DTU Management
Department of Technology, Management, and Economics

Integrated Fault Localization and Field Service Logistics

Applied Multi-Disciplinary Approaches in Telecommunications

Siv Sørensen

PhD Dissertation

Integrated Fault Localization and Field Service Logistics

PhD Dissertation
October 2024

By
Siv Sørensen

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Technology, Management, and Economics, Akademivej, Building 358, 2800 Kgs. Lyngby Denmark
www.man.dtu.dk

Author

Siv Sørensen
PhD Candidate, Department of Technology, Management, and Economics
Technical University of Denmark

Main supervisor

David Pisinger
Professor, Department of Technology, Management, and Economics
Technical University of Denmark

Co-supervisor

Stefan Røpke
Professor, Department of Technology, Management, and Economics
Technical University of Denmark

Summary (English)

This dissertation addresses the need for more efficient field service operations in the telecommunications industry, focusing on reducing the kilometers driven by technician workforces while maintaining high service quality. The work is conducted as part of the project, GREENFORCE, an initiative aimed at leveraging advanced methodologies to enhance operational efficiency and reduce the environmental impact of field service operations, with this dissertation's primary focus being achieving these goals through optimized routing and scheduling using Operations Research.

The telecommunications sector faces unique challenges in managing its field service operations, particularly in *last-mile* networks, which connect customers to the main grid. These networks are characterized by their dynamic nature, with fault and maintenance tasks arising unpredictably every day, often with uncertain task durations and locations. Incomplete or outdated information about network topologies further complicates routing and task planning, leading to inefficiencies and unnecessary driving. These issues contribute to what the dissertation refers to as *micro-routing*—the additional driving and time spent during task completion that is often underestimated during the planning phase.

This dissertation addresses both micro-routing challenges and optimization problems related to improving the underlying data foundation for routing and scheduling decisions. By enhancing data accuracy and availability—particularly with respect to network topology and fault locations—this work aims to support more efficient planning and resource allocation in telecommunications field service operations.

The research outcomes are structured around five papers that collectively demonstrate the potential for both theoretical and practical advancements. The first paper introduces a novel method for reconstructing incomplete network topolo-

gies using phylogenetic principles. This method produces accurate reconstructions of the cabling topology of last-mile networks, reducing the uncertainty technicians face when responding to fault and maintenance tasks. The second paper builds on this by proposing a framework for extracting valuable insights from time series data collected by customer modems, enabling more informed approaches to fault detection and network maintenance

The third paper introduces a scoring system for identifying the most likely locations of network faults, providing technicians with a decision-making tool that guides them to high-probability fault locations. This not only reduces driving time but also enhances the accuracy of fault resolution. The fourth paper integrates the results of the first three and presents a fast, recursive algorithm that computes the optimal search strategy for technicians in the context of fault resolution with stochastic resolution locations, demonstrating how improved data and micro-routing considerations can lead to an 80% reduction in combined service and driving time. Finally, the fifth paper explores the topic of strategic workforce investments, offering a fast and practical framework for deciding how to hire an efficient field service workforce based on detailed operational scenarios, further demonstrating how an improved data foundation can inform long-term decision-making.

This dissertation contributes to both academia and practitioners by introducing novel methodologies, cross-disciplinary approaches, and solutions tailored to real-world challenges in telecommunications. It highlights the importance of stepping beyond traditional optimization methods to address complex operational problems and demonstrates how Operations Research can drive innovation, reduce emissions, and enhance efficiency in a rapidly evolving industry. Through its focus on both micro-routing and improved data foundations, this research paves the way for future advancements in field service operations and has the potential for significant environmental and economic impact.

Summary (Danish)

Denne afhandling adresserer behovet for mere effektive feltservice-operationer i telekommunikationsindustrien med fokus på at reducere de kørte kilometer af en gruppe teknikkere, samtidig med at den høje servicekvalitet opretholdes. Afhandlingen er udført som en del af projektet GREENFORCE, et initiativ, der sigter mod at udnytte avancerede metoder til at forbedre operationel effektivitet og reducere miljøpåvirkningen af driften af feltservice-operationer, hvor denne afhandlings primære fokus er at opnå disse mål gennem optimeret ruteplanlægning og tidsplanlægning ved brug af operationsanalyse.

Telekommunikationssektoren står over for unikke udfordringer i styringen af sine feltservice-operationer, især i *sidste-mil*-netværk, som forbinder kunder med hovednetværket. Disse netværk er kendetegnet ved deres dynamiske natur, hvor fejl- og vedligeholdsopgaver uforudsigeligt opstår hver dag, ofte med usikre løsningstider og lokationer. Ufuldstændige eller forældede oplysninger om netværkstopologier komplicerer yderligere ruteplanlægning og opgaveplanlægning, hvilket fører til ineffektivitet og unødvendig kørsel. Disse problemer bidrager til det, afhandlingen refererer til som *mikro-ruteplanlægning*—den ekstra kørsel og tid, der bruges under opgaveløsningen, som ofte undervurderes i planlægningsfasen.

Denne afhandling adresserer både udfordringerne ved mikro-ruteplanlægning og optimeringsproblemer relateret til forbedring af det underliggende datagrundlag for beslutninger om ruteplanlægning og tidsplanlægning. Ved at forbedre datanøjagtigheden og tilgængeligheden—særligt med hensyn til netværkstopologi og fejllokationer—sigter denne afhandling mod at understøtte mere effektiv planlægning og ressourceallokering i driften af feltservice-operationer i telekommunikationsindustrien.

Forskningsresultaterne er struktureret omkring fem artikler, der tilsammen demonstrerer potentialet for både teoretiske og praktiske fremskridt. Den første

artikel introducerer en ny metode til at rekonstruere ufuldstændige netværks topologier ved brug af fylogenetiske principper. Denne metode producerer nøjagtige rekonstruktioner af kabelføringen i sidste-mil-netværk og reducerer den usikkerhed, teknikerne står overfor, når de reagerer på fejl- og vedligeholdsopgaver. Den anden artikel bygger videre på dette ved at foreslå en metode til at udtrække værdifulde indsigter fra tidsseriedata, indsamlet af kundemodemer, hvilket muliggør mere informerede tilgange til fejldetektion og netværksvedligeholdelse.

Den tredje artikel introducerer et scoringssystem til at identificere de mest sandsynlige lokationer for netværksfejl, hvilket giver teknikerne et beslutningsværktøj, der guider dem til lokationer med høj sandsynlighed for fejl. Dette reducerer ikke kun køretiden, men forbedrer også nøjagtigheden af fejlløsninger. Den fjerde artikel integrerer resultaterne fra de første tre og præsenterer en hurtig, rekursiv algoritme, der beregner den optimale søgestrategi for teknikere i forbindelse med fejlretningsopgaver med stokastiske løsningslokationer, hvilket viser, hvordan forbedrede data og mikro-ruteplanlægningshensyn kan føre til en reduktion på omtrent 80% i samlet service- og køretid. Endeligt kigger den femte artikel på emnet "strategiske arbejdskraftsinvesteringer"og tilbyder en hurtig og praktisk metode til beslutningstagning om, hvordan man ansætter en effektiv feltservicebesætning baseret på detaljerede operationelle scenarier, hvilket yderligere viser, hvordan et forbedret datagrundlag kan informere langsigtet beslutningstagning.

