the first stage of the algorithm's authority to swiftly find a collection of initial communities with excellent quality w.r.t. optimizing a connectivity-based criterion they established.

In this procedure, the degree of a node and its set of neighbors are defined naturally. Additionally, the inner and outer edges of a community C, denoted as E_C^{in} and E_C^{out}, respectively, represent the edges with both endpoints and one endpoint within C. Let $\alpha \in \{1, \ldots, diameter(G)\}$ be an integer and define $P_\alpha \subseteq V$ as a sequence of nodes arranged in descending order according to their degrees, provided they are all at least α steps away from each other. P_α is referred to as the *influential nodes* of G with respect to α. The distance between a vertex $j \notin C$ and the community C is then determined

as the length of the shortest path from j to the influential node of C. The radius of a community C, denoted by $r(C)$, is the maximum distance from its influential node to other vertices within the community.

Subsequently, the authors of [63] proposed a measure to compute the quality of a community C:

$$q(C) = \frac{|E_C^{out}| + r(C)}{|E_C^{in}|}. \tag{5.20}$$

Furthermore, given a set of communities $\mathbb{C} = \{C_1, \ldots, C_k\}$, the quality of the partition $\mathbb{C}$ is defined as follow:

$$q_G^{\mathbb{C}} = \sum_{C \in \mathbb{C}} q(C). \tag{5.21}$$

Having everything considered, we can now summarize the procedure of determining an appropriate set of initial communities in the following manner:

- Repeat the procedure below for all α between 2 and $diameter(G)$ and pick $\mathbb{C}_\alpha$ with the minimum $q_G^{\mathbb{C}_\alpha}$ as the best initial set of communities.
  - Establish P_α as defined above.
  - Let each $p \in P_\alpha$ initially designate a sole community, leading to a set of $|P_\alpha|$ communities $\mathbb{C}_\alpha = \{C_1, \ldots, C_{|P_\alpha|}\}$.
  - Assign every $v \notin \mathbb{C}_\alpha$ to its closest community until all nodes belong to a community.

An essential insight of this approach revolves around the idea of ensuring a meaningful spatial distribution of the influential nodes of the network, striking a balance between proximity to facilitate cohesive communities and sufficient distance to avoid interference. To achieve this, the notion of $\alpha-$far nodes is introduced to serve as a criterion to evaluate the distance between potential influential vertices. By leveraging this parameter, one can successfully identify the best set of influential nodes capable of bunching and leading their surrounding vertices.

Now, by putting everything together, we devise the following tractable procedure for optimally solving (IPs-MM):

- Start with the initial communities obtained with the method explained above and employ the following row generation technique:
  1. Consider (IPs-MM) without any constraints.
  2. Use the CPLEX solver and apply the branch and bound technique to obtain an optimal solution x^* to (IPs-MM).
  3. Verify whether all constraints of (IPs-MM) are satisfied by x^*. If not, add the violated ones to the model and go to (2).

Table 5.1: Networks under-study

ID	Network	n	m
1	Zachary's karate club [184]	34	78
2	Mexican Politicians [79]	35	117
3	Dolphin network [120]	62	159
4	Les Miserables [120]	77	254
5	p53 protein [84]	104	226
6	Books about U.S. politics [122]	105	441
7	American college football [80]	115	613
8	Citation graph drawing [124]	311	640
9	USAir97 [21]	332	2126
10	C. Elegans [33]	453	2025
11	Erdos collaboration [129]	472	1314
12	Electronic circuit [39]	512	819

5.4 Computational results

Within this section, we conduct a comprehensive performance evaluation of our proposed methodology. To maintain fairness, we take into account exactly the set of 12 networks used in [64]. These networks, outlined in Table 6.1, are among the recognized and commonly utilized real-world networks utilized in this context, and each of them possesses a corresponding ground truth, representing the optimal community structures. Hence, it becomes convenient to evaluate the effectiveness of a community detection algorithm by quantifying the similarities between the algorithmically derived communities and the ground truth. To facilitate this evaluation, we employ the widely acknowledged performance metric NMI (*Normalized Mutual Information*).

5.4.1 Normalized Mutual Information (NMI)

NMI, as described in [50], is a widely recognized and established metric for evaluating the similarity between clusters. However, it can effectively measure the agreement between the optimal communities and those discovered by an algorithm. Consider a network G with n nodes, where $\mathcal{C}(\mathcal{A}) = C_1, \ldots, C_k$ represents the communities obtained by algorithm $\mathcal{A}$, and $\mathcal{C}' = C'_1, \ldots, C'_{k'}$ denotes the ground truth communities. The NMI value corresponding to the algorithm $\mathcal{A}$ can be computed as follows

$$NMI(\mathcal{A}) = \frac{-2\sum_{x=1}^{|\mathcal{C}|}\sum_{y=1}^{|\mathcal{C}'|}\frac{|C_x \cap C'_y|}{n} log(\frac{n|C_x \cap C'_y|)}{|C_x||C'_y|})}{\sum_{x=1}^{|\mathcal{C}|}\frac{C_x}{n} log(\frac{C_x}{n}) + \sum_{y=1}^{|\mathcal{C}'|}\frac{C'_y}{n} log(\frac{C'_y}{n})} \tag{5.22}$$

When the detected communities perfectly align with the ground truth, NMI attains its maximum value of one. Conversely, if the two sets exhibit no similarity, the NMI score

is zero. In general, a higher NMI value indicates a more accurate and effective discovery of community structures.

5.4.2 Experiments

In what follows, we present a thorough evaluation demonstrating our proposed method's superiority in achieving high-quality communities over several competing algorithms. All the experiments are conducted on a computer system with a processor *Intel(R) Core i9-12900KF @ 3.2GHz, 16.6 Core(s)* and *128 GB* of *RAM*. Algorithms are implemented with C++, and CPLEX optimizer 12.9 is used for solving linear programming.

Fig. 5.2 delivers the comparisons by evaluating communities that are discovered based on the following cases:

- Our method (the blue curve): Obtaining the communities by optimally solving (IPs-MM) with the proposed method consisting of a row and column generation procedure.
- Method proposed in [64] (the red diagram): Solving the linear relaxation model of (IP-MM) via a row and column generation technique and applying a local search manner rounding procedure for obtaining the communities.
- Max-Min Modularity, proposed in [41] (green diagram): Using a user-defined relation matrix and applying a hierarchical heuristic algorithm for maximizing the Max-Min Modularity.

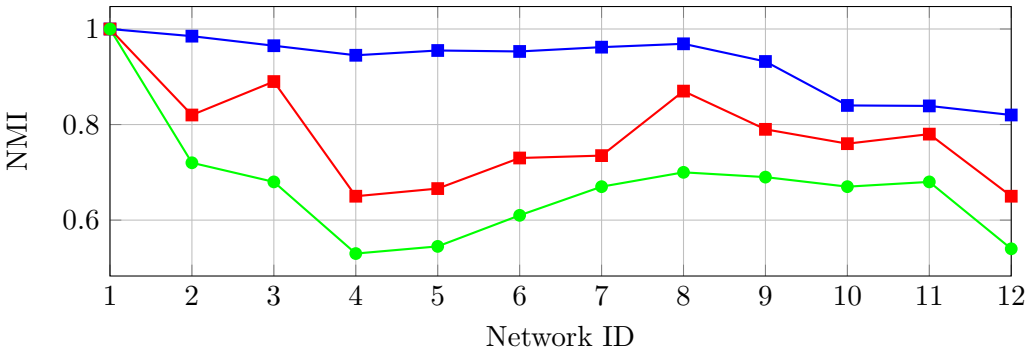
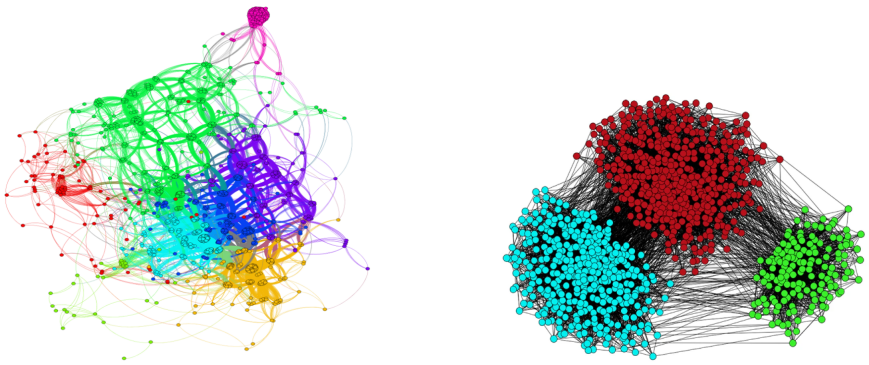


Figure 5.2: Comparison between NMI values achieved by (i) blue curve: our method, (ii) red curve: method in [64], and (iii) green curve: the conventional Max-Min modularity.

It is evident to conclude the promising outperformance of our proposed community detection method. In particular, the considerable gap between the blue and red curves clearly shows the advancement of using an exact method rather than relying on just heuristic approaches. While the technique in [64] took into account optimally solving the liner relaxation version of the (IP-MM), their proposed local search-based rounding procedure for obtaining the solution to (IP-MM) causes a significant error. The lever provided by the simpler model (IPs-MM), which was proven to be equivalent to (IP-MM), enabled us to seek an optimal solution to the model and, therefore, discover high-quality



(a) Erdos collaboration network (b) C. Elegans network

Figure 5.3: Two networks from Table 6.1 and their detected communities using the proposed algorithm.

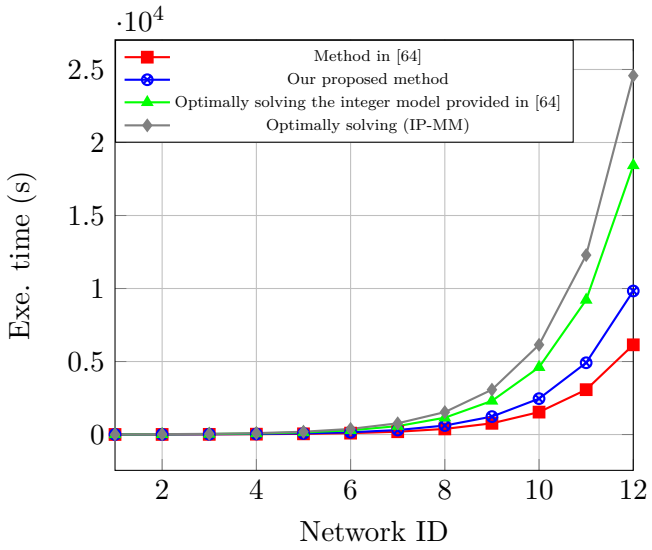


Figure 5.4: The time elapsed (in terms of seconds) for solving different methods.

communities considerably better than those in [64]. This dominance could be more pronounced when noticing that the communities obtained in [64] were superior to a wide range of other algorithms.

Herein, for the sake of more visualization, Fig. 5.3 displays the schematic representations of the *Erdos collaboration* and *C. elegans* networks, along with the communities identified by the proposed algorithm.

Furthermore, to strengthen the assessment of our algorithm, we also decided to examine its execution time, for which we came to follow twofold perspectives: first, to determine how solving (IPs-MM) instead of (IP-MM) could enhance time complexity, and second, to compare the execution time of our model with that of the integer programming model proposed in [64] for the Max-Min Modularity Maximization problem. Fig. 5.4 illustrates the results of these comparisons.

Our proposed row and column generation technique, coupled with the intelligently determined initial solution, resulted in a significant speed improvement when optimally solving (IPs-MM), surpassing the performance of solving (IP-MM). This highlights the substantial impact of our simplified model and solution approach. Furthermore, our method demonstrated faster execution compared to solving the equivalent integer formulation of the sub-problem for the Max-Min Modularity Maximization problem proposed in [64], which further validates the efficiency of our simplification. As could be naturally expected due to the NP-hardness nature of the problem, our model performs slower than when solving the LP relaxation version of the model in [64] plus using the rounding algorithm. Nevertheless, the inaccurate results obtained in [64] reveal its untrustworthy against this work's proposed model.

We complete this section by highlighting that even though the method presented in [64] can yield promising community structures in large-scale networks, for situations where accuracy is paramount, exact methods become significantly crucial. In such cases, the proposed method in this work can provide substantial assistance.

5.5 Conclusion

In this study, we addressed the Max-Min Modularity Maximization problem, a widely recognized metric for community evaluation. To enhance the problem's solution efficiency, we proposed an integer programming model that exhibits a reduced number of variables and constraints while preserving the same set of optimal solutions as the original model. By incorporating a row and column generation technique guided by an intelligently determined initial feasible solution, we were able to achieve optimal solutions in a remarkably efficient manner. The resulting solution provided us with a set of communities that exhibit notable similarities with the optimal community structures, indicating the effectiveness of our approach. This not only improved the overall quality of the obtained communities but also demonstrated the advantages of our model in terms of computational time.

theoremstyledefinition

Chapter 6

Gain and Pain in Graph Partitioning: Finding Accurate Communities in Complex Networks

Published in: *Journal of Algorithms*
Gain and Pain in Graph Partitioning: Finding Accurate Communities in Complex Networks.
A. Ferdowsi & M.DehghanChenary. © 2024 The Authors.
10.3390/a17060226

Abstract This paper presents an approach to community detection in complex networks by simultaneously incorporating a connectivity-based metric and Max-Min Modularity. By leveraging the connectivity-based metric and employing a heuristic algorithm, we develop a novel complementary graph for the Max-Min Modularity that enhances its effectiveness. We formulate community detection as an integer programming problem of an equivalent yet more compact counterpart model of the revised Max-Min Modularity maximization problem. Using a row generation technique alongside the heuristic approach, we then provide a hybrid procedure for near-optimally solving the model and discovering high-quality communities. Through a series of experiments, we demonstrate the success of our algorithm, showcasing its efficiency in detecting communities, particularly in extensive networks.