Denne afhandling bidrager til både akademia og industrien ved at introducere nye metoder, tværfaglige tilgange og løsninger skræddersyet til virkelige udfordringer i telekommunikationsindustrien. Den fremhæver vigtigheden af at gå ud over traditionelle optimeringsmetoder for at tackle komplekse operationelle problemer og viser, hvordan operationsanalyse kan drive innovation, reducere emissioner og forbedre effektiviteten i en hurtigt udviklende industri. Gennem sit fokus på både mikro-ruteplanlægning og forbedrede datagrundlag baner denne forskning vejen for fremtidige fremskridt inden for feltservice-operationer og har potentiale til at have en betydelig miljømæssig og økonomisk indvirkning.

Preface

This dissertation was prepared at DTU Management at the section of Operations Research in partial fulfillment of the requirements for acquiring a PhD in Operations Research at the Technical University of Denmark, DTU.

The PhD project was funded by DTU and the Innovation Fund Denmark (Innovationsfonden) as a part of project 0224-00055B, GREENFORCE, which is a collaboration between commercial partners TDC-NET and Qampo as well as DTU Management and DTU Compute. The aim of GREENFORCE is to enhance the efficiency of field service operations and transportation by developing solutions that reduce the kilometers driven by field service workforces, without compromising customer satisfaction. This dissertation addresses this objective from a cross-disciplinary perspective rooted in Operations Research, with a particular focus on the telecommunications sector.

The work contained in this dissertation is the culmination of a three-year project under the supervision of Professor David Pisinger and Professor Stefan Røpke from the 15th of October 2021 until the 14th of October 2024.

A three-month research stay at the Discrete Mathematics & Optimization research group at the Technical University of Delft (TU Delft) was part of the project. This stay was supervised by Associate Professor Leo van Iersel.

The dissertation work amounted to the following four journal papers which are included and discussed in detail in this dissertation:

[I] Pisinger, D., and Sørensen, S., *Topology Reconstruction using Time Series Data in Telecommunication Networks. Networks*, 83(2):408–427, 2024.

[II] Rasmussen, T. E., Sørensen, S., Pisinger, D., Jørgensen, T. M., Baum, A., *Topology Reconstruction in Telecommunication Networks:*

Embedding Operations Research within Deep Learning. Under review at *Computers and Operations Research.* Preprint: [129].

[III] Sørensen, S. and Pisinger D., *Fault Estimation in Telecommunication Networks: A Maximum Parsimony Approach.* Working paper.

[IV] Sørensen, S. and Pisinger D., *Technician Routing and Scheduling for Tasks with Stochastic Resolution Locations.* Submitted to *Transportation Science.*

[V] Sørensen, S., Nielsen, C. C., Pisinger, D., Fürstenheim, J. I., *A Consensus Fixing Heuristic to Workforce Investments in Field Service.* Submitted to *Computers and Operations Research.*

The following extended abstract was also prepared but is not included in this dissertation:

[i] Rasmussen, T. E. and Sørensen, S., *Encoding Binary Events from Continuous Time Series in Rooted Trees using Contrastive Lerning.* Extended abstract presented as a poster at the *Northern Lights Deep Learning conference 2024 (NLDL 24)* by Rasmussen, T. E.. Preprint: [128].

Siv Sørensen

Kgs. Lyngby, October 14, 2024

Acknowledgements

This PhD would not have been possible without the help and support of numerous people and organizations.

First and foremost, my biggest thanks go to my main supervisor, Professor David Pisinger, who has been an immense support both professionally and personally throughout the past three years. David goes above and beyond in his duties as a supervisor and has always made it a priority to help and support the progression of my PhD. For this, I am immensely grateful. It has been inspiring, fun, and a great learning experience to work alongside David, and I will greatly miss us working together. I would also like to thank Stefan Rødpke for stepping up to the challenge of being my co-supervisor.

My thanks are also extended to all the partners in the GREENFORCE consortium; TDC-NET, Qampo, and DTU Compute. I am especially grateful for the close collaboration with my GEENFORCE colleagues at DTU Management and DTU Compute, David Pinger, Mette Gamst, Clara Chini Nielsen, Ewa Magdalena Behr, Joaquín Ignacio Fürstenheim, Andreas Baum, Thomas Martini Jørgensen, Bjarne Kjær Ersbøll, Tobias Engelhardt Rasmussen, and Meadhbh Healy. I have enjoyed our technical discussions and brainstorming sessions, and I am grateful for all the feedback on my work I have gotten from all of you.

I would also like to thank all my colleagues here at DTU Management. Many of you have provided early invaluable feedback on my papers; Alberto Tamburini, Rowan Hoogervorst, Atefeh Hemmati Golsefidi, Mikkel Lassen Johansen, Siv Cartland Hansen, Oliver Rise Thomsen, and David Ajit Kirpekar-Sauer, thank you especially for that. I have also enjoyed our many interesting lunch discussions and coffee chit-chats.

I am also grateful to Associate Professor Leo van Iersel and Mark Jones for hosting me at TU Delft during my research stay. Thank you for showing an

interest in my research and for the nice discussions and feedback. Thank you also to the entire team at the *Discrete Mathematics & Optimization* group at TU Delft, who were all very welcoming during my stay.

A special thank you also goes out to Tobias Engelhardt Rasmussen, with whom I had the pleasure of writing a paper. The collaboration reminded me of my 'young' uni days, and I enjoyed it immensely. Tobias' work efforts are commendable, and I wish to have his discipline level when I grow up. Also, Tobias provided the template for this dissertation, for which I am very grateful. Thank you also to Clara Chini Nielsen, with whom I have shared an office for the past three years and with whom I have also collaborated on a joint paper as well as other smaller projects.

I also had the pleasure of joining the board of the PhD Association at DTU in the last year of my PhD. It has been very fulfilling on a personal level to advocate for better working conditions for young researchers at DTU. Moreover, I am grateful for all the new friends and acquaintances I have gained in the Association. You all inspire me, and I wish you all the very best.

I would also like to thank my family—my sisters and my parents—for always believing in me and supporting me. I would not be where I am today without you. Thank you also to my friends; each of you brings me joy and your support and kindness mean the world to me. An honorable mention goes out to 'Trunterne' and 'Matmax'.

Lastly, I want to thank my biggest supporter through these past, at times, very difficult three years, my partner Bram. You have sacrificed so much for me, put up with so many of my frustrations, and been the rock I have clung to through both stormy and calm seas. Thank you for being the sunshine in my life.

Contents

Part I

Motivation and Scope

Chapter 1

Introduction

Humanity is standing at the brink of perhaps its most serious crisis: climate change. As the global population continues to grow, projected to peak around 10.3 billion by the mid-2080s [155], the average living standard worldwide also continues to rise [156]. While the improvement in global living standards is undoubtedly positive when considered in isolation, it poses a significant challenge to the climate when viewed alongside the anticipated population growth.

There is a well-established correlation between rising living standards and increased resource consumption and pollution [132, 70]. As living standards improve—particularly in areas such as income, industrialization, and energy consumption—societies tend to use more natural resources and emit more greenhouse gases (GHG).

If the world is to meet the ambitious goals of the renowned *Paris Agreement* [105]—limiting global warming to well below 2°C, preferably to 1.5°C, compared to pre-industrial levels and, and in the long term, reach net-zero emissions by the second half of the century—global GHG emissions must be reduced by at least 28% compared to current policy emission projections [157]. This leaves a significant gap in solutions and initiatives, where innovation, technological advancements, and research can play a pivotal role.

Project GREENFORCE was initiated at the beginning of 2021 to contribute to fill this gap. Supported by the Innovation Fund Denmark, GREENFORCE set out to increase the efficiency of field service operations and transportation by identifying ways to reduce the kilometers driven by field service workforces without compromising customer satisfaction levels [66]. The GREENFORCE

consortium consists of two industry partners and two academic partners. The industry partners are: TDC NET, the largest telecommunications infrastructure owner in Denmark, and Qampo, a software company specializing in decision science solutions. The academic partners are DTU Compute, with expertise in Machine Learning (ML) and Artificial Intelligence (AI), and DTU Management, with expert knowledge in Operations Research (OR) and Decision Science (DS).

The goal of project GREENFORCE is to develop cross-disciplinary solutions and research that address the complex challenges associated with efficiently managing a field service operation, such as the technician workforce employed by TDC NET. TDC NET employs 989 technicians who, among other tasks, service 4,181 mobile sites and maintain 19,750 km of transport fibre across Denmark [1]. Their technician workforce drives more than 80,000 kilometers daily—the equivalent of circling the Earth twice. One of the key outcomes expected from GREENFORCE is to decrease TDC NET's emissions by 1,400-1,900 tCO2e annually, through the collaborative efforts of the consortium. Specifically, this target will be achieved by reducing the yearly driven distance of TDC NET technicians by 18%-25% by 2025 compared to 2019, through improved field service planning.

This PhD dissertation focuses primarily on the work package titled *Routing Optimization* within the GREENFORCE project. Route optimization is a broad term, but in the context of field service, it corresponds to the well-known problem in research literature called the *Technician Routing and Scheduling Problem* (TRSP) [152]. This problem will be introduced in more detail later; broadly speaking, the TRSP deals with the challenge of assigning a set of on-site tasks to a field service workforce, ensuring that each task is completed within a specified time frame, service level agreements are met, and skill requirements are respected. In addition to task assignment, an efficient route—or task sequence—for each technician must be planned. Many objectives can be considered when evaluating solutions to this problem, but the most common are minimizing total driving distance or minimizing total time—both driving and resolution time.

With its diverse team of collaborators, the efforts of GREENFORCE are driven by the fusion of different skill sets and perspectives. While each technical work package has a clearly defined objective, it has been equally important to explore ways in which ideas and methodologies could be combined to create integrated solutions across operational, tactical, and strategic levels. For example, a technician dispatch algorithm can deliver more effective results in a dynamic environment when the dynamic input is thoroughly analyzed and understood; strategic decisions become clearer when grounded in an efficiently run operation that maximizes the use of all available resources. Understanding how low-level data collection can influence high-level strategic decisions—and everything in between—opens up intriguing new research challenges, fosters cross-disciplinary collaboration, and ultimately leads to more effective solutions with greater impact for both stakeholders and the environment.

This dissertation serves as a testament to these ideas, illustrating how computational biology, machine learning, computer science, and stochastic optimization, combined with OR, can address complex, real-world problems in telecommunication and field service. Furthermore, it demonstrates the value of enhancing the knowledge base for decision-making—particularly in understanding faults in communication signals—and how strategic decision-making can benefit from considering detailed daily dispatch decisions.

1.1 Dissertation outline

This dissertation is divided into two main parts:

Motivation and Scope This part contains a single chapter—the current one—which begins with the motivation and context for the dissertation. Next, an overview of the Technician Routing and Scheduling Problem (TRSP) and its challenges is presented. The purpose and contributions of the dissertation are then outlined, along with summaries of the five scientific papers that the work amounted to. The end of the chapter is marked by brief concluding remarks.

Research Outcome This part contains five chapters, one for each of the papers that constitute the scientific outcome. Four of these papers have been either published, submitted, or are currently under review, while the last paper is a working paper. Each paper is thoroughly introduced in subsection 1.4.1.

The final chapter concludes the dissertation by summarizing the work and outlining potential directions for future research.

1.2 Motivation

Society continues to evolve at a rapid pace, with technological advances leading the way. Automation, digitalization, Machine Learning (ML), the Internet of Things (IoT), Big Data, Artificial Intelligence (AI), and, more recently, Large Language Models (LLMs), are some of the terms that most people have become familiar with in recent years. With the promise of improved efficiency and cost-saving, companies have had to respond to these trends to stay competitive and relevant. However, many companies struggle to navigate this new landscape of opportunities, facing various practical challenges that stand in the way of profitability [91, 36].

Data is being collected and stored on an unprecedented scale [140]. Advances in data density, materials science, and computing power have significantly improved the efficiency and capacity of physical storage devices, reducing the cost per gigabyte for both hard drives (HDDs) and solid-state drives (SSDs). This

has made it possible to store massive amounts of data at very low cost [106]. However, the real challenge lies in translating this data potential into tangible value through the use of advanced technological tools.

In 2001, Doug Laney introduced the concept of Big Data characterized by the "three Vs"—*volume*, *velocity*, and *variety*. Since then, two more Vs have been added: *veracity* and *value* [172]. *Volume* refers to the size of the data, *velocity* to the speed at which it is generated and collected, and *variety* to the diverse sources or types of data. *Veracity* captures the uncertainty or trustworthiness of the data, while *value* points to the hidden insights that can potentially be extracted from it. The combination of these factors poses significant challenges for data-driven techniques like ML, which depend on large, high-quality datasets to uncover patterns, make predictions, or enable classifications. Here, practitioners, perhaps somewhat naïvely, have been led to believe that with enough data over a sufficiently long period, ML or AI would inevitably be able to uncover *something* of value that humans cannot. While there are many excellent real-world cases where this has proven true, it certainly is not always the case.

Operations Research (OR) offers a distinct approach, as it is model-driven rather than data-driven. This difference may also explain why OR has not received as much attention as, for example, ML and AI. At first glance, data-driven approaches appear to have lower entry barriers, as many companies are already accumulating vast amounts of data. In contrast, models are more complex to define and require a deep understanding of the problem being addressed as well as the external factors influencing it. Despite its apparent complexity, OR has, and still continues to deliver cost-saving and revenue-generating solutions to organizations across various industries. Hillier and Lieberman [62] provide numerous real-world examples where OR applications resulted in annual savings, often reaching hundreds of millions of US dollars.