6.1 Introduction and literature review

In the intricate landscape of complex networks, the exploration of inherent structures, specifically the identification of cohesive subsets known as *communities*, has emerged as a foundational pursuit within network science and graph theory. Understanding communities and utilizing community detection in complex networks is pivotal for revealing real-world systems' intricate structures and interconnections. This process not only aids in deciphering the underlying dynamics of diverse networks, ranging from distributed communication networks to biological networks, but also empowers us to recognize outliers, predict the system's failures and future behaviors with greater precision, and optimize performance. These communities, comprised of highly interactive vertices, underpin diverse systems and offer insights into complex relationships, thereby transcending domains such as social, biological, and cosmological network analysis, distributed computing, signal processing, and data mining [100, 102, 90, 187, 77, 144, 121]. The quest to unveil and understand these communities has given rise to a spectrum of methodologies, each addressing the challenges posed by networks of varying complexities.

At the heart of community detection lies the concept of partitioning a network, represented as a graph $G = (V, E)$, into distinct subsets or communities $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$, where vertices within each community are as densely connected as possible. Various forms of the problem arise

Table 6.1: Several connectivity-based quality functions for measuring the quality of a community C. $|E_C^{out}|$, $|E_C^{in}|$, $|C|$ and $deg(i)$ respectively represents the number of external connections of C, number of internal connections of C, the number of vertices belonging to C, and the degree of node i.

	Function	Definition
IC	Internal density [142]	$\frac{\lvert E_C^{in}\rvert}{\lvert C\rvert(\lvert C\rvert-1)/2}$
	Edge inside [142]	$\lvert E_C^{in}\rvert$
	Average degree [142]	$\frac{2\lvert E_C^{in}\rvert}{\lvert C\rvert}$
EC	Expansion [142]	$\lvert E_C^{out}\rvert$
	Cut Retio [177]	$\frac{\lvert E_C^{out}\rvert}{\lvert C\rvert(\lvert V\rvert-\lvert C\rvert)}$
MIE	Conductance [156]	$\frac{\lvert E_C^{out}\rvert}{2\lvert E_C^{in}\rvert+\lvert E_C^{out}\rvert}$
	Normalized Cut [156]	$\frac{\lvert E_C^{out}\rvert}{2\lvert E_C^{in}\rvert+\lvert E_C^{out}\rvert}+\frac{\lvert E_C^{out}\rvert}{2(\lvert E\rvert-\lvert E_C^{in}\rvert)+\lvert E_C^{out}\rvert}$
	Out Degree Fraction [67]	$\max_{i\in C}\frac{\lvert(i,j)\in E : j\notin C\rvert}{deg(i)}$
	Flake-ODF [67]	$\frac{\lvert i\in C,\lvert(i,j)\in E: j\in C\rvert<deg(i)/2\rvert}{\lvert C\rvert}$

based on factors such as whether the network is weighted or unweighted, directed or undirected, whether the desired number of communities is predetermined, and whether communities exhibit overlap or complete separation [80, 41, 32, 155, 55, 133].

Crucially notable is that while there are non-optimization-based algorithms like the *Label Propagation* algorithm [143], a majority of community detection methods adhere to an optimization strategy. This involves first defining or selecting a specific metric, referred to as a *quality function*, to assess the excellence of communities and the devising an optimization algorithm to detect communities by optimizing the chosen quality function. Global optimization procedures, such as mathematical programming approaches [61, 64], provide accurate results. However, they face computational challenges when dealing with large networks [151]. Heuristic techniques, on the other hand, like bottom-up (merging communities), top-down (splitting communities), and local search (migrating nodes), present viable solutions, especially for extensive networks [40] but they come at the cost of losing accuracy. Although various community discovery approaches have surfaced in recent years [76, 148, 63], due to space limitations, our discussion concentrates on methods pertinent to our study.

Regarding optimization techniques, a wide variety of quality measures have been proposed, each offering a unique perspective to assess the goodness and accuracy of partitioning. Chakraborty et al. [38] categorize community quality functions into four broad types: *internal-connectivity-based* metrics (IC), *external-connectivity-based* metrics (EC), *mixed-internal-external*metrics (MIE), and metrics based on *network topology*. Table 6.1 outlines some well-known connectivity-based quality functions. For example, the *Average degree* function evaluates a community's quality based on both the number of edges with nodes inside the community and the number of nodes within the community.

While connectivity-based metrics have achieved widespread recognition [38], their exclusive focus on link structures, such as the number of edges and degrees, limits their ability to consider other significant features of the network, such as potential relationships between nodes. Therefore, topological metrics, among which Newman's *Modularity* [134] stands out as a widely used and recognized one, have garnered remarkable attention. Modularity assesses the quality of a partition by measuring the disparity between the observed number of edges within communities and the expected number based on a null model. Nevertheless, the traditional Modularity metric has faced criticism for its limitation in exclusively considering existing edges, disregarding the possibility

of connections between disconnected nodes within the same community. To overcome this drawback, an extension known as the *Max-Min Modularity* [41] introduces a complementary graph associated with the main network to capture relationships between disconnected nodes. The essential idea of Max-Min Modularity revolves around maximizing the Modularity value within the given network while concurrently minimizing this value across a complementary graph consisting of edges between unrelated nodes. This adjustment penalizes the Modularity measure when unrelated nodes coexist within a community. This is while, despite its improved accuracy, current Max-Min modularity approaches, like the one presented by Chen et al. [41], depend on user-defined relation criteria that introduce potential faultiness into the community detection process.

The naive dependency on an oracle for determining the complementary graph, as well as the not very successful heuristic hierarchy algorithm used in [41] for maximizing the Max-Min Modularity measure, directed Ferdowsi et al. in [64] and [61] to provide a more structured way to address the deficiencies. Their focus was on methodically structuring an effective complementary graph that accurately reflects the absence of connections between nodes. They enhanced this graph by applying the optimal solution to the *linear relaxation* formulation of the Modularity maximization problem. Using this systematically obtained complementary graph, they devised an *integer programming* representation for the Max-Min Modularity maximization problem and solved it using various methods. In [64], they applied the rational solution of the model's linear relaxation version to a local search rounding technique to obtain integer solutions and identified communities. In contrast, [61] theoretically developed and solved an equivalent yet sparse integer model, determining the exact solution to the problem. Consequently, whereas [64] succeeded in deriving community structure in a prompt manner due to the heuristic fashion employed, [61] achieved more precise results by obtaining optimal solutions to the integer model.

Both approaches yielded promising results, particularly those presented in [61], as it optimally solve a corresponding integer model. Nevertheless, several potential downsides exist. A primary drawback, not exclusive to these two schemes, is that almost all community detection methodologies take only one quality function into consideration. As a matter of fact, regardless of the optimization technique, each quality function tends to emphasize some distinct aspects of the network's structure. For instance, as mentioned, whereas a connectivity-based measure is geared more toward determining communities with densely connected nodes, topological approaches are capable of also concentrating on grouping related nodes. Another source of inefficiency lies in time complexity, which is especially evident when comparing two optimization methods, e.g., a heuristic and an exact one, both apply to the same quality function. Although the exact procedure outperforms the heuristic with regard to the quality of communities, it loses the ground to a heuristic approach in terms of time execution. Thus, enhancing time complexity remains a persistent challenge when employing an exact method.

Main contributions:

(1) We synergistically integrate two highly effective community quality functions: (i) the connectivity-based metric introduced in [63] and (ii) the Max-Min Modularity [64]. Specifically, we employ the connectivity-based metric and the heuristic algorithm proposed in [63] to establish a novel complementary graph associated with the Max-Min Modularity and to obtain a high-quality initial solution to the Max-Min Modularity maximization problem.

(2) We develop our community detection problem as an integer programming formulation of a revised Max-Min Modularity maximization problem (extending and improving the approach previously described in [61]).

(3) To mitigate time complexity, by deriving analytical insights, we demonstrate the time efficiency of the employed heuristic method, thereby validating the efficiency of obtaining the refined complementary graph. Besides, we present a more compact yet equivalent

model to the original integer formulation and demonstrate through theoretical arguments that the proposed sparse model yields the same optimal solution but with significantly fewer variables and constraints.

(4) We design a bound for controlling the degree to which the algorithm is supposed to delve into optimally solving the model and when it should get assistance from the heuristic method. More specifically, taking the promising communities generated by the heuristic method as the initial solution to the sparse counterpart of the proposed Max-Min Modularity maximization problem's integer model, we devise a procedure for balancing between heuristic and exact optimization based on the size of the given network. This facilitates the detection of communities even in extensive networks.

(5) Through various experiments, we showcase the success of our proposed algorithm.

Paper organization: In Section 6.2, in the sequel, we review the heuristic approach introduced in [63] and propose some theoretical insights regarding its efficiency. We then recap the traditional Modularity and Max-Min Modularity methodologies. Afterward, in Section 6.2.3, we delve into presenting our approach, and finally, in Section 6.3, we bring forward the experimental results. Some conclusions are provided in Section 6.4.

6.2 Methodology

This section aims to amalgamate two optimization approaches: 1) a heuristic method for minimizing a connectivity-based quality metric and 2) an exact approach for maximizing a topology-based quality measure. The objective is to formulate a hybrid community detection algorithm capable of identifying accurate communities within a reasonable timeframe.

6.2.1 Heuristic but fast

This section reviews the two-stage heuristic optimization-based community detection algorithm proposed in [63], which involves optimizing a connectivity-based quality metric. We first revisit the requirements and definition of the suggested quality metric, then outline the optimization algorithm. Afterward, we develop some theoretical arguments contributing to a more comprehensive evaluation of the algorithm's validity across varying network sizes.

What is worth noting about this fast yet accurate heuristic approach is our aim to 1) propose a systematic way to improve the complementary graph of the Max-Min Modularity and 2) obtain a high-quality initial solution for the exact optimization procedure that will be explained in Section 6.2.2.

Let the *degree* of a node and its set of neighbors be defined naturally. Additionally, the *inner* and *outer edges* of a community C, denoted as E_C^{in} and E_C^{out}, respectively, represents the edges with both endpoints and one endpoint within C. We proceed to define the *influential nodes*, which can be envisioned as vertices endowed with authority and the capacity to control and affect other nodes, particularly in our context, shaping communities. As elucidated in [63], nodes with high degrees that maintain an appropriate distance from each other emerge as suitable candidates for such roles. The intuition comes from the fact that the degree of nodes can substantially reflect the importance of vertices within the network. A high-degree vertex influences its surroundings more than a low-degree vertex. This is because it can pass messages to multiple nearby vertices and impose various features and information on them. For instance, a high-degree vertex in a communication network can significantly impact the spread of data and messages, affecting the behavior of its neighbors and the overall dynamics of the network. Think of the effect of a high-degree byzantine faulty process in a distributed computing network. Definition 1 gives a more formal characterization of the influential nodes.

Definition 1 (Influential node) *Let $\alpha \in \{1, \ldots, diameter(G)\}$ be an integer and define $P_\alpha \subseteq V$ as a sequence of nodes arranged in descending order according to their degrees, provided they are all at least α steps away from each other. P_α is referred to as the influential nodes of G with respect to α.*

Note that the influential node of a community is not necessarily unique. In such a case, the influential node will be randomly picked among the candidates.

Definition 2 (Neighboring communities) *Two communities C and C' are said to be neighbors if there exists an edge $(i, j) \in E$ connecting $i \in C$ to $j \in C'$. In this case, we refer to both i and j as boundary vertices of C respectively C'. Let $BV(C)$ denotes the set of all boundary vertices of C.*

Definition 3 (Distance to a community) *The distance between a vertex $j \notin C$ and the community C is determined as the length of the shortest path from j to the influential node of C.*

Definition 4 (Radius) *The radius of a community C, denoted by $r(C)$, is the maximum distance from its influential node to other vertices within the community.*

The influence a particular node exerts on others is evidently correlated with the distance between them. Valuable insights into this relationship can be gleaned from studies such as [9, 64]. For instance, note a more significant impact a high-degree node in a communication graph has on vertices connecting to it directly (distance = 1) than on those that are connected to it through several intermediaries (distance ¿ 1). Consequently, it is natural to assume that a community with a small radius is likely to be more interactive and homogeneous. Subsequently, the authors of [63] proposed the following measure to compute the quality of a community C:

$$q(C) = \frac{|E_C^{out}| + r(C)}{|E_C^{in}|}. \tag{6.1}$$

Furthermore, given a set of communities $\mathbf{C} = \{C_1, \ldots, C_k\}$, the quality of the partition $\mathbf{C}$ is defined as follows:

$$q_G^{\mathbf{C}} = \sum_{C \in \mathbf{C}} q(C). \tag{6.2}$$

Note that the quality of the partition $\mathbf{C}$ increases as $q_G^{\mathbf{C}}$ decreases. More specifically, the proposed community quality measure determines the quality of a community C by minimizing the sum of the following two terms: $\frac{|E_C^{out}|}{|E_C^{in}|}$ and $\frac{r(C)}{|E_C^{in}|}$. As minimizing the number of outer edges while simultaneously maximizing the number of inner edges naturally leads to enhancing the quality of community structures, reducing the radius yields a similar outcome. Recall the significant role of the vertex's degree in controlling and stimulating neighboring nodes due to high-degree vertices' capability to impose distinct attributes on surrounding nodes [86]. Specifically, many (complex) networks exhibit a small number of influential high-degree vertices, such as social media influencers, pivotal commodities in co-purchasing networks, or a high-degree processor in a distributed computing network. These high-degree vertices are typically surrounded by numerous lower-degree vertices [138, 60]. In such networks, each vertex is either directly connected to or only a few links away from a high-degree node. With this in mind, having communities with small radii breaks the task into finding influential nodes that are at a proper distance from each other, as it turned out in [63] that an appropriate distance between high-degree nodes results in the formation of well-isolated communities.