In many real-life OR applications, increased profitability goes hand-in-hand with more efficient resource usage, thus contributing to a reduction in carbon footprint. A notable example is the United Parcel Service (UPS), which developed the *On-Road Integrated Optimization and Navigation* (ORION) platform. ORION is estimated to reduce each of their driver's travel distance by 10 to 14 miles per day, saving approximately 10 million gallons of fuel annually [158]. This initiative won the Franz Edelman Award in 2016, recognized for its vast scale and impact, and is considered one of the largest and most influential OR projects globally [65].

The telecommunications industry is another sector with significant potential for OR solutions to enhance profitability while benefiting the climate. Global data consumption over telecommunications networks is projected to nearly triple, from 3.4 million petabytes (PB) in 2022 to 9.7 million PB in 2027 [126]. Additionally, the global telecommunications sector was responsible for 133 million tCO2e in operational emissions in 2022, accounting for 0.4% of global energy-

related greenhouse gas (GHG) emissions. Notably, 5% of these emissions were attributed to fuel consumption by various operational activities such as large vehicle fleets [168].

Field service employees, such as technicians and engineers, account for a substantial part of the workforce in the telecommunication industry. At TDC NET, for example, field technicians make up 38% of the full-time workforce [1]. As the name implies, field service tasks are associated with external locations, meaning that transportation is often a significant part of the job. Within telecommunication, field service tasks can generally be divided into two categories: maintenance and servicing of existing infrastructure, and the deployment of new infrastructure, such as 5G. Of these, maintenance tasks tend to be the most challenging to manage efficiently for several reasons: task duration can be difficult to estimate in advance, particularly for fault-related tasks; dynamic tasks arrive sporadically throughout the day and need to be integrated into the existing schedule; the precise location of a task might be unknown due to incomplete infrastructure information or the unknown nature of a fault; and factors such as driving conditions and parking, especially during peak hours, add further uncertainty. Common to all of the aforementioned difficulties is *uncertainty*. The common theme underlying all these challenges is *uncertainty*. By either reducing the uncertainty associated with field service or effectively incorporating it into the planning process, field service workforce management can be significantly optimized. This dissertation demonstrates just that.

When uncertainty plays a role, the 'best' solution ultimately depends on the strategic risk profile a company is willing to adopt. In an ideal world, the field service workforce would be large enough to comfortably meet any demand on any given day, with all technicians possessing the necessary skills to handle all tasks and would be geographically spread out in accordance with the distribution of tasks. However, due to cost constraints and hiring limitations, this ideal scenario is rarely achievable. This is where Operations Research (OR) can contribute by transparently highlighting which strategic solutions are optimal given budget constraints, customer satisfaction requirements, or other limitations. A common approach to strategic decision-making in the face of uncertainty is to conduct a what-if analysis. However, a limitation of this approach is that while it may yield valuable insights, it does not always offer clear recommendations for action. Even when specific risks or opportunities are identified, what-if analyses do not always indicate the most appropriate strategic response. OR, by contrast, can offer complimentary insights as it is centered around decisions and provides concrete solutions to not only operational problems but also strategic ones.

Although improving operational planning offers immediate benefits, such as reducing travel time for technicians from one day to the next, real, long-term impact is often achieved by addressing strategic issues as well. To meet the ambitious goals of the Paris Agreement, practitioners must find innovative so-

lutions across all levels of their business, from day-to-day operational decisions to long-term strategic planning. This dissertation aspires to inspire such innovation.

1.3 Technician Routing and Scheduling

The Technician Routing and Scheduling Problem (TRSP) is a key problem within the broader research area of Workforce Scheduling and Routing Problems (WSRP). A WSRP encompasses any problem involving the mobilization of personnel in order to perform work-related activities at various locations [108]. In such problems, the number of tasks to be completed typically exceeds the size of the available workforce, meaning that each worker must perform multiple tasks over the given planning period. In the TRSP, technicians carry out various maintenance and installations installation tasks typically related to infrastructure like telecommunication networks or utility services (e.g., gas, electricity, water). Another notable problem under the WSRP umbrella is the Home Health Care (HHC) problem, where healthcare professionals provide services at the homes of elderly or sick individuals—a widely studied problem in the literature due to its extensive real-world relevance [45].

While the routing aspect of the WSRP is clear—a set of tasks located at different locations naturally involves personnel routing—the scheduling component stems from the time windows associated with each task. Factors like service agreements or access to customer premises often drive these time-window constraints. Although these problem aspects might suggest that the WSRP is essentially a Vehicle Routing Problem with Time Windows (VRPTW), key differences set the two problems apart. The VRPTW primarily addresses logistical concerns, focusing on routing vehicles to minimize costs or travel time, with the fleet usually being homogeneous apart from vehicle capacity. In contrast, the WSRP involves routing and scheduling personnel, taking into account their skills, availability, and the dynamic nature of task demands. This makes the WSRP more complex and service-oriented than the VRPTW. Additionally, in the WSRP, service times are often both significant in duration and characterized by high variability, unlike in the VRPTW, where the time spent at each customer location is typically negligible.

The TRSP is generally considered to be NP-hard due to its close relationship with the VRPTW, which has been proven to be NP-hard in the strong sense [139]. The TRSP extends the VRPTW by incorporating specific constraints related to technician skills, task requirements, and scheduling, which further increases the complexity of the problem.

Although the various problems under the term WSRP appear similar, small but key differences significantly impact the nature and challenges of each problem.

Consequently, separate and self-contained research subfields have developed for each subproblem. For example, in home-care services, repeat customers are frequent, and customer satisfaction is often closely linked to having the same small set of workers attend to the same customer, also known as *continuity of care*, as elders, for instance, tend to prefer people they know and feel comfortable with [82]. In the TRSP, the relationship between the customer and the technician is generally inconsequential, but other challenges, such as spare part availability and dynamic task arrivals, play a more significant role.

Many variants of the TRSP exist but some of the most common characteristics of the problem include:

Time-windows: A time frame within which a task must be started or finished, depending on the definition. Both soft and hard time-window constraints can exist. Hard constraints often involve scenarios where, for example, a customer must be present to allow a technician entry into their home. Soft constraints are typically linked to service level agreements, where time-window violations are permitted but induce a penalty or cost. Pourjavad & Almehdawe [120] considered an overnight version of the TRSP with soft time-windows, in which a technician spends two nights on the road before returning to the depot at the end of the third day. Their model demonstrated substantial savings in both travel costs and the number of required technicians when tasks are distributed across a large geographical area.

Start and end location: Each technician typically starts and ends their day either at a central depot or at their personal home, depending on the setup.

Skills: Each technician possesses a specific set of skills that determine the subset of tasks they can handle. Most studies consider a heterogeneous skill distribution among technicians.

Experience level: While the skill set defines *which* tasks a technician can perform, a technician's experience level determines *how quickly* they can complete a task. Chen et al. [24] addressed a multi-period technician routing problem incorporating experience-based learning, where technicians become faster at handling tasks they repeatedly encounter. They used a Markov Decision Process model to capture the learning dynamics over a multi-period timeline, aiming to minimize the completion time of the last task. Their results showed that accounting for technician learning led to better solutions compared to assuming static productivity levels.

Spare parts and tools: Maintenance and installation tasks typically require a variety of spare parts and tools. This is why most TRSP research considers vehicles, particularly vans, for transportation. Due to the limited capacity for spare parts and the sometimes unpredictable nature of service requirements, technicians might need to restock during the workday. Pillac et al. [116] addressed this challenge by proposing a parallel matheuristic

approach comprising three sequential components: a constructive heuristic, a parallel adaptive large neighborhood search (pALNS), and a final post-optimization step. Their method found 44 out of the 55 best-known solutions on the Solomon VRPTW benchmark instances [146] in under 30 seconds.

Service time: The TRSP is inherently focused on technicians performing tasks with varying durations and levels of unpredictability. Naturally, both the expected and actual durations of tasks significantly contribute to the real-world complexity of the TRSP.

Unserved tasks: Depending on the context of the studied TRSP, there may not be enough technicians to handle all task requests. Some models address this by incorporating a penalty for each unserved task, while others focus on maximizing the number of tasks completed. Other common approaches for mitigating unserved tasks also include outsourcing and allowing overtime [175, 54].

Dynamic tasks: Many TRSP problems include tasks with a dynamic and urgent nature. Faults in supply networks, such as water or electricity, impact many customers and must be handled as soon as they arrive. At TDC NET, approximately 25% of all tasks are dynamic and require immediate attention. Nielsen and Pisinger [101] proposed a tactical framework to handle dynamic tasks more effectively. They studied a multi-period planning horizon in which they for each day grouped the known tasks together from selected areas to decrease the required driving for the start-of-the-day routes. They proposed several strategies for how to shape and position these areas to ensure short service times for the dynamically arriving tasks and they demonstrated an around 10% reduction in total driving distance compared to a traditional ad-hoc planning approach.

Workload balancing: To increase technician job satisfaction and fairness, different forms of workload balance have also been studied. This could for instance be in relation to the duration of each technician's work-day or with respect to the idle time in their respective schedules. Anoshkina and Meisel [6] studied a TRSP variant involving team-based technician deployments. They presented a mixed-integer programming (MIP) formulation with a multi-objective function that considers service quality, costs, and fairness in workload distribution. Their solution approach decomposes the problem into sequential subproblems, starting with team formation, followed by job assignment and routing. Their computational study provided both quantitative and qualitative insights into how different solution strategies impact the three aspects of the objective function.

Due to its complexity and the size of real-life applications, the research literature for the TRSP predominantly consists of heuristics approaches. However, a handful of papers also consider exact methodologies for smaller instances [16, 88, 175, 33]. Recent heuristic solution methodologies in the litera-

ture include approximation heuristics based on Markov decision Process (MDP) models [104, 25], Iterative Local Search (ILS) [58, 170], Adaptive Large Neighborhood Search [101, 54], and Genetic Algorithms (GA) [111, 35], just to name a few.

For a thorough review of the TRSP literature focused on real-life applications, prominent properties, solution methods, and research prospects, we refer to the review by Paraskevopoulos et al. [108] on *resource constrained routing and scheduling*.

1.3.1 TRSP challenges in telecommunication

Technicians in the telecommunication industry primarily handle installation and maintenance tasks in the part of a telecommunication network referred to as the *last-mile network*. The last-mile network connects customers (private homes and businesses) to a fiber-optic backbone grid. Last-mile networks are essentially tree topologies, where each leaf node represents a customer, and the root node serves as the singular connection to the backbone grid. These networks can range from a few customers to over a thousand, with a couple of hundreds being most common [4]. The distance, or range, of a last-mile network varies depending on the wired technology (coax, fiber, or copper), spanning from a few hundred meters to tens of kilometers [69]. At TDC NET, the majority of their last-mile networks are Hybrid Fiber-Coaxial (HFC) networks. The reader is referred to the dissertation by Rasmussen [127] for a more thorough introduction to broadband networks and technologies including a data overview as well as common types of faults.

Installation tasks are usually scheduled in advance and are often associated with either a half-day or full-day time window, especially if the installation requires access to a customer premise. Fault tasks or other emergency tasks can arrive at any time during the day and generally must be addressed the same day they occur. Tasks that are known at the start of the planning period are referred to as *static* tasks, while those that arrive during the planning period and require immediate attention are known as *dynamic* tasks. Tasks are mostly located at either customer premises, somewhere within the network infrastructure, or at distribution or central hubs. As part of completing a task, technicians may need to access components located in roadside cabinets, underground, or in buildings with restricted access.

Uncertainty

Various aspects of uncertainty further complicate technician scheduling and routing in telecommunication:

- Driving times and distances are difficult to estimate due to factors like traffic conditions and parking availability.

- Task durations are especially challenging to predict for several reasons. Firstly, a technician's experience level and work ethic significantly influence the time required to complete a task. Secondly, prior knowledge about fault tasks is often limited; technicians rarely know the precise nature, extent, or location of a fault before they arrive on-site.
- The daily arrival of dynamic tasks, their locations, and the resources needed to handle them add to the complexity.
- Fluctuating task demand forces management to make tactical and strategic decisions that affect daily routing, such as overtime policies, deployment of emergency night-shift technicians, or adjustments to both the daily and long-term workforce size.

Despite the clear need to incorporate uncertainty into TRSP research to enhance its practical applicability, a recent review of the literature [108] found that studies addressing stochastic environments are very limited, identifying this as a major research gap in the field.

Poor information foundation

Limited or outdated information related to tasks significantly impacts both the time and travel required to complete them:

- Missing or outdated details about the topology of last-mile networks complicate both installation and fault tasks, making them prone to errors. Network topology records are often lost during company mergers, remain undigitized, or are not systematically updated. For example, at TDC NET, only 15% of their 4,562 coax networks have more than 90% of the network topology documented, and for 45% of their networks, the complete network topology is entirely missing.
- For fault tasks, technicians are solely responsible for identifying and localizing faults based on signal quality measurements and diagnostics that they gather themselves on-site. This leads to a tedious and often inefficient troubleshooting process.

All of the above-mentioned challenges highlight an often overlooked aspect of the TRSP problem in telecommunication, which we have come to refer to as *micro-routing*. Micro-routing encompasses all additional driving and time requirements associated with a task, beyond the planned travel between scheduled tasks. In the telecommunication sector, the lack of focus on micro-routing during the planning phase of TRSP scenarios frequently results in significant underestimations of the actual travel time and distance. By including or enhancing micro-routing considerations within the TRSP, task durations and overall driving can be reduced, and more realistic work schedules can be generated.

Figure 1.1 illustrates some examples of micro-routing in telecommunications.