For this, the proposed optimization algorithm works in a way that, with the general goal of minimizing the quality measures (6.1) and (6.2), it first establishes a set of promising initial communities (Algorithm 1) that then get purified with a local search manner (Algorithm 2). Simply put, the first stage seeks for the best α value leading to well-separated communities formed surrounding the set of influential vertices P_α. Equivalently, the goal of this phase restricts the objective to finding authoritative nodes that are at the right distance from each other, letting them freely absorb other nodes into their territory.

Algorithm 1 Initial partitioning (IP)

Require: Graph G
Ensure: Set of initial communities $\mathbf{C}$ and the corresponding set of influential nodes P

1: **for** $\alpha = 2$ **to** *diameter*(G) **do**
2: Establish $P_\alpha = \{p_1, p_2, \ldots, p_m\}$ ▷ based on Definition 1
3: $\mathbf{C}_\alpha \leftarrow \{\, C_1 = \{p_1\},\ \ldots,\ C_{|P_\alpha|} = \{p_m\} \,\}$
4: **while** $\exists\, v \notin \bigcup_{i=1}^{|P_\alpha|} C_i$ **do**
5: Assign v to its closest community $C_i \in \mathbf{C}_\alpha$
6: Record $\mathbf{C}_\alpha$, the corresponding P_α, and the value $q_G^{\mathbf{C}_\alpha}$
7: **return** the set of communities $\mathbf{C}$ and influential nodes P associated with the minimum quality value $q_G^{\mathbf{C}}$

The second phase, afterward, updates initial communities using a local search technique that iteratively selects a community, favoring worse quality values. It then greedily improves communities with one of three functions: *Merge*, *Split*, and *Swap*. While the *Merge* function assesses merging a community C into neighbor communities for quality improvement, the *Split* function checks if applying the CIC stage to the induced sub-graph G' formed by C can divide C into separate communities that result in a better quality value. The *Swap* function, on the other hand, optimizes node swaps between C and its neighbors. The procedure updates communities and influential vertices after each iteration, and it continues until either no further local search improvements are possible or a certain number of iterations have been done.

This heuristic two-stage community detection algorithm stands out as a powerful approach to identifying well-separated communities within complex networks concerning minimizing the introduced quality measure. Evidently, depending on the quality of initial communities obtained in the first step, one can effectively reduce the need for reforms in the revising stage. This is particularly significant when time constraints are critical, resulting in fewer executions of the revising stage. And this is while according to [63], the first stage indeed returns high-quality communities, enabling us to confidently decrease the number of iterations in the second phase or even consider skipping it. In the theoretical explorations below, we state some mathematical foundations of this algorithm, aiming to unveil essential insights into its efficiency and solution quality. Particularly, since the algorithm's first phase aims to establish initial communities by determining the best set of influential nodes P_α, to maintain efficiency, $|P_\alpha|$ should be manageable.

Lemma 1 *Consider a graph $G = (V, E)$ and let α be an integer such that $2 \leq \alpha \leq$ diameter(G). Let P_α be the set of corresponding influential nodes, meaning, the set of nodes arranged in descending order according to their degrees, provided they are all at least α steps away from each other. Then, the time complexity of building P_α is of $O(n(n+m))$.*

Proof 1 *Let n be the number of nodes in G. To analyze the time complexity of building P_α, we need to consider the steps involved in constructing this set:*

Algorithm 2 Partition Improvement Communities (PI)

Require: Graph G, set of initial communities $\mathbf{C}$, set of influential vertices P, integer k
Ensure: Set of revised communities $\mathbf{C}$

$q \leftarrow q_G^{\mathbf{C}}$
$q_{\text{temp}} \leftarrow 0$
Iteration $\leftarrow 1$
while $q \leq (1-\epsilon)\, q_{\text{temp}}$ **and** Iteration $\leq k$ **do** ▷ Small constant ϵ ensures polynomial running time [13, 85]
    $q_{\text{temp}} \leftarrow q$
    select a community $C \in \mathbf{C}$ with probability $\frac{q(C)}{q_G^{\mathbf{C}}}$
    $(\mathbf{C}, P) \leftarrow \text{BestMovement}(C, \mathbf{C}, P)$
    $q \leftarrow q_G^{\mathbf{C}}$
    Iteration $\leftarrow$ Iteration $+1$
return $\mathbf{C}$

function BestMovement$(C, \mathbf{C}, P)$
    $(\mathbf{C}_{\text{tempM}}, P_{\text{tempM}}) \leftarrow \text{Merge}(C, \mathbf{C}, P)$
    $(\mathbf{C}_{\text{tempS}}, P_{\text{tempS}}) \leftarrow \text{Split}(C, \mathbf{C}, P)$
    $(\mathbf{C}_{\text{tempSw}}, P_{\text{tempSw}}) \leftarrow \text{Swap}(C, \mathbf{C}, P)$
    Among the returned partitions, retrieve the best set of communities $\mathbf{C}$ and its influential vertices P corresponding to the minimum obtained quality value
    return $(\mathbf{C}, P)$
function Merge$(C, \mathbf{C}, P)$
    for every $C' \in \mathbf{C} \setminus C$ such that C' is a neighbor of C **do**
        merge C and C'
        update and record $\mathbf{C}$ and its influential vertices P
    retrieve the best $(\mathbf{C}, P)$ based on minimum obtained quality value
    return $(\mathbf{C}, P)$
function Split$(C, \mathbf{C}, P)$
    $IG \leftarrow$ induced subgraph of G on vertices of C
    $(\mathbf{C}', P') \leftarrow \text{CIC}(IG)$
    if $|\mathbf{C}'| > 1$ **and** $q_{G'}^{\mathbf{C}'} \leq q(C)$ **then**
        $\mathbf{C} \leftarrow (\mathbf{C} \setminus C) \cup \mathbf{C}'$
        update P
    return $(\mathbf{C}, P)$
function Swap$(C, \mathbf{C}, P)$
    for every community $C' \in \mathbf{C}$ that is a neighbor of C **do**
        for every pair $(i, j) \in BV(C, C')$ **do**
            swap the communities of i and j if it decreases the quality value
    return $(\mathbf{C}, P)$

1. *Sorting nodes by degree: sorting n nodes based on their degrees takes $O(n \log(n))$ time.*

2. *Constructing P_α: we iterate through the n nodes. For each node v, we need to check its distance to all nodes in P_α. In other words, for each node v, we should verify that it is at least α steps away from all nodes in P_α. To do this, we can use* Breadth-First Search *(BFS) starting from v. BFS runs in $O(n+m)$ time, where m is the number of edges in Graph G.*

Thus, the total time complexity for constructing P_α is dominated by the distance verification step, giving us: $O(n(n+m))$.

Theorem 1 (Time efficiency of the first phase of the algorithm) *For a graph $G = (V, E)$ with n nodes and m edges, the first phase of the algorithm, which involves the identification of α-far nodes and the creation of initial communities, operates in $O(n^2(n+m))$ time, where k is a constant.*

Proof 2 *The first phase involves constructing P_α and creating initial communities from α-far nodes. As shown in Lemma 1, P_α can be determined in $O(n(n+n))$ time. Hence, as the assignment of the remaining nodes to communities can take place in at most $O(n)$ time, the construction and initialization steps require $O(n^2(n+m))$ time.*

Having the polynomial time complexity of the first phase of the algorithm proved, one can deduce the similar complexity for the whole algorithm.

Theorem 2 (Time efficiency of the whole algorithm) *For a graph $G = (V, E)$ with n nodes, the whole heuristic algorithm consisting of establishing initial communities and refining them operates in $O(n^4(n+m))$ time.*

Proof 3 *Assume that the second phase iterates for p times. Then the worst case complexity of this stage would be of $O(3pn^2)$, since all the three revising operations, Merge, Split, and Swap in this stage occur in $O(n^2)$. Therefore, taking the time complexity of the first phase into account, we can conclude that the whole approach functions in $O(n^4(n+m))$.*

Theorem 3 (An approximation community detection algorithm) *For a graph $G = (V, E)$, the heuristic algorithm provides an approximation algorithm for community detection.*

Proof 4 *Note that a sufficiently small value of ϵ makes the second phase pass through all combinations of possible partitioning, pushing it to visit ultimately the optimal set of communities $\mathbf{C}_{OPT}$ with the quality value of $q^* = q_G^{\mathbf{C}_{OPT}}$. Suppose that it happens after a finite number of iterations l. Moreover, due to the improvement factor $1-\epsilon$, it is evident that after r iterations of this phase, the quality value will be less than or equal to $(1-\epsilon)^{r-1}q_{init}$, where q_{init} is the quality value of the set of initial communities achieved by the first phase. Now, point out that in order for the two-stage algorithm to be a β-approximation algorithm, one needs to find a constant β that βq^* becomes less than or equal to the quality value of the partitioning of the algorithm. Assuming that the algorithm runs for a given k_1 times results that $\beta q^* = \beta(1-\epsilon)^{l-1}q_{init} \leq (1-\epsilon)^{k_1-1}q_{init}$, which lead to $\beta < (1-\epsilon)^{k_1-l}$.*

We conclude this section by highlighting the leverage of the first phase not only to yield promising communities that stand alone as reliable results but also to achieve such outcomes in a sufficiently rapid manner (i.e., in $O(n^2(n+m))$). In addition, joining it to the practical second phase for improving the communities provides an approximation algorithm guaranteeing a final set of accurate communities. What is crucial to mention at the end is our choice in determining the extent to which the algorithm delves into the second phase. This pivotal decision allows for a reasonable equilibrium, finely attuned to the structure and magnitude of the network. Consequently, scaling up the network size offers a seamless avenue to curtail the number of iterations required for the second phase. As will be illustrated in Section 6.3, this strategic approach significantly optimizes computational efficiency.

6.2.2 Toward exactness

Inevitably, the primary objective of any community detection algorithm is to identify communities as accurately as possible within a reasonable timeframe. For the former, it seems necessary to shift the focus from heuristic methods to exact optimization techniques, even though it would fight against the time complexity. On the other hand, utilizing a mix of quality metrics could also lead to accuracy improvement, as each quality metric can be tailored to suit specific graph structures within a defined range.

Accordingly, in this section, to enhance the quality of communities, we first concentrate on *Max-Min Modularity*, which is recognized as one of the most applicable topology-based metrics. Using this metric, we then express the Max-Min Modularity maximization problem as a mathematical programming. Solving this formulation allows us to optimize the measure and, consequently, detect highly accurate communities. For this, let us briefly explain the foundational principles of Modularity. This framework serves as the basis for the introduction of various Max-Min Modularity variants, which we will also discuss subsequently.

Modularity, as introduced by Newman [134], is widely acknowledged as a measure in network analysis. For a network G with n vertices and m edges, the Modularity value (Q) of a given partitioning $\mathbf{C}$ is mathematically expressed as follows:

$$Q(\mathbf{C}) = \frac{1}{2m} \sum_{i,j \in V} [a_{ij} - \frac{d_i d_j}{2m}] \sigma(i,j). \tag{6.3}$$

Here, $A = (a_{ij})$ is the adjacency matrix of G, where a_{ij} is set to one if an edge exists between node i and node j, and zero otherwise. d_i represents the degree of node i, defined as the sum of all entries in the i-th row of the adjacency matrix. Additionally, $\sigma(i,j)$ is one if nodes i and j are in the same community and zero otherwise. In simpler terms, Modularity quantifies the number of edges within a community minus the expected number of such edges. Communities with higher Modularity values are considered of better quality. Therefore, maximizing Modularity aids in identifying high-quality communities within a network.

However, despite the widespread use of Modularity, it has known limitations (refer to [59, 38] for details). Notably, Modularity only accounts for existing edges in the network, evaluating the goodness of a community solely based on its fit with observed edges. It overlooks disconnected nodes (absent edges) within the same community, which is a drawback since node disconnection does not necessarily imply an absence of underlying relations between them. To address this limitation, an extension of Modularity called *Max-Min Modularity* [41] has been developed, aiming to enhance accuracy by penalizing Modularity when disconnected nodes are present in the same community. The core concept behind Max-Min Modularity involves maximizing the Modularity value within the given network G while simultaneously minimizing this value across a complementary graph G'. The latter has an identical set of nodes as G, and the presence of an edge between nodes i and j in G' indicates that, firstly, i and j are not connected in G, and secondly, these nodes are relatively unrelated. The rationale behind Max-Min Modularity is quite intuitive: by refining communities to reduce the occurrence of unrelated nodes, nodes not only disconnected but also unrelated within the same communities, the quality of the partitioning is significantly enhanced.

With this in mind, the Max-Min Modularity (QMM) of a given partition $\mathbf{C}$ of V can be defined as follows:

$$Q_{MM}(\mathbf{C}) = \sum_{i,j \in V} [\frac{1}{2m}(a_{i,j} - \frac{d_i d_j}{2m}) - \frac{1}{2m'}(a'_{i,j} - \frac{d'_i d'_j}{2m'})] \sigma(i,j), \tag{6.4}$$

where $A' = (a'_{ij})$ is the adjacency matrix of G' and d'_i is the degree of node i in G'. Besides, m' represents the number of edges in G'.