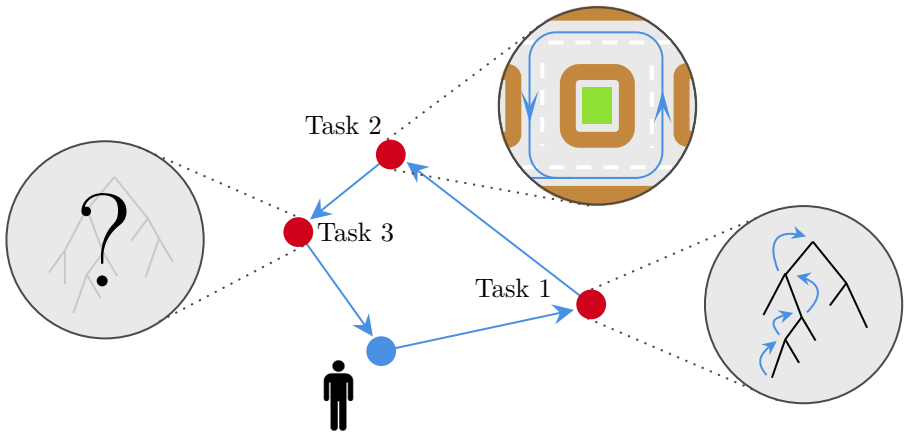


Figure 1.1: An illustrative example of micro-routing in telecommunications. A technician is assigned a route with three tasks where the blue arrows in between the tasks illustrate the planned route. However, when the technician arrives at Task 1 (a fault task) it is not immediately clear where the fault is located, so the technician has to drive around visiting multiple sites in the faulty network. When the technician arrives at Task 2, located in an urban area, no parking spots are available and the technician circles the block until a spot frees up. When arriving at Task 3, the technician has to carry out a trivial installation task in a last-mile network, but no topology information is available. Through trial and error, the technician eventually manages the installation but spends much longer time than planned.

Specifically, the purpose of this dissertation is to identify and solve micro-routing problems and identify ways in which OR can be used to improve the data foundation on which routing and scheduling decisions are made on in telecommunications. In achieving this aim, adopting an experimental and creative approach to problem-solving has been necessary, as opposed to relying solely on traditional methods. Moreover, in light of the aim of reducing driven kilometers, the dissertation is also focused on solving problems of a more applied nature as opposed to more theoretical ones, while still providing value to the research community.

1.4 Purpose and contributions

Within the *Routing Optimization* work package of project GREENFORCE, the overarching purpose has been to explore how Operations Research (OR) can contribute to reducing the kilometers driven by a field service workforce, particularly in the telecommunications industry. The work under this goal has evolved into two tracks with distinct scopes:

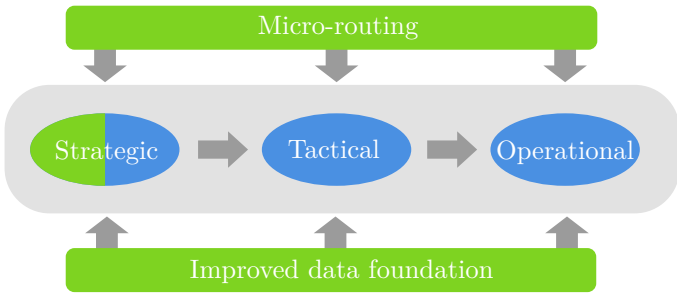


Figure 1.2: An illustration of the scope of the two tracks within the *Routing Optimization* work package of project GREENFORCE and how they tie together. The scope of Track 1 is highlighted with blue and the scope of Track 2 is highlighted with green. This dissertation is concerned with track 2. Where Track 1 is focused on core TRSP problems, Track 2 demonstrates how both an improved data foundation as well as micro-routing can be used to improve various aspects of the core problem, effectively boiling down to a more efficient field service operation.

Track 1 Focuses on the traditional core of field service routing and scheduling problems, covering both operational, tactical, and strategic problems. This track resulted in the dissertation by Nielsen [100].

Track 2 Primarily focuses on exploring opportunities beyond the traditional scope, identifying new challenges and opportunities for improvements in field service operations. This dissertation is the outcome of this track.

The two tracks complement each other, collectively addressing the various decision layers and factors that influence the efficiency of a field service workforce, as illustrated in Figure 1.2.

The specific aim of this dissertation is to identify and solve micro-routing problems and to explore how OR can enhance the data foundation for routing and scheduling decisions in telecommunications. In achieving this aim, adopting an experimental and creative approach to problem-solving has been necessary, rather than relying solely on conventional methods. Moreover, in light of the overarching aim of reducing driven kilometers, the focus of this dissertation is on addressing problems of a more applied nature, while still introducing novel methods and advancing research within the academic community.

This effort has culminated in five research papers that collectively address these objectives. Moreover, the work makes significant contributions to the research community by proposing new exact and heuristic methodologies, identifying novel research challenges, and extending cross-disciplinary solutions. Paper I demonstrates how unknown telecommunication network topologies can be inferred remotely by exploiting faults and inconsistencies in customer modem data as well as geographical distances. Knowing the full network topology reduces the number of locations technicians must drive to to resolve certain tasks. Pa-

per II extends Paper I by proposing a framework for identifying and encoding *relevant* information in modem data with respect to the aim of inferring missing topology. Paper III also builds on the ideas of paper I and proposes a framework for how customer data can be used to estimate the likelihood of different components in a telecommunication network being faulty. This information can be used to effectively search for and resolve network faults. This is demonstrated in Paper IV which presents an optimal search policy for a technician to follow when trying to pinpoint the source of a fault in a telecommunication network. Finally, Paper V extends the scope by providing a framework for strategic workforce investments, capable of considering detailed operational-level nuances. With the exception of Paper V, the remaining works introduce novel cross-disciplinary methodologies.

1.4.1 Research summaries

Chapter 2 (Paper I): Topology Reconstruction using Time Series Data in Telecommunication Networks

Status: Published in *Networks* on November 2023.

Authors: Pisinger D. · Sørensen, S.

Presented: as a talk at the 2022 EURO Conference.

This paper addresses the challenge of unknown *last-mile* telecommunication network topologies, which presents a significant obstacle for technicians responsible for network maintenance. In these tree-like networks, the location of components—such as customer nodes and cable-splitter nodes—are known, but frequently the cabling is not. The paper proposes a topology reconstruction heuristic based on the available information, namely; customer modem data and the geographical locations of known network components.

The methodology is based on the assumption that customer modem data can be translated into a series of *events*, and that two customers sharing many connections in a network will observe many similar network events. Building on this assumption, the paper proposes to use the *maximum parsimony* principle from phylogenetic inference, traditionally used in computational biology, in combination with geographical distances, within a novel *variable neighborhood search* (VNS) heuristic. Moreover, three neighborhood functions are introduced, along with three construction heuristics.

The proposed methodology is capable of reconstructing large topologies of up to 500 customers with a high degree of accuracy, even with up to 30% background noise in the modem data. The paper draws parallels between topology reconstruction and inferring family trees, demonstrating that phylogenetic methods can be effectively applied to more general topology reconstruction problems beyond biology. This encourages further cross-disciplinary research, where either

new phylogenetic principles can be explored or different topology reconstruction problems can be tackled.

Chapter 3 (Paper II): Topology Reconstruction in Telecommunication Networks: Embedding Operations Research within Deep Learning

Status: Paper is submitted and under peer review for publication in *Computers and Operations Research*.

Authors: Rasmussen, T. E. · Sørensen, S. · Pisinger D. · Jørgensen, T. M. · Baum, A.

Presented: as a poster at the Northern Lights Deep Learning conference[1] (NLDL 2024).

This paper addresses the overarching aim of reconstructing the cabling arrangements in *last-mile* telecommunication networks using data from customer modems. Each modem monitors the quality of the received downstream signal through a series of continuous data metrics. The state of the data, when it reaches the modem, is contingent upon the path it traverses through the network and can be affected by, e.g., corroded cable connectors. The paper hypothesizes that by identifying time points at which a modem is affected by irregularities or faults from further up in the network, the network topology can be reconstructed.

The paper proposes training an encoder capable of identifying irregular, inherited information in modem data and subsequently encoding it as discrete binary sequences. Specifically, the encoding scheme is based on unsupervised contrastive learning, where a Siamese neural network is trained on a positive (true) topology, its associated modem data, and a set of negative (false) topologies. The weights of the Siamese network are updated continuously using a modified version of the Maximum Parsimony optimality criterion. In this way, the encoder learns which data makes the true topology the most *likely* topology under this criterion.

The data encoding scheme developed in this paper is intended as a preprocessing step for the heuristics proposed in Paper I, but the encoder also demonstrates promising results on its own. In networks with up to 20 internal nodes, the learned encoder achieves an estimated topology reconstruction accuracy of 99.9% and a data event identification accuracy of 93.4%. Furthermore, the paper shows that the synergy between OR and deep learning can be harnessed by integrating an optimization algorithm directly into a deep learning loss function. Additionally, a new stochastic version of the Maximum Parsimony algorithm is also introduced.

[1] A poster was prepared together with co-author T. E. Rasmussen, but only he presented it.

Chapter 4 (Paper III): Fault Estimation in Telecommunication Networks: A Maximum Parsimony Approach

Status: Working paper.

Authors: Sørensen, S. · Pisinger D.

This paper explores the use of the *Maximum Parsimony criterion* for fault detection and localization in telecommunication networks. It introduces a novel framework that reconstructs a set of likely downstream data *states* at each interior node of the network, based on the observed modem data. These state sets are subsequently utilized in a scoring function, where higher scores at interior nodes indicate a higher likelihood of being the source of a network fault.

We simulate a diverse range of network faults across the instances described in Paper I, investigating the impact of fault interference levels as well as different ways in which a fault can affect existing modem data. The results show that the framework accurately identifies the correct fault location within its top two predictions in 90% of the cases analyzed. These findings create new avenues for future research, particularly in integrating fault localization uncertainties into routing problems like the Technician Routing and Scheduling Problem (TRSP), as we demonstrate in Paper IV.

Ongoing work includes developing a method to convert the *scores* of each potential failure point in the network into a more concrete estimation of the likelihood of failure.

Chapter 5 (Paper IV): Technician Routing and Scheduling for Tasks with Stochastic Resolution Locations

Status: Paper is submitted for publication in *Transportation Science*

Authors: Sørensen, S. · Pisinger D.

Presented: as a talk at the 2024 EURO Conference.

This paper presents a novel stochastic extension of the TRSP, where each task is associated with multiple potential resolution locations, each with a distinct probability of success. This problem corresponds to the challenge of locating and resolving faults in a telecommunication network. For each task, all resolution locations are arranged in a linear sequence, and when a technician inspects one of the locations, they will either be able to resolve the task by finding the fault or determine whether the fault lies further along in the sequence, either to the left or right.

The objective is to determine a policy such that a technician workforce can resolve all tasks using the least amount of expected time—considering both resolution and driving time. The paper proposes a set of greedy policies in

line with current industry practices. Moreover, the paper demonstrates that the problem can be decomposed into an inner and outer problem, proves that any solution to the inner problem exhibits a *Binary Search Tree* structure, and, lastly, presents a recursive algorithm that finds the optimal policy for the inner problems in polynomial time.

The findings offer practical strategies for improving fault localization in telecommunication networks and other sectors, providing real-world applicability with potential time savings of around 80%. The proposed algorithm is shown to be effective even in an uninformed setting where no prior information about resolution probabilities is available, further underscoring its practical potential. For the research community, this problem extension opens the door to new research opportunities; in particular, considering other sequential dependencies or added complexity presents intriguing avenues for future exploration.

Chapter 6 (Paper V): A Consensus Fixing Heuristic to Workforce Investments in Field Service

Status: Paper is submitted for publication in *Computers and Operations Research*.

Authors: Sørensen, S. · Nielsen, C. C. · Pisinger, D. · Fürstenheim, J. I.

This paper presents a novel approach to workforce investments in field service. The work adapts an innovative investment methodology proposed by Pisinger [118], where *consensus* among a set of training scenarios is used to make strategic investment decisions. The paper specifically addresses investment decisions in the form of hiring new technicians with varying skills and starting locations. The workforce is built iteratively from scratch, one technician at a time, in a greedy fashion, selecting the best hire based on a scoring function at each step.

The objective is to minimize the combined costs of hiring, driving, and penalties for unserved tasks. For each investment decision, a full TRSP model is heuristically solved for a set of real-life scenarios, each representing a full day of operations at TDC NET. The workforce assembled by the proposed investment framework is then evaluated on an out-of-sample set of scenarios representing future daily demands. This evaluation is performed by comparing the results to a set of novel *scenario-specific* solutions.

The proposed framework is highly parallelizable and can be integrated with any existing field service model and solver, making the barriers to implementation low. The framework is also flexible regarding the types of investments considered, as long as the investment set is discrete. Results showed that decision makers must balance two extremes: hiring a large robust workforce that can serve all tasks in all scenarios or hiring a workforce that minimizes total

cost but at the expense of not serving all tasks. Since the investment framework is fast, decision makers can readily generate workforces of different sizes and transparently choose which trade-off between cost and customer service is most appropriate for their business. Overall, the framework provides multiple valuable managerial insights regarding the mitigation of unserved tasks, operation under budget constraints, and identifying which future scenarios pose the greatest challenges for a proposed set of investments and why.

1.4.2 Contributions

The main contribution of this dissertation is demonstrating the value of thinking beyond the traditional scope of a Workforce Scheduling and Routing operation. Specifically, the dissertation does this by considering optimization problems in micro-routing and optimization problems aimed at improving the data foundation for route planning. This insight is valuable to both practitioners in the telecommunication sector and the research community within Workforce Scheduling and Routing problems.

What initiated and inspired the direction that the research ended up in was the industry partner, TDC NET, who emphasized the importance of being able to efficiently reconstruct their missing topology records. This would not only help minimize micro-routing but also provide technicians with better information, reducing the likelihood of mistakes. Based on the information TDC NET explained was available, the Variable Neighborhoods Search algorithm (VNS) from Paper I was developed. This algorithm shows significant potential in reconstructing the network topology of real last-mile networks, achieving accurate reconstructions for networks of up to 500 customers, even with up to 30% random background noise. Without background noise, the algorithm was able to perfectly reconstruct 11 out of the 15 networks considered.The algorithm is based on an unconventional multi-disciplinary approach, that showcases the potential in using phylogenetic principles and methods for general topology reconstruction beyond the scope of phylogenetics. This contribution highlights to the research community the value of identifying similar structural problems in other fields.