Particularly relevant in this regard is that despite different approaches for optimizing the Max-Min Modularity metric, a pivotal concern lies in defining an appropriate complementary graph G' capable of accurately representing the absence of relations between nodes. For instance, [41] naively relied on an expert to determine such a graph. In a more systematic fashion, [61] and [64] managed to obtain the complementary graph by employing the optimal solution to the linear relaxation formulation of the Modularity maximization problem. Utilizing the acquired complementary graph, the authors of both [61] and [64] formulated an integer representation of the Max-Min Modularity maximization problem and approached its solution through distinct methods. In the case of [64], the rational solution derived from the linear relaxation of the problem was input into a local search rounding technique to determine integer solutions and extract communities. On the other hand, [61] theoretically formulated an equivalent yet sparse integer model to the primary model, facilitating the direct solution of the integer model. While the adoption of the linear programming (LP) model and a heuristic rounding approach by [64] expedited the community identification process, [61] achieved notably more accurate results by optimizing the solution of the integer model.

Remarkably, despite the promising outcomes from both methodologies, we identified potential for enhancement, particularly in terms of simultaneously reducing execution time and refining community quality. Our primary avenue for such improvement stems from a modified formulation of the Max-Min Modularity maximization problem, which we introduce in the subsequent subsection.

6.2.3 The proposed hybrid approach

In this section, we bring forward a successful approach for obtaining accurate communities by optimally optimizing a revised version of the well-known topological community quality metric, *Max-Min Modularity*. What we intend to do is to use the already introduced heuristic method to propose a new complementary graph (Definition 5) and then apply our theoretical findings in [61] to derive a sparse integer model for the revised Max-Min Modularity, which we then try to solve optimally using the branch and bound technique.

Definition 5 (Complementary graph G') *Let $G = (V, E)$ be a given network with a corresponding set of communities $\mathbf{C} = \{C_1, \ldots, C_l\}$ identified by the heuristic algorithm. We associate with $G = (V, E)$ a complementary graph $G' = (V, E')$, such that $(i, j) \in E'$ if and only if $(i, j) \notin E \ \wedge \{i, j\} \notin C_r, r = 1, 2, \ldots, l$.*

In essence, two disconnected nodes i and j in G, establishes an edge in the complementary graph G', provided that i and j do not belong to a same community specified by the heuristic algorithm 6.2.1. It is crucial to emphasize the advantages of incorporating this complementary graph into the Max-Min Modularity metric. Firstly, the high-quality communities already obtained by the heuristic algorithm validate the absence of connections between nodes belonging to different communities. Secondly, this approach allows the simultaneous consideration of two quality metrics. While maximizing the topology-based metric, Modularity, over the main network has been proven in the literature to yield highly accurate communities, minimizing it over a graph consisting of unrelated nodes identified by a connectivity-based metric can significantly enhance the results. Additionally, in terms of time complexity, our theoretical findings regarding the heuristic method, coupled with the ability to confidently adjust the number of iterations in the revision phase of the algorithm or even omit it due to the favorable quality of the initial phase, enable us to achieve results in a shorter timeframe.

Having G' defined, we now provide the integer formulation of our *Revised Max-Min Modularity maximization problem* (IP-RMM):

$$\max \sum_{(i,j)\in I_{all}} c_{ij}(1-x_{ij}) \qquad \text{(IP-RMM)}$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \qquad \forall\ (i,j),(j,k),(i,k) \in I_{all} \qquad (6.5)$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0 \qquad \forall\ (i,j),(j,k),(i,k) \in I_{all} \qquad (6.6)$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \qquad \forall\ (i,j),(j,k),(i,k) \in I_{all} \qquad (6.7)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in I_{all} \qquad (6.8)$$

where the binary variable x_{ij} indicates if nodes i and j belong to the same community ($= 0$) or not ($= 1$). Moreover, $I_{all} = \{(i,j) \in V^2 \mid i < j\}$, $c_{ij} = \frac{q_{ij}}{m} - \frac{q'_{ij}}{m'}$ with $q_{ij} = a_{ij} - \frac{d_i d_j}{2m}$, $q'_{ij} = a'_{ij} - \frac{d'_i d'_j}{2m'}$, $d'_i = \sum_{l=1}^{n} a'_{il}$, and $m' = \sum_{(i,j)\in I_{all}} a'_{ij}$ for each $(i,j) \in I_{all}$. Constraints (6.5)-(6.7) satisfy the triangle inequalities and guarantee that if i and j are in the same community and j and k are in the same community, then so are i and k.

The primary consideration is that solving (IP-RMM) falls into the category of *NP-hard* problems, presenting a challenge for optimal resolution. Consequently, we embarked on an investigation to explore the possibility of simplifying (IP-RMM) while preserving the same set of optimal solutions. To attain this objective, we derived the following conceptual insights from the notion of row generation, all borrowed from [61]: The subsequent lemma illustrates that the optimal solution remains unaffected when concentrating solely on variables x_{ij} with $c_{ij} > 0$, and the subsequent theorem establishes that the optimal solution to (IP-RMM), without constraints involving x_{ij} where $c_{ij} \leq 0$, is equivalent to the optimal solution to (IP-RMM).

Theorem 4 *Excluding variables x_{ij} and corresponding constraints with $c_{ij} \leq 0$ from the integer programming problem* (IP-RMM) *does not change the optimality. That is, the optimal solution of the modified problem remains the same as the one for* (IP-RMM).

Proof 5 *Suppose we have a binary variable x_{ij} that satisfies the constraints* (6.5), (6.6), *and* (6.7). *We shows that if $c_{ij} \leq 0$, then the variable x_{ij} will not contribute to maximizing the objective function* (IP-RMM). *Consider the term $c_{ij}(1-x_{ij})$ in the objective function* (IP-RMM). *If $c_{ij} \leq 0$, irrespective of the value of x_{ij} (0 or 1), the term $c_{ij}(1-x_{ij})$ will be non-positive. Consequently, incorporating x_{ij} in the objective function with $c_{ij} \leq 0$ does not play any role in maximizing the objective. Conversely, if $c_{ij} > 0$, including x_{ij} in the objective function can potentially enhance the objective value by setting x_{ij} to 0 (i.e., nodes i and j belong to the same community) since $c_{ij}(1-0) = c_{ij}$. Therefore, it suffices to consider only the variables x_{ij} for which $c_{ij} > 0$ in the objective function* (IP-RMM).

In addition, The constraints (6.5), (6.6), *and* (6.7) *ensure that the variables x_{ij} adhere to the triangle inequalities, maintaining consistency in community assignments. Removing constraints involving x_{ij} with $c_{ij} \leq 0$ does not change the feasible region of the problem because these constraints do not affect the feasibility of the solution.*

Thus, excluding variables x_{ij} and corresponding constraints where $c_{ij} \leq 0$ does not change the optimal solution.

Taking into account the above theorem, we can simplify the original model (IP-RMM) by focusing only on the variables x_{ij}, where $c_{ij} > 0$. With letting $I_{pos} = \{(i,j) \in I_{all} \mid c_{ij} > 0\}$, the revised integer programming model can be formulated as follows:

$$\max \sum_{(i,j)\in I_{pos}} c_{ij}(1 - x_{ij}) \qquad \text{(IPs-MM)}$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \qquad \forall\ (i,j),(j,k),(i,k) \in I_{pos} \qquad (6.9)$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0 \qquad \forall\ (i,j),(j,k),(i,k) \in I_{pos} \qquad (6.10)$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \qquad \forall\ (i,j),(j,k),(i,k) \in I_{pos} \qquad (6.11)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in I_{pos} \qquad (6.12)$$

An encouraging observation is that (IPs-MM) exhibits significantly fewer constraints and variables in comparison to the original (IP-RMM) model while maintaining an equivalent set of optimal solutions. Nevertheless, obtaining the optimal solution for average to large-scale networks might still pose challenges. Undoubtedly, a viable approach to expedite the branch-and-bound technique, especially when utilizing the CPLEX[1] solver, is to provide it with a sufficiently good initial solution. Commencing with a judicious selection from the feasible solution space has proven to significantly enhance performance. In this context, we once again leverage the proposed heuristic algorithm. Specifically, we employ the communities generated by the heuristic approach as an initial solution for solving (IPs-MM).

With all elements considered, Algorithm 3 frames our method for (near) optimally identifying communities. It is essential to note the integer inputs k_1 and k_2 of the algorithm, which control the number of iterations in the second phase of the heuristic algorithm and the number of repetitions considered for solving (IPs-MM), respectively. The latter parameter is intertwined with the number of the model's constraints involved in the optimization. Note that due to the NP-hard nature of (IPs-MM), a row generation technique has been employed. This technique initially solves the problem without considering any constraints and then iteratively adds those that are not satisfied by the obtained solution. Whereas assuming $k_2 = \infty$ lets the algorithm terminate whenever it finds the optimal solution, this could prolong execution times for huge networks. To address this, we came to introduce a bounding mechanism, limiting the number of times the algorithm is allowed to add violated constraints. While this bounding introduces some inaccuracy, due to the smart choice of the initial solution, we do not end far from the optimal solution. Furthermore, the promising initial solution reduces the likelihood of encountering cases where the algorithm would reach the optimal solution with all constraints satisfied after k_2 iterations.

As a complementary to the above discussion, one might point out that whereas the golden standard for determining optimal communities is to let both k_1 and k_2 be sufficiently large, it does not sound logical especially when dealing with large networks. What is important to note here is the advantage one can take from the trade-off between these two parameters. More specifically, despite the promising quality of the initial communities, the more the number of k_1, the better the communities become, which leads to less concern regarding the optimality of the (IPs-MM). In contrast, the fewer reforms done due to the small value of k_1 seem to require more care with finalizing the communities, which highlights the importance of achieving an optimal solution to (IPs-MM) with as many satisfied constraints as possible, which itself depends on picking a large value of k_2. To summarize, one can conclude that picking a large value of k_1 can relax a large choice of k_2, and vice versa. On the other hand, as already observed in [63], k_1 does not need to be large due to already high-quality initial communities obtained by the first phase of the heuristic algorithm, which has turned out to be efficient also in terms of time execution. This provides us with a final useful hint: Choosing generally a relatively small value of k_1, especially in the case of dealing with large networks, and presuming k_2 to be large enough to make a safe

[1]CPLEX is a powerful optimization software package developed by IBM that employs advanced algorithms to solve linear, mixed-integer, and quadratic programming problems efficiently.

margin about having the majority of constraints of (IPs-MM) satisfied could lead us to obtain (near) optimal community structures. The experiments in the next section validate this assertion.

Algorithm 3 The proposed algorithm for community discovery

Require: Graph G and integers k_1 and k_2
Ensure: Set of communities $\mathbf{C}$
$\mathbf{C} \leftarrow UC(G, CIS(G), k_1)$ ▷ Heuristic approach (Algorithms 1 and 2)
Iteration $\leftarrow 1$
Formulate (IPs-MM) without any constraints
while true **do**
    Solve (IPs-MM) using the branch and bound method to obtain the optimal solution x^*
    if all constraints of (IPs-MM) are satisfied by x^* **or** Iteration $> k_2$ **then**
        break
    else
        Add the violated constraints to the model
        Iteration $\leftarrow$ Iteration $+1$
return the set of communities $\mathbf{C}$ associated with the solution x^*

6.3 Evaluation of accuracy

In this section, we delve into verifying the efficiency of our algorithm throughout various experiments. Due to our method having been built upon the heuristic and the exact method introduced in [63] and [61], we came to, first of all, compare outcomes with the one determined by those two approaches.

It is worth noting that [63] succeeded in surpassing various state-of-the-art heuristic community detection algorithms, like Distributed Louvain (DLouvain) [78], Variable Neighborhood Search (VNS) [8], Simulated Annealing (SA) [180], and FastGreedy [133] algorithms, both in terms of execution time and communities quality. In addition, [61] managed to detect communities superior to the ones obtained by [64] and [41], where in the former, the optimal solution to the linear relaxation model of ordinary Max-Min Modularity was fed into a local search manner rounding procedure for obtaining the communities. In the later, communities were discovered by using the user-defined complementary graph introduced in [41] and also applying its proposed hierarchical heuristic algorithm for maximizing the Max-Min Modularity.

Having this in mind, if our algorithm can outperform all the above approaches, it would substantiate its efficiency. For this reason, we take all the network benchmarks used in [63, 61] and compare the results of all algorithms with each other. Furthermore, we also generate and use six *Lancichinetti Fortunato Radicchi* (LFR) artificial benchmarks [111], which closely resemble real-world complex networks by incorporating a pre-defined community structure. Generating these artificial graphs, which will be explained later, is based on user-defined parameters fully elucidated in [111]. What makes the comparison fair is that all the under-study graphs, including the real-world networks presented in Table 6.2 and the artificial ones, are among the recognized and commonly utilized networks in this context. More importantly, each possesses the corresponding ground truth, representing the optimal community structures. This allows us to evaluate the effectiveness of a community detection algorithm by measuring the similarities between the algorithmically derived communities and the optimal structures. For this evaluation, we employ the widely acknowledged performance metric *Normalized Mutual Information* (NMI), which we explain in Section 6.3.1.

Table 6.2: Networks under-study

ID	Network	n	m
1	Zachary's karate club [184]	34	78
2	Mexican Politicians [79]	35	117
3	Dolphin network [120]	62	159
4	Les Miserables [120]	77	254
5	p53 protein [84]	104	226
6	Books about U.S. politics [122]	105	441
7	American college football [80]	115	613
8	Citation graph drawing [124]	311	640
9	USAir97 [21]	332	2126
10	C. Elegans [33]	453	2025
11	Erdos collaboration [129]	472	1314
12	Electronic circuit [39]	512	819
13	Email-Eu-core [113]	986	16064
14	Amazon network [Mkhitaryan et al., 182]	334863	925872
15	LiveJournal social network [157, 182]	3997962	34681189

Subsequently, we focus exclusively on the proposed algorithm, delving into the role of its input parameters, namely k_1 and k_2, delineated in Algorithm 3, in improving community establishments.

It is worth mentioning that all the experiments are conducted on a computer system with a processor *Intel(R) Core i9-12900KF @ 3.2GHz, 16.6 Core(s)* and *128 GB* of *RAM*. Algorithms are implemented with C++, and CPLEX optimizer 12.9 is used for solving linear programming.

6.3.1 Normalized Mutual Information (NMI)

Whereas Normalized Mutual Information (NMI) [2], as defined in [50], stands as a widely acknowledged metric for assessing the similarity between clusters, it is particularly effective in measuring the agreement between optimal communities and those discovered by an algorithm. Consider a network G with n nodes, where $\mathcal{C}(\mathcal{A}) = C_1, \ldots, C_k$ represents the communities obtained by algorithm $\mathcal{A}$, and $\mathcal{C}' = C'_1, \ldots, C'_{k'}$ denotes the ground truth communities. The NMI value corresponding to algorithm $\mathcal{A}$ can be computed as follows:

$$NMI(\mathcal{A}) = \frac{-2\sum_{x=1}^{|\mathcal{C}|}\sum_{y=1}^{|\mathcal{C}'|}\frac{|C_x \cap C'_y|}{n} log(\frac{n|C_x \cap C'_y|)}{|C_x||C'_y|})}{\sum_{x=1}^{|\mathcal{C}|}\frac{|C_x|}{n} log(\frac{|C_x|}{n}) + \sum_{y=1}^{|\mathcal{C}'|}\frac{|C'_y|}{n} log(\frac{|C'_y|}{n})}. \tag{6.13}$$

When the identified communities precisely match the ground truth, NMI achieves its maximum value of one. Conversely, if there is no resemblance between the two sets, the NMI score drops to zero. Generally, a larger NMI value signifies a more precise and effective unveiling of community structures.

[2]We have additionally utilized the *Adjusted Rand Index* (ARI) [183], a recognized metric for community evaluation. However, as the results were consistent with those obtained using NMI, we have opted not to include them in order to maintain conciseness in the paper.

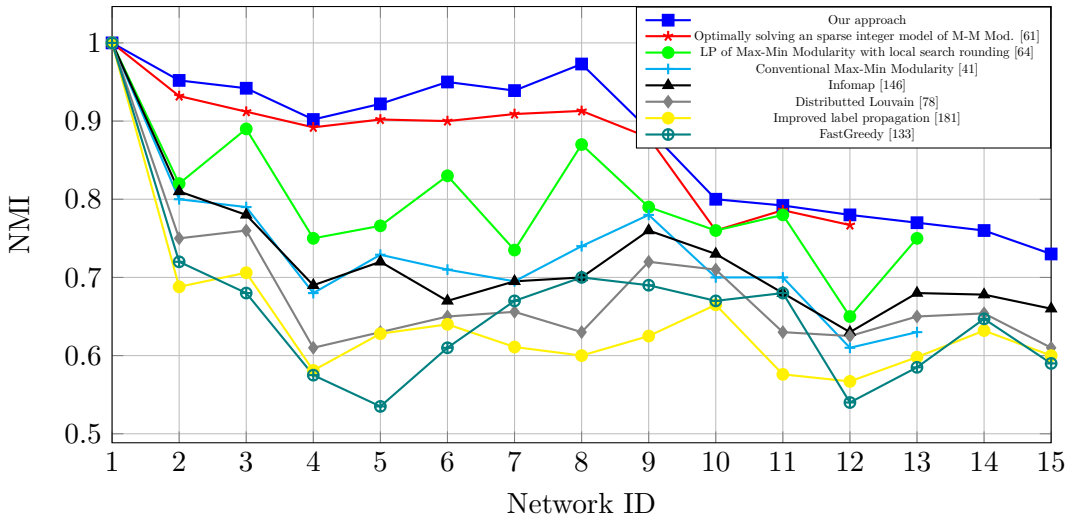


Figure 6.1: Comparison between NMI values achieved by our approach, the blue curve, and other understudy algorithms using real-world networks presented in Table 6.2.

Table 6.3: Used values for the algorithm's input parameters k_1 and k_2 associated with each of the real-world networks depicted in Table 6.2.

	Network ID														
Parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
k_1	0	0	0	2	2	2	2	3	4	4	4	4	5	5	5
k_2	2	2	3	3	3	3	5	6	8	8	11	15	18	25	29

6.3.2 Experiments

In our initial series of experiments, our objective is to assess the proposed algorithm in comparison to those presented in [63] and [61] using the 15 real-world networks. Figure 6.1 portrays the computed NMI values for each methodology. To offer a more comprehensive perspective on the advancements introduced by each algorithm, the results from the competitive algorithms utilized in [63, 61] have also been depicted in the figure.

While the superiority of our proposed algorithm against all other methodologies across various real-world benchmarks is evident, it is crucial to highlight the relatively small values of the parameters k_1 and k_2 required to achieve these results. Table 6.3 outlines the used parameter values. Whereas the selection of k_1 can be made relatively freely, often contingent on the network size, the utilized values of k_2, here, represent the minimum number of iterations required to solve the corresponding (IPs-MM) with all constraints satisfied.

The small values of these parameters affirm, firstly, the existence of high-quality initial communities that necessitate minimal restructuring. Secondly, they substantiate the claim that all constraints of (IPs-MM) are swiftly satisfied in the optimal solution to the model, particularly within the initial iterations of the applied row generation technique. This revelation not only bodes well for achieving refined communities but also ensures a reasonable timeframe for the algorithm. Bring to mind that our baseline methods [61] and [64] exhibit limitations in producing results within a reasonable timeframe, particularly when applied to relatively large networks. Specifically, the former experiences complete failure when confronted with networks 13, 14, and 15, while the latter collapses when applied to networks 14 and 15 within a reasonable timeframe.

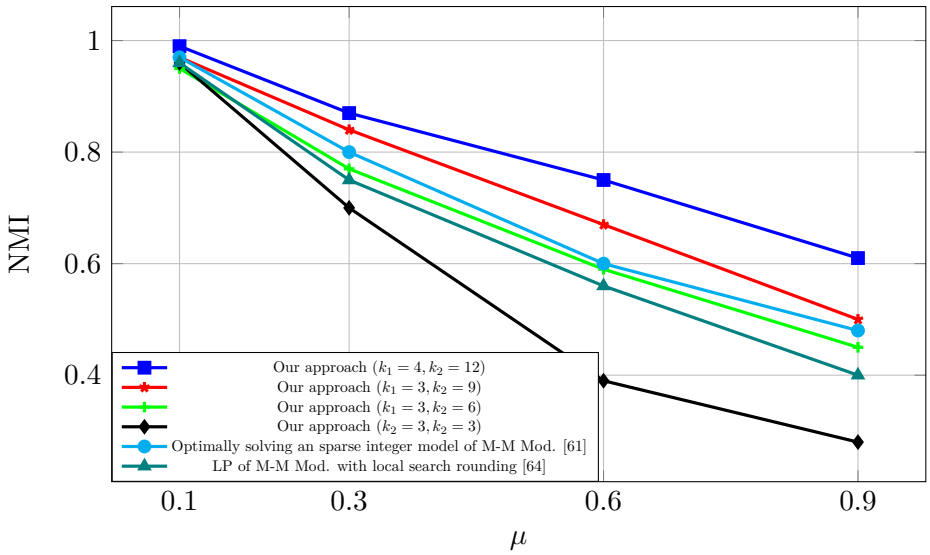


Figure 6.2: Comparison between NMI ranks of each algorithm applying to the LFR synthetic graphs, generated by considering four different mixing parameters μ.

By the same token, this breakthrough can be extrapolated to artificial networks. As mentioned, the generation process of the artificial graphs we take into account relies on what has been introduced in [111]. These networks adhere to power-law distributions for both the degree and community size, with user-defined exponents γ and β, respectively. Additionally, the mixing parameter μ dictates that each vertex in the graph should share a fraction of $1-\mu$ of its links with other nodes in its community and a fraction of μ with other nodes in the network. The parameter μ effectively controls the noise, with higher values leading to less distinct community structures. Thus, a higher μ poses challenges for community detection algorithms in identifying optimal communities [60]. In this study, we utilize four different LFR graphs with distinct mixing parameters μ ($\mu = 0.1, 0.3, 0.6$, and 0.9), each comprising 1000 nodes and 5 communities. All other parameters are set to default values as detailed in [111].

We employ these benchmarks for a dual purpose: to assess the precision of our algorithms in comparison to other competing methods and to investigate the impact of the algorithm's input parameters, namely k_1 and k_2. Figure 6.2 illustrates the resultant NMI values corresponding to each scenario.

As can be discerned, in scenarios $\mu = 0.1$ and $\mu = 0.3$, which indicate less intricate community interconnections in the synthesized networks, all algorithms more or less effortlessly detected structures closely aligned with the ground truths. Particularly noteworthy is the superior performance of the proposed approach, even with relatively small values of k_1 and k_2 (depicted by the blue curve). The selection of k_1 underscores the algorithm's ability to bypass unnecessary iterations in the community revision step to sufficiently improve the complementary graph. Meanwhile, the small choice of k_2 demonstrates the algorithm's rapid convergence toward the optimal solution by satisfying all the constraints of (IPs-MM).

In contrast, as the level of noise within the network's structure increases, notably with an elevated value of μ, especially for $\mu = 0.9$, all competing algorithms encounter difficulties in capturing the optimal community structure. Intriguingly, the proposed method also faces challenges in this scenario when k_1 and k_2 are not sufficiently large, specifically for the black curve, wherein $k_1 = k_2 = 3$, the proposed algorithm exhibits a complete degradation against other techniques.

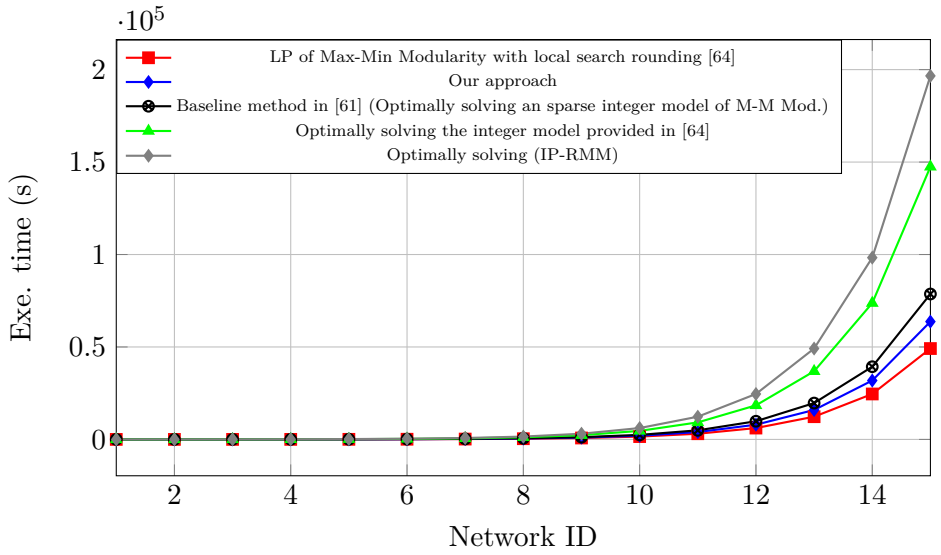


Figure 6.3: Execution time associated with each algorithm applied to the real-world networks presented in Table 6.2.

From a time complexity perspective, as illustrated in Figure 6.3, our algorithm operates significantly faster compared to other methods. In previous research [61], it was demonstrated that our baseline model (depicted by the black curve in Figure 6.3) holds an advantage over all competing algorithms except for the one introduced in [64]. The latter solves the LP relaxation version of the model and employs a rounding algorithm to determine the integer solution and, consequently, the community structures (red curve in Figure 6.3). Notably, our augmented method (the blue curve) not only surpasses the baseline model but also closely approaches a fast heuristic but not very accurate approach presented in [64].

We state that, as expected, not only does our proposed augmented method outperform our baseline approach [61] in terms of time complexity (as depicted in Figure 6.3), but it also yields higher quality communities that closely align with the ground truths (as depicted in Figure 6.1 and 6.2). This is achieved through a heuristic design for enhancing the Max-Min Modularity complementary graph and expediting solving the sparse yet equivalent integer counterpart we introduced for the problem. As a final note regarding the parameters selection, as networks increase in size, a smaller value for k_1 becomes more reasonable. This is attributed primarily to the high quality of the initial communities obtained in the first phase of the heuristic method and, secondly, to the time efficiency of this phase, as elucidated in Section 6.2.1. As for k_2, although the sparse model (IPs-MM) expedites the solving process, opting for a larger value may not be necessary, especially when dealing with massive graphs. This is particularly evident and plausible given the typically swift convergence of the model, wherein all constraints are satisfied by an optimal solution obtained in the early iterations.

6.4 Conclusion

Communities and community detection in complex networks are crucial for uncovering hidden structures and relationships, which enhance our understanding of various real-world systems, from communication networks to biological systems. By identifying these communities, we can improve strategies for targeted interventions, optimize network functionalities, and predict system behaviors more accurately. This paper introduced a novel approach to community detection in

complex networks, combining the strengths of a connectivity-based metric and the Max-Min Modularity. The synergistic incorporation of these two effective community quality functions resulted in the development of a robust algorithm. By leveraging the connectivity-based metric incorporated into a heuristic algorithm, a novel complementary graph associated with Max-Min Modularity was established, demonstrating enhanced effectiveness in community detection. The analytical insights derived substantiated the time efficiency of the employed heuristic method, addressing the time complexity concerns. The formulation of community detection as an integer programming problem and the presentation of a more compact yet equivalent model contributed to the optimization of computational resources. The proposed sparse model significantly reduced the number of variables and constraints. Furthermore, depending on the network size, the paper introduced a systematic procedure to balance between optimally solving the sparse model and relying on the heuristic method to facilitate community detection in very large networks. Through comprehensive experiments, the success of the algorithm was showcased, highlighting its efficiency in detecting communities.

Chapter 7

Conclusion

This thesis developed, analysed, and evaluated combinatorial optimization methods for two complementary classes of network problems: (i) the design and operation of hybrid urban waste-collection systems that combine conventional truck routing with automated pneumatic networks, and (ii) the discovery of meaningful community structure in large complex networks through optimization-driven formulations. Across these two themes, a common methodological foundation emerged: problems were formulated as integer or mixed-integer programming models; scalable exact, heuristic, and matheuristic strategies were designed using decomposition, column generation, and local search; and all methods were validated through extensive computational experiments on benchmark datasets and real-world instances. The resulting contributions advance both applied logistics optimization and theoretical tools for network analysis. This chapter summarizes the key findings, synthesizes overarching insights, and outlines promising avenues for future work.

Summary of Contributions

Chapter 2 introduced a comprehensive mathematical model for the *Combined Waste Collection Problem* (CWCP), which integrates an Automated Waste Collection System (AWCS) into a traditional point-to-point truck-based collection framework. The chapter proposed a holistic solution approach based on a BPC-based matheuristic, in which the AWCS-related subproblems are solved through heuristic column-generation algorithm. This integrated method simultaneously determines, for each collection point, whether it is served by the AWCS or by a truck and constructs both the AWCS tree and the corresponding truck routes. A comparison with an exact branch-and-price algorithm on small benchmark instances showed that the matheuristic consistently produces near-optimal solutions, with deviations from optimality below one percent, and often outperforms the exact method within a one-hour time limit due to the computational bottleneck posed by exact tree generation. Extensive experiments on 45 larger CVRP-based instances, ranging in size from 100 to 308 nodes, further confirmed the robustness and scalability of the approach, yielding feasible solutions across all scenarios and demonstrating that AWCS installation is particularly beneficial when demands are high relative to vehicle capacity. The chapter also presented a Vienna-based case study, showing that strategic placement of the AWCS facility can reduce total system cost by approximately five percent and that the AWCS tends to operate at or near capacity in high-demand settings.

Chapter 3 extended the CWCP by incorporating a realistic operational constraint: a hop limit on each branch of the AWCS pipeline network. By explicitly modeling this restriction, the AWCS tree-design component becomes a variant of the *Steiner Tree Problem with Revenues and Hop Constraints* (STPRH), thereby capturing practical pneumatic-system limitations such as pressure loss along the pipeline. To solve the CWCP with a hop limit exactly, the chapter developed four branch-price-and-cut algorithms based on two alternative decomposition strategies: a full decomposition that treats routing and AWCS tree generation independently in the pricing problems, and a partial decomposition that embeds the tree structure directly in the master problem while applying column generation only to the routing component. In addition, two state-of-the-art STPRH formulations, the partial-ordering and layer-assignment models, were incorporated and adapted to this setting. Computational experiments on benchmark data sets revealed that decomposition choice dominates computational behaviour: full decomposition

provides tighter bounds but struggles with larger instances due to the high cost of solving the tree-pricing problem, whereas partial decomposition scales reliably and consistently returns high-quality solutions. Across all metrics, the partial-ordering formulation outperformed its layer-assignment counterpart, with the partial decomposition and partial-ordering formulation achieving the strongest overall performance by producing best-known bounds on more than 95% of instances. The experiments further showed that instance characteristics such as hop limits, subsystem cost ratios, and tree capacities strongly influence difficulty, and that modelling enhancements such as subtour elimination constraints substantially improve algorithmic efficiency.

Chapter 4 focused on the analysis of large-scale, complex networks through the lens of community detection, the task of identifying densely connected groups of vertices that serve as structural or functional units. The chapter introduced a new connectivity-based community-quality metric that integrates structural cohesion with centrality-driven notions of influence by explicitly modeling the impact of each community's most influential vertex. This metric simultaneously minimizes inter-community connectivity and community radius, thereby favouring well-separated and internally compact communities. Building on this foundation, the chapter developed *TSCDA*, a dynamic two-stage community-discovery algorithm in which a set of highly influential and mutually distant nodes is first selected to generate high-quality initial partitions, followed by a refinement phase based on local search. Extensive experiments on real-world and synthetic networks, including LFR benchmarks and widely studied social and biological graphs, demonstrated that TSCDA achieves community partitions that compare favourably with state-of-the-art methods in terms of normalized mutual information and adjusted Rand index, while scaling efficiently to networks with hundreds of thousands of vertices.

Chapter 5 examined community detection from the perspective of *Max-Min Modularity*, a robust variant of the classical modularity measure designed to mitigate well-known issues such as the resolution limit and sensitivity to small perturbations. Building on existing integer and linear programming formulations, the chapter proposed a substantially more compact integer model that drastically reduces the number of variables and constraints while provably preserving optimality. The solution approach combines an intelligently selected initial feasible solution, constructed using the first stage of TSCDA from Chapter 4, with a row-and-column generation scheme embedded in a branch-and-bound framework. Computational experiments on twelve widely used real-world networks with established community structure showed that the resulting exact method consistently attains higher Max-Min Modularity values and produces partitions that align more closely with ground-truth communities than existing heuristic and exact approaches. Moreover, the compact formulation and tailored strategy yield significant computational savings, making high-quality modularity optimization feasible for medium-sized networks previously beyond the reach of exact models.

Chapter 6 developed a hybrid community-detection framework that unifies the connectivity-based metric of Chapter 4 with the compact Max-Min Modularity formulation of Chapter 5. The method begins by applying TSCDA to generate high-quality initial communities and then constructs a *complementary graph* that more accurately captures the absence of meaningful interactions between non-adjacent node pairs. On this enriched representation, the chapter reformulates Max-Min Modularity using a sparse equivalent integer model that retains optimality while substantially reducing memory requirements and computational effort. The resulting hybrid optimization procedure adaptively balances exact and heuristic steps via problem-specific row generation, enabling near-optimal solutions for large graphs at a fraction of the cost of purely exact algorithms, while preserving strong guarantees for small and medium instances. Comprehensive experiments on a broad collection of synthetic and real-world networks confirmed that this approach consistently outperforms classical modularity heuristics and modern optimization-based community-detection methods, achieving superior partitions while remaining scalable to networks with tens of thousands of vertices.

Future Research Directions

The findings of this thesis suggest several promising lines of investigation:

- **Richer CWCP models.** Incorporating temporal dynamics, stochastic demand, time windows, and multi-objective trade-offs (such as environmental impacts) would improve the operational fidelity of CWCP models.
- **Scalable methodological side.** Integrating advanced acceleration techniques, such as dynamic cut selection and dual stabilization, could further enhance computational scalability. Alternative schemes, including column-and-row generation approaches, may also improve efficiency for large-scale data sets.
- **Richer structural modelling in complex networks.** Incorporate additional node and edge properties, such as attributes, temporal information, and multilayer structure, to enhance community-detection accuracy and expressiveness.
- **Scalability and robustness in complex networks.** Extend the framework to handle extremely dense and complex networks (e.g., biological or astronomical networks) while improving approximation bounds and ensuring robustness to Byzantine behaviour and message loss.
- **Enhancing optimization algorithms with neuromorphic computation.** Neuromorphic Computing has emerged as a promising paradigm to address the increasing demands of intelligent systems for low energy consumption, scalability, and real-time responsiveness. They might enhance the efficiency of each algorithm mentioned in this thesis.

Closing Remarks

Taken together, the contributions of this thesis demonstrate the versatility and power of combinatorial optimization as a framework for both *structure design* in engineered systems and *structure discovery* in complex networks. By integrating exact, matheuristic, and heuristic strategies, the thesis provides practical tools and theoretical insights for tackling computationally demanding network problems. The methods developed here offer a foundation for further advances in both operational research and network science.

Bibliography

[1] Abd Manaf, L., Samah, M. A. A., and Zukki, N. I. M. (2009). Municipal solid waste management in Malaysia: Practices and challenges. *Waste Management*, 29(11):2902–2906.

[2] Achterberg, T. (2007). *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin.

[3] Agarwal, G. and Kempe, D. (2008). Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418.

[4] Ahmed, F. and Abulaish, M. (2013). A generic statistical approach for spam detection in online social networks. *Computer Communications*, 36(10-11):1120–1129.

[5] Ai, J., Liu, Y., Su, Z., Zhang, H., and Zhao, F. (2019). Link prediction in recommender systems based on multi-factor network modeling and community detection. *EPL (Europhysics Letters)*, 126(3):38003.

[6] Akhtar, M., Hannan, M. A., Begum, R. A., Basri, H., and Scavino, E. (2017). Backtracking search algorithm in cvrp models for efficient solid waste collection and route optimization. *Waste Management*, 61:117–128.

[7] Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., and Liberti, L. (2010). Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112.

[8] Aloise, D., Caporossi, G., Hansen, P., Liberti, L., Perron, S., and Ruiz, M. (2012). Modularity maximization in networks by variable neighborhood search. *Graph Partitioning and Graph Clustering*, 588:113.

[9] Alozie, G. U., Arulselvan, A., Akartunalı, K., and Pasiliao Jr, E. L. (2021). Efficient methods for the distance-based critical node detection problem in complex networks. *Computers & Operations Research*, 131:105254.

[10] Aparicio, S., Villazón-Terrazas, J., and Álvarez, G. (2015). A model for scale-free networks: application to twitter. *Entropy*, 17(8):5848–5867.

[11] Arenas, A., Fernandez, A., Fortunato, S., and Gomez, S. (2008). Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224001.

[12] Artiles, J., Gonzalo, J., and Sekine, S. (2007). The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the fourth international workshop on semantic evaluations (semeval-2007)*, pages 64–69.

[13] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. (2004). Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562.

[14] Atay, Y., Koc, I., Babaoglu, I., and Kodaz, H. (2017). Community detection from biological and social networks: A comparative analysis of metaheuristic algorithms. *Applied Soft Computing*, 50:194–211.

[15] Augerat, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. PhD thesis, InstitutNational Polytechnique de Grenoble – INPG.

[16] Aviyente, S. (2021). A signal processing perspective to community detection in dynamic networks. *Digital Signal Processing*, 119:103192.

[17] Azaouzi, M., Rhouma, D., and Romdhane, L. B. (2019). Community detection in large-scale social networks: state-of-the-art and future directions. *Social Network Analysis and Mining*, 9(1):1–32.

[18] Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283.

[19] Barabási, A.-L. and Bonabeau, E. (2003). Scale-free networks. *Scientific american*, 288(5):60–69.

[20] Barghi, A. R., Ferdowsi, A., and Abhari, A. (2018). Musical preferences prediction by classification algorithm. In *Proceedings of the Communications and Networking Symposium*, pages 1–12.

[21] Batagelj, V. and Mrvar, A. (2009). Pajek datasets (2006).

[22] Bedi, P. and Sharma, C. (2016). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(3):115–135.

[23] Béjar, R., Fernàndez, C., Manyà, F., Mateu, C., and Sole-Mauri, F. (2012a). Optimizing energy consumption in automated vacuum waste collection systems. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, pages 291–298. IEEE.

[24] Béjar, R., Fernández, C., Mateu, C., Manya, F., Sole-Mauri, F., and Vidal, D. (2012b). The automated vacuum waste collection optimization problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 264–266.

[25] Beliën, J. and Demeulemeester, E. (2007). On the trade-off between staff-decomposed and activity-decomposed column generation for a staff scheduling problem. *Annals of Operations Research*, 155(1):289–307.

[26] Beltrami, E. J. and Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94.

[27] Bienstock, D., Goemans, M. X., Simchi-Levi, D., and Williamson, D. (1993). A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59(1?3):413–420.

[28] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

[29] Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, 49(2):369–383.

[30] Bohlin, L., Edler, D., Lancichinetti, A., and Rosvall, M. (2014). Community detection and visualization of networks with the map equation framework. In *Measuring scholarly impact*, pages 3–34. Springer.

[31] Bommarito II, M. J., Katz, D. M., Zelner, J. L., and Fowler, J. H. (2010). Distance measures for dynamic citation networks. *Physica A: Statistical Mechanics and its Applications*, 389(19):4201–4208.

[32] Boudebza, S., Cazabet, R., Azouaou, F., and Nouali, O. (2018). Olcpm: An online framework for detecting overlapping communities in dynamic social networks. *Computer Communications*, 123:36–51.

[33] Cangelosi, A. and Parisi, D. (1997). A neural network model of caenorhabditis elegans: the circuit of touch sensitivity. *Neural processing letters*, 6(3):91–98.

[34] Casciaro, T. and Lobo, M. S. (2005). Competent jerks, lovable fools, and the formation of social networks. *Harvard business review*, 83(6):92–99.

[35] Cazabet, R., Rossetti, G., and Amblard, F. (2017). Dynamic community detection.

[36] Ceselli, A., Damiani, M. L., Righini, G., and Valorsi, D. (2018). Mathematical programming algorithms for spatial cloaking. *INFORMS Journal on Computing*, 30(4):710–723.

[37] Chàfer, M., Sole-Mauri, F., Solé, A., Boer, D., and Cabeza, L. F. (2019). Life cycle assessment (lca) of a pneumatic municipal waste collection system compared to traditional truck collection. sensitivity study of the influence of the energy source. *Journal of Cleaner Production*, 231:1122–1135.

[38] Chakraborty, T., Dalmia, A., Mukherjee, A., and Ganguly, N. (2017). Metrics for community analysis: A survey. *ACM Computing Surveys (CSUR)*, 50(4):54.

[39] Chand, S. and Mehta, S. (2017). Community detection using nature inspired algorithm. In *Hybrid Intelligence for Social Networks*, pages 47–76. Springer.

[40] Cheikh, S., Sara, B., and Sara, Z. (2020). A hybrid heuristic community detection approach. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–7. IEEE.

[41] Chen, J., Zaïane, O. R., and Goebel, R. (2009). Detecting communities in social networks using max-min modularity. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 978–989. SIAM.

[42] Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

[43] Comellas, F. and Sampels, M. (2002). Deterministic small-world networks. *Physica A: Statistical Mechanics and its Applications*, 309(1-2):231–235.

[44] Corberán, Á. and Laporte, G., editors (2014). *Arc Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[45] Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.

[46] Cordone, R. and Trubian, M. (2006). An exact algorithm for the node weighted Steiner tree problem. *4OR*, 4:124–144.

[47] Costa, A. M., Cordeau, J.-F., and Laporte, G. (2008). Fast heuristics for the steiner tree problem with revenues, budget and hop constraints. *European Journal of Operational Research*, 190(1):68–78.

[48] Costa, A. M., Cordeau, J.-F., and Laporte, G. (2009). Models and branch-and-cut algorithms for the steiner tree problem with revenues, budget and hop constraints. *Networks: An International Journal*, 53(2):141–159.

[49] Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985.

[50] Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008.

[51] Dao, V. L., Bothorel, C., and Lenca, P. (2020). Community structure: A comparative evaluation of community detection methods. *Network Science*, 8(1):1–41.

[52] Daskin, M. S. and Maass, K. L. (2015). The p-median problem. In *Location Science*, pages 21–45. Springer.

[53] DehghanChenary, M., Hartl, R. F., Irnich, S., and Tilk, C. (2025). On combining conventional point-to-point and automated waste collection systems. *Networks*.

[54] Desrosiers, J., Lübbecke, M., Desaulniers, G., and Gauthier, J. B. (2024). *Branch-and-Price*. ReseachGate.

[55] Devi, J. C. and Poovammal, E. (2016). An analysis of overlapping community detection algorithms in social networks. *Procedia Computer Science*, 89:349–358.

[56] Dinh, T. N. and Thai, M. T. (2011). Finding community structure with performance guarantees in complex networks. *arXiv preprint arXiv:1108.4034*.

[57] Duin, C. W. and Volgenant, A. (1987). Some generalizations of the steiner problem in graphs. *Networks*, 17(3):353–364.

[58] Ebel, H., Mielsch, L.-I., and Bornholdt, S. (2002). Scale-free topology of e-mail networks. *Physical review E*, 66(3):035103.

[59] Ferdowsi, A. (2022). An integer programming approach reinforced by a message-passing procedure for detecting dense attributed subgraphs. In *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 569–576. IEEE.

[60] Ferdowsi, A. and Abhari, A. (2020). Generating high-quality synthetic graphs for community detection in social networks. In *Proceedings of the 2020 Spring Simulation Conference*, pages 1–10.

[61] Ferdowsi, A. and DehghanChenary, M. (2023). Toward an optimal solution to the network partitioning problem. In *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 111–117. IEEE.

[62] Ferdowsi, A. and DehghanChenary, M. (2024). Gain and pain in graph partitioning: Finding accurate communities in complex networks. *Algorithms*, 17(6):226.

[63] Ferdowsi, A., DehghanChenary, M., and Khanteymoori, A. (2022). Tscda: a dynamic two-stage community discovery approach. *Social Network Analysis and Mining*, 12(1):46.

[64] Ferdowsi, A. and Khanteymoori, A. (2021). Discovering communities in networks: A linear programming approach using max-min modularity. In *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 329–335. IEEE.

[65] Fernández, C., Manyà, F., Mateu, C., and Sole-Mauri, F. (2014). Modeling energy consumption in automated vacuum waste collection systems. *Environmental Modelling & Software*, 56:63–73.

[66] Fernàndez Camon, C., Manyà Serres, F., Mateu Piñol, C., and Solé Mauri, F. (2015). Approximate dynamic programming for automated vacuum waste collection systems. *Environmental Modelling & Software*, 67:128–137.

[67] Flake, G. W., Lawrence, S., and Giles, C. L. (2000). Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160.

[68] Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3-5):75–174.

[69] Fortunato, S. and Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the national academy of sciences*, 104(1):36–41.

[70] Fortunato, S. and Hric, D. (2016). Community detection in networks: A user guide. *Physics reports*, 659:1–44.

[71] Freitas, L. M. and Carneiro, M. G. (2019). Community detection to invariant pattern clustering in images. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 610–615. IEEE.

[72] Fu, Z.-H. and Hao, J.-K. (2014). Breakout local search for the steiner tree problem with revenue, budget and hop constraints. *European Journal of Operational Research*, 232(1):209–220.

[73] Fu, Z.-H. and Hao, J.-K. (2015). Dynamic programming driven memetic search for the steiner tree problem with revenues, budget, and hop constraints. *INFORMS Journal on Computing*, 27(2):221–237.

[74] Gallier, J. (2016). Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey. *arXiv preprint arXiv:1601.04692*.

[75] Gao, S., Chen, A., Rahmani, A., Jarada, T., Alhajj, R., Demetrick, D., and Zeng, J. (2013). Mcf: A tool to find multi-scale community profiles in biological networks. *Computer methods and programs in biomedicine*, 112(3):665–672.

[76] Gao, Y., Yu, X., and Zhang, H. (2021). Overlapping community detection by constrained personalized pagerank. *Expert Systems with Applications*, 173:114682.

[77] Gawande, N., Ghosh, S., Halappanavar, M., Tumeo, A., and Kalyanaraman, A. (2022). Towards scaling community detection on distributed-memory heterogeneous systems. *Parallel Computing*, 111:102898.

[78] Ghosh, S., Halappanavar, M., Tumeo, A., Kalyanaraman, A., Lu, H., Chavarria-Miranda, D., Khan, A., and Gebremedhin, A. (2018). Distributed louvain algorithm for graph community detection. In *2018 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 885–895. IEEE.

[79] Gil-Mendieta, J. and Schmidt, S. (1996). The political network in mexico. *Social Networks*, 18(4):355–381.

[80] Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826.

[81] Good, B. H., De Montjoye, Y.-A., and Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106.

[82] Gouveia, L. (1998). Using variable redefinition for computing lower bounds for minimum spanning and steiner trees with hop constraints. *INFORMS Journal on Computing*, 10(2):180–188.

[83] Grötschel, M. and Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1-3):59–96.

[84] Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F., and Arenas, A. (2003). Self-similar community structure in a network of human interactions. *Physical review E*, 68(6):065103.

[85] Gupta, A. and Tangwongsan, K. (2008). Simpler analyses of local search algorithms for facility location. *arXiv preprint arXiv:0809.2554*.

[86] Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159.

[87] Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467.

[88] Hernandez, F., Feillet, D., Giroudeau, R., and Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, 249(2):551–559.

[89] Hess, C., Dragomir, A. G., Doerner, K. F., and Vigo, D. (2024). Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research*, 32(2):399–434.

[90] Hu, L., Zhang, J., Pan, X., Yan, H., and You, Z.-H. (2021). Hiscf: leveraging higher-order structures for clustering analysis in biological networks. *Bioinformatics*, 37(4):542–550.

[91] Huang, L., Chao, H.-Y., and Xie, Q. (2020). Mumod: A micro-unit connection approach for hybrid-order community detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 107–114.

[92] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

[93] Hui, P., Yoneki, E., Chan, S. Y., and Crowcroft, J. (2007). Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, pages 1–8.

[94] Iriarte, A., Gabarrell, X., and Rieradevall, J. (2009). LCA of selective waste collection systems in dense urban areas. *Waste Management*, 29(2):903–914.

[95] Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148.

[96] Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, chapter 2, pages 33–65. Springer.

[97] Irnich, S., Toth, P., and Vigo, D. (2014). The family of vehicle routing problems. In Vigo, D. and Toth, P., editors, *Vehicle Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, chapter Chapter 1, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[98] Jabrayilov, A. (2020). On the hop-constrained steiner tree problems. *arXiv preprint arXiv:2007.07405*.

[99] Jabrayilov, A. and Mutzel, P. (2019). A new integer linear program for the steiner tree problem with revenues, budget and hop constraints. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 107–116. SIAM.

[100] Jalali, M., Tsotsalas, M., and Wöll, C. (2022). Mofsocialnet: Exploiting metal-organic framework relationships via social network analysis. *Nanomaterials*, 12(4):704.

[101] Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

[102] Jiang, L., Shi, L., Liu, L., Yao, J., and Yousuf, M. A. (2019). User interest community detection on social media using collaborative filtering. *Wireless Networks*, pages 1–7.

[103] Kamga, C., Miller, B., Spertus, J., Douglass, L., Ross, B., and Eickemeyer, P. (2013a). Eliminating trucks on roosevelt island for the collection of wastes. Technical report, University Transportation Research Center, City College of New York, New York, NY.

[104] Kamga, C., Miller, B., Spertus, J., Douglass, L., Ross, B., Eickemeyer, P., et al. (2013b). A study of the feasibility of pneumatic transport of municipal solid waste and recyclables in Manhattan using existing transportation infrastructure. Technical report, New York State Energy Research and Development Authority.

[105] Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307.

[106] Kim, J. and Hastak, M. (2018). Social network analysis: Characteristics of online social networks after a disaster. *International Journal of Information Management*, 38(1):86–96.

[107] Kinobe, J. R., Bosona, T., Gebresenbet, G., Niwagaba, C. B., and Vinnerås, B. (2015). Optimization of waste collection and disposal in kampala city. *Habitat International*, 49:126–137.

[108] Kogler, T. (2007). Waste collection. *ISWA Working Group on Collection and Transportation Technology*.

[109] Krioukov, D., Kitsak, M., Sinkovits, R. S., Rideout, D., Meyer, D., and Boguñá, M. (2012). Network cosmology. *Scientific reports*, 2:793.

[110] Kumar, S. and Hanot, R. (2020). Community detection algorithms in complex networks: A survey. In *International Symposium on Signal Processing and Intelligent Recognition Systems*, pages 202–215. Springer.

[111] Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110.

[112] Layeb, S. B., Hajri, I., and Haouari, M. (2013). Solving the steiner tree problem with revenues, budget and hop constraints to optimality. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pages 1–4. IEEE.

[113] Leskovec, J., Kleinberg, J., and Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2.

[114] Leskovec, J., Lang, K. J., and Mahoney, M. (2010). Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM.

[115] Li, Z., Zhang, S., Wang, R.-S., Zhang, X.-S., and Chen, L. (2008). Quantitative function for community detection. *Physical review E*, 77(3):036109.

[116] Linares, O. A., Botelho, G. M., Rodrigues, F. A., and Neto, J. B. (2017). Segmentation of large images based on super-pixels and community detection in graphs. *IET Image Processing*, 11(12):1219–1228.

[117] Liu, X., Cheng, H.-M., and Zhang, Z.-Y. (2019). Evaluation of community detection methods. *IEEE Transactions on Knowledge and Data Engineering*, 32(9):1736–1746.

[118] Ljubi?, I., Weiskircher, R., Pferschy, U., Klau, G. W., Mutzel, P., and Fischetti, M. (2005). An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. *Mathematical Programming*, 105(2?3):427–449.

[119] Ljubić, I. (2021). Solving steiner trees: Recent advances, challenges, and perspectives. *Networks*, 77(2):177–204.

[120] Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., and Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405.

[121] Magnani, M., Hanteer, O., Interdonato, R., Rossi, L., and Tagarelli, A. (2021). Community detection in multiplex networks. *ACM Computing Surveys (CSUR)*, 54(3):1–35.

[122] Mahajan, A. and Kaur, M. (2016). Various approaches of community detection in complex networks: a glance. *International Journal of Information Technology and Computer Science (IJITCS)*, 8(35).

[123] Malliaros, F. D. and Vazirgiannis, M. (2013). Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142.

[124] Meghanathan, N. (2016). A greedy algorithm for neighborhood overlap-based community detection. *Algorithms*, 9(1):8.

[125] Miller, B., Spertus, J., and Kamga, C. (2014). Costs and benefits of pneumatic collection in three specific new york city cases. *Waste Management*, 34(11):1957–1966.

[126] Miyauchi, A. and Miyamoto, Y. (2013). Computing an upper bound of modularity. *The European Physical Journal B*, 86(7):302.

[127] Miyauchi, A. and Sukegawa, N. (2015). Redundant constraints in the standard formulation for the clique partitioning problem. *Optimization Letters*, 9(1):199–207.

[Mkhitaryan et al.] Mkhitaryan, K., Mothe, J., and Haroutunian, M. Detecting communities from networks: Comparison of algorithms on real and synthetic networks.

[129] Mrvar, A. and Batagelj, V. (2020). *Pajek: Programs for Analysis and Visualization of Very Large Networks: Reference Manual: List of Commands with Short Explanation Version 5.10*. A. Mrvar.

[130] Nagy, G. and Salhi, S. (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672.

[131] Nakou, D., Benardos, A., and Kaliampakos, D. (2014). Assessing the financial and environmental performance of underground automated vacuum waste collection systems. *Tunnelling and Underground Space Technology*, 41:263–271.

[132] Newman, M. E. (2004a). Analysis of weighted networks. *Physical review E*, 70(5):056131.

[133] Newman, M. E. (2004b). Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133.

[134] Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582.

[135] Newman, M. E., Barabási, A.-L. E., and Watts, D. J. (2006). *The structure and dynamics of networks.* Princeton university press.

[136] Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113.

[137] Oh, J. H., Lee, E.-J., Oh, J. I., Kim, J.-O., and Jang, A. (2016). A comparative study on per capita waste generation according to a waste collecting system in Korea. *Environmental Science and Pollution Research*, 23:7074–7080.

[138] Orman, G. K. and Labatut, V. (2009). A comparison of community detection algorithms on artificial networks. In *International conference on discovery science*, pages 242–256. Springer.

[139] Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9:61–100.

[140] Perliger, A. and Pedahzur, A. (2011). Social network analysis in the study of terrorism and political violence. *PS: Political Science & Politics*, 44(1):45–50.

[141] Punkkinen, H., Merta, E., Teerioja, N., Moliis, K., and Kuvaja, E. (2012). Environmental sustainability comparison of a hypothetical pneumatic waste collection system and a door-to-door system. *Waste Management*, 32(10):1775–1781.

[142] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663.

[143] Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106.

[144] Ramakrishna, R. and Scaglione, A. (2021). Grid-graph signal processing (grid-gsp): A graph signal processing framework for the power grid. *IEEE Transactions on Signal Processing*, 69:2725–2739.

[145] Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273.

[146] Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

[147] Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, 9(1):1–17.

[148] Sahu, S. and Rani, T. S. (2022). A neighbour-similarity based community discovery algorithm. *Expert Systems with Applications*, 206:117822.

[149] Salhi, S. and Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156.

[150] Saxena, A., Ramaswamy, M., Beale, J., Marciniuk, D., and Smith, P. (2021). Striving for the united nations (un) sustainable development goals (SDGs): What will it take? *Discover Sustainability*, 2:1–14.

[151] Schaeffer, S. E. (2007). Graph clustering. *Computer science review*, 1(1):27–64.

[152] Segev, A. (1987). The node-weighted Steiner tree problem. *Networks*, 17(1):1–17.

[153] Shang, R., Liu, H., Jiao, L., and Esfahani, A. M. G. (2017). Community mining using three closely joint techniques based on community mutual membership and refinement strategy. *Applied Soft Computing*, 61:1060–1073.

[154] Shchur, O. and Günnemann, S. (2019). Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*.

[155] Shen, H.-W., Cheng, X.-Q., and Guo, J.-F. (2009). Quantifying and identifying the overlapping community structure in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(07):P07042.

[156] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107.

[157] Shi, X., Lu, H., He, Y., and He, S. (2015). Community detection in social network with pairwisely constrained symmetric non-negative matrix factorization. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 541–546. IEEE.

[158] Sinnl, M. (2011). Branch-and-price for the steiner tree problem with revenues, budget and hop constraints. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms.

[159] Sinnl, M. and Ljubić, I. (2016). A node-based layered graph approach for the steiner tree problem with revenues, budget and hop-constraints. *Mathematical Programming Computation*, 8:461–490.

[160] Sobolevsky, S., Campari, R., Belyi, A., and Ratti, C. (2014). General optimization technique for high-quality community detection in complex networks. *Physical Review E*, 90(1):012811.

[161] Sun, Z., Sun, Y., Chang, X., Wang, Q., Yan, X., Pan, Z., and Li, Z.-p. (2020). Community detection based on the matthew effect. *Knowledge-Based Systems*, 205:106256.

[162] Teerioja, N., Moliis, K., Kuvaja, E., Ollikainen, M., Punkkinen, H., and Merta, E. (2012). Pneumatic vs. door-to-door waste collection systems in existing urban areas: a comparison of economic performance. *Waste Management*, 32(10):1782–1791.

[163] Tilk, C. and Irnich, S. (2018). Combined column-and-row-generation for the optimal communication spanning tree problem. *Computers & Operations Research*, 93:113–122.

[164] Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, 261(2):530–539.

[165] Tonissen, D. D., van den Akker, J. M., and Hoogeveen, J. A. (2017). Column generation strategies and decomposition approaches for the two?stage stochastic multiple knapsack problem. *Computers & Operations Research*, 83:125–139.

[166] Tremblay, N. and Borgnat, P. (2014). Graph wavelets for multiscale community mining. *IEEE Transactions on Signal Processing*, 62(20):5227–5239.

[167] Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858.

[168] UN (2022). The sustainable development goals: Report 2022. Technical report, United Nations, Department of Economic and Social Affairs.

[169] Usón, A. A., Ferreira, G., Vásquez, D. Z., Bribián, I. Z., and Sastresa, E. L. (2013). Environmental-benefit analysis of two urban waste collection systems. *Science of the Total Environment*, 463:72–77.

[170] Van den Akker, J., Bouman, P. C., Hoogeveen, J., and Tönissen, D. D. (2016). Decomposition approaches for recoverable robust optimization problems. *European Journal of Operational Research*, 251(3):739–750.

[171] Varsha, K. and Patil, K. K. (2020). An overview of community detection algorithms in social networks. In *2020 International Conference on Inventive Computation Technologies (ICICT)*, pages 121–126. IEEE.

[172] Vigo, D. and Toth, P., editors (2014). *Vehicle Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[173] Volland, J., Fügener, A., and Brunner, J. O. (2017). A column generation approach for the integrated shift and task scheduling problem of logistics assistants in hospitals. *European Journal of Operational Research*, 260(1):316–334.

[174] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

[175] Wang, M., Wang, C., Yu, J. X., and Zhang, J. (2015). Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment*, 8(10):998–1009.

[176] Wei, H., Pan, Z., Hu, G., Zhang, L., Yang, H., Li, X., and Zhou, X. (2018). Identifying influential nodes based on network representation learning in complex networks. *PloS one*, 13(7):e0200091.

[177] Wei, Y.-C. and Cheng, C.-K. (1989). Towards efficient hierarchical designs by ratio cut partitioning. In *1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 298–301. IEEE.

[178] Wenping, Z., Chenhao, C., Yuhua, Q., and Jie, W. (2018). A two-stage community detection algorithm based on label propagation. *Journal of Computer Research and Development*, 55(9):1959.

[179] Whang, J. J., Gleich, D. F., and Dhillon, I. S. (2016). Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1272–1284.

[180] Xie, J.-R. and Wang, B.-H. (2017). Modularity-like objective function in annotated networks. *Frontiers of Physics*, 12(6):128903.

[181] Xu, G., Guo, J., and Yang, P. (2020). Tns-lpa: An improved label propagation algorithm for community detection based on two-level neighbourhood similarity. *IEEE Access*, 9:23526–23536.

[182] Yang, J. and Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213.

[183] Yip, K. Y., Cheung, D. W., and Ng, M. K. (2004). Harp: A practical projected clustering algorithm. *IEEE Transactions on knowledge and data engineering*, 16(11):1387–1397.

[184] Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473.

[185] Zeng, J. and Yu, H. (2018). A distributed infomap algorithm for scalable and high-quality community detection. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–11.

[186] Zhang, W., Shang, R., and Jiao, L. (2020). Complex network graph embedding method based on shortest path and moea/d for community detection. *Applied Soft Computing*, 97:106764.

[187] Zhao, D., Li, J., and Jiang, Z. (2021). Effects of link perturbation on network modularity for community detections in complex network systems. *Modern Physics Letters B*, 35(13):2150214.

Abstract

This thesis develops and evaluates novel combinatorial optimization techniques applied to two state-of-the-art network-based case studies. Its overarching goal is to identify, design, and evaluate efficient algorithms for analyzing large-scale systems representable as graphs, whether these systems arise from complex network interactions or are engineered for operational research applications.

The first part of the thesis focuses on combinatorial optimization and addresses a novel *Combined Waste Collection Problem* (CWCP). The CWCP as a two-stage decision problem, is defined on an undirected graph where each edge is associated with two transportation costs: one corresponding to a *Point-to-Point Waste Collection System*, in which waste is collected from designated points using trucks, and another associated with an *Automated Waste Collection System* (AWCS), in which waste is transported via a pipeline network to a central terminal. The CWCP aims to simultaneously decide whether each collection point should be served by a truck or by the AWCS while also seeking a set of feasible truck routes and, at most, one valid rooted tree that together serve all customer nodes in the graph at minimum cost. This two-stage optimization problem involves both a *Capacitated Vehicle Routing Problem* (CVRP) and a *Hop/Node-Weighted Steiner Tree Problem*, which are strongly interdependent. To solve the CWCP, matheuristic and exact algorithms are developed based on set-partitioning formulations, column generation, and a branch-price-and-cut framework. Two decomposition strategies, separate and combined, are explored to manage the coupling between routing and tree-design decisions. Computational experiments have been conducted on benchmark datasets and real-world data from Vienna.

The second part of the thesis investigates optimization-based approaches for analyzing large-scale complex networks, with a particular focus on *community detection*, the task of identifying densely interconnected groups of nodes within a graph. While this research is more closely aligned with network science than with classical operational research, it applies a similar analytical framework grounded in optimization theory. The studies presented in this thesis formulate community detection as integer and mixed-integer programming problems, where exact, heuristic, and hybrid techniques from combinatorial optimization are adapted to uncover structural patterns and functional modules within complex networks. Experimental evaluations on standard benchmark datasets were conducted to assess the performance of the proposed methods relative to leading approaches in the literature.

Zusammenfassung

Diese Arbeit entwickelt und bewertet neuartige kombinatorische Optimierungstechniken, die auf zwei hochmoderne netzwerkbasierte Fallstudien angewendet werden. Ihr übergeordnetes Ziel ist es, effiziente Algorithmen zur Analyse groß angelegter Systeme zu identifizieren, zu entwerfen und zu bewerten, die als Graphen darstellbar sind, unabhängig davon, ob diese Systeme aus komplexen Netzwerkinteraktionen entstehen oder für Anwendungen in der Betriebsforschung entwickelt wurden.

Der erste Teil der Arbeit konzentriert sich auf die kombinatorische Optimierung und befasst sich mit einem neuartigen kombinierten Abfallsammelproblem (CWCP). Das CWCP als zweistufiges Entscheidungsproblem ist auf einem ungerichteten Graphen definiert, wobei jede Kante mit zwei Transportkosten verbunden ist: eine entspricht einem Punkt-zu-Punkt-Abfallsammelsystem, bei dem der Abfall mit Lastwagen von bestimmten Punkten abgeholt wird, und eine andere einem automatisierten Abfallsammelsystem (AWCS), bei dem der Abfall über ein Rohrleitungsnetz zu einem zentralen Terminal transportiert wird. Das CWCP zielt darauf ab, gleichzeitig zu entscheiden, ob jeder Sammelpunkt von einem Lkw oder vom AWCS bedient werden soll, und gleichzeitig eine Reihe von realisierbaren Lkw-Routen und höchstens einen gültigen Wurzelbaum zu finden, die zusammen alle Kundenknoten im Graphen zu minimalen Kosten bedienen. Dieses zweistufige Optimierungsproblem umfasst sowohl ein Kapazitätsbeschränktes Fahrzeug-Routing-Problem (CVRP) als auch ein Hop/Node-gewichtetes Steiner-Baum-Problem, die stark voneinander abhängig sind. Zur Lösung des CWCP werden matheuristische und exakte Algorithmen entwickelt, die auf Set-Partitioning-Formulierungen, Spaltengenerierung und einem Branch-and-Price-and-Cut-Framework basieren. Es werden zwei Dekompositionsstrategien untersucht, eine separate und eine kombinierte, um die Kopplung zwischen Routing- und Baumdesign-Entscheidungen zu verwalten. Computergestützte Experimente mit Benchmark-Datensätzen und realen Daten aus Wien belegen die Effizienz, Skalierbarkeit und praktische Relevanz der vorgeschlagenen Methoden.

Der zweite Teil der Arbeit untersucht optimierungsbasierte Ansätze zur Analyse großer komplexer Netzwerke, wobei ein besonderer Schwerpunkt auf der Gemeinschaftserkennung liegt, also der Aufgabe, dicht miteinander verbundene Gruppen von Knoten innerhalb eines Graphen zu identifizieren. Obwohl diese Forschungsrichtung eher in der Netzwerkwissenschaft als in der klassischen Operationsforschung verwurzelt ist, verwendet sie einen analogen analytischen Rahmen, der auf der Optimierungstheorie basiert. Die in diesem Teil vorgestellten Studien formulieren die Community Detection als ganzzahlige und gemischt-ganzzahlige Programmierprobleme, bei denen exakte, heuristische und hybride Techniken aus der kombinatorischen Optimierung angepasst werden, um strukturelle Muster und funktionale Module innerhalb komplexer Netzwerke aufzudecken. Es wurden experimentelle Auswertungen anhand von Standard-Benchmark-Datensätzen durchgeführt, um die Leistungsfähigkeit der vorgeschlagenen Methoden im Vergleich zu führenden Ansätzen in der Literatur zu bewerten.