However, the algorithm from Paper I relies on the input data being in a discrete format and containing relevant information about signal quality. While TDC NET collects various continuous performance metrics related to signal quality, these metrics are mainly used by technicians on-site to manually assess if values fall within 'normal' thresholds. This gap in effectively understanding and utilizing the collected data led to the work in Paper II. In this paper, a novel encoder based on unsupervised contrastive learning successfully extracts valuable information from time series data. Specifically, the paper demonstrates that the encoder can identify and encode *network events*, such as faults, which affect the data state in descendant network components. This is achieved by

integrating an optimization algorithm directly into the encoder's loss function. The approach achieves an event prediction accuracy of up to 93%, strongly suggesting that the learned data can be leveraged for topology reconstruction in a framework like the one in Paper I. The paper adds value to the industry by demonstrating that modem data can be harnessed effectively when focusing on a specific purpose and designing methods accordingly. Rather than asking, *"what can I learn from my data?"*, the approach encourages asking, *"what can my data tell me about X, and only X?"*. To the research community, the methodology shows that a loss function does not need to be a simple calculation, but can represent the solution—or an approximate solution—to an optimization model, which could inspire future novel cross-disciplinary approaches between OR and machine learning.

Having shown in Paper II that it is possible to encode modem data containing *network events*, this led to the idea of using the same data to estimate *where* in a network a faulty component might be located. In other words, providing technicians with more concrete information on where to look for network faults, or where resources are best allocated for predictive maintenance. Paper III achieves this by assigning a *score* to each potential fault location, with higher scores indicating a higher likelihood that the location is faulty. The framework for calculating these scores, similar to the topology reconstruction algorithm from Paper I, utilizes an optimality criterion from phylogenetics. In addition to the aforementioned practical benefits this framework could offer the industry, it introduces a new and straightforward approach to the research field of anomaly detection, which is traditionally dominated by machine learning and unsupervised methods.

The fourth paper is a culmination of the first three papers and directly demonstrates the combined value of both an improved data foundation and micro-routing in reducing the kilometers driven by a technician workforce in telecommunications. The paper presents a recursive algorithm that computes the optimal strategy for how a technician should search for an unknown fault affecting a customer in a last-mile network. Results show that TDC NET could reduce the combined time spent on service and routing related to fault tasks by around 80% using the proposed, optimal search strategy compared to their current approach. However, the optimal strategy can only be computed when the network topology is known and is most effective when the probability of failure at each potential fault location is also known—or, can be estimated. All of this information should be feasible to obtain, as demonstrated by the first three papers. The paper arrives at the optimal strategy (policy) by modeling the problem as a Markov Decision Process, highlighting the importance of properly modeling stochastic problems before immediately trying to formulate some approximation framework. Since the optimal policy can be obtained in polynomial time, the framework is well-suited for live, rolling-horizon scheduling and routing.

The last paper provides a practical and fast framework for investing in new technicians based on detailed operational scenarios. Although this paper is not directly related to the other papers in this dissertation, the investment decisions are only as good as the routing scenarios it considers. This means that the more accurate and realistic the schedules for each technician are, the better the investment decisions. By factoring in micro-routing and the effects of improving the organization's data foundation, investment decisions should also improve indirectly, highlighting how micro-routing can impact even strategic decisions. The proposed framework can be readily integrated with existing routing software, making the barriers to implementation for the industry low. For the research community, the idea of *consensus fixing*, which the investment framework is based on, offers an interesting and flexible new approach to strategic investment problems beyond workforce investments.

1.5 Concluding remarks

It is the hope that this introduction has motivated and prepared the reader well for the full papers included in the next part of this dissertation. This current part of the dissertation has tried to present the history and the thought process behind the work that has been carried out over the past three years.

Although this dissertation by no means follows a traditional structure, focusing not on a singular optimization problem but rather a range of interconnected challenges, this broader approach reflects the complexity of real-world problems, especially in the telecommunication industry. Addressing the multiple facets of technician routing, scheduling, micro-routing, and data-driven improvements is critical to developing holistic solutions with real impact.

The decision to take a non-traditional path was driven by the need to extend beyond theoretical advancements and engage with the practical realities faced by organizations like TDC NET. This dissertation showcases how interdisciplinary approaches, combining principles from Operations Research, phylogenetics, and machine learning, can push the boundaries of what is considered typical in both academic and industrial contexts. The work highlights that solving practical challenges often requires stepping outside conventional methodologies and exploring novel frameworks that address emerging issues in real-world systems.

Part II

Research Outcome

Chapter 2

Paper I

Topology Reconstruction using Time Series Data in Telecommunication Networks

Status: Published in *Networks* on November 2023

Authors: David Pisinger[a], Siv Sørensen[a]
[a] DTU Management, Akademivej, Building 358, DK 2800 Kgs. Lyngby

Keywords: Dynamic programming, Maximum parsimony, Network reconstruction, Phylogenetic trees, Telecommunication networks, Variable neighborhood search

Abstract

We consider Hybrid fiber-coaxial (HFC) networks in which data is transmitted from a root node to a set of customers using a series of splitters and coaxial cable lines that make up a tree. The physical locations of the components in a HFC network are always known but frequently the cabling is not. This makes cable faults difficult to locate and resolve. In this study we consider time series data received by customer modems to reconstruct the topology of HFC networks. We assume that the data can be translated into a series of events, and that two customers sharing many connections in the network will observe many similar events. This approach allows us to use maximum parsimony to minimize the total number of character-state changes in a tree based on observations in the leaf nodes. Furthermore, we assume that nodes located physically close to each other have a larger probability of being closely connected. Hence, our objective is a weighted sum of data distance and physical distance. A variable-neighborhood search heuristic is presented for minimizing the combined distance. Furthermore, three greedy heuristics are proposed for finding an initial solution. Computational results are reported for both real-life and synthetic network topologies using simulated customer data with various degrees of random background noise. We are able to reconstruct large topologies with a very high precision.

2.1 Introduction

Worldwide there are millions of kilometers of coaxial cables, used to carry cable television and internet signals. Leading companies like Shenyu Communication produce more than 1.3 million km of coaxial cables each year [130]. Although fiber net is being used more frequently for high bandwidth signals, coaxial cables are still used for shorter connections, and the market is expected to grow to $34.43 billion in 2023 [13]. Maintaining this infrastructure obviously also demands still more resources. In Denmark alone, the Danish infrastructure owner TDC-NET has more than 1000 technicians driving more than 80,000 km each day. In order to be able to quickly repair failures in the network and to maintain the infrastructure, it is essential to know the topology of the network.

We consider the *last-mile network*, connecting customers (modems) with a main hub though a singular point known as a Coaxial Media Converter (CMC). The last-mile network is frequently composed of coax cables, as opposed to the backbone grid, that typically is fiber-optic. Last-mile networks typically do not contain cycles, and hence can be described by a spanning tree from a hub (CMC) to the individual modems. With respect to The Open Systems Interconnection (OSI) model[68], which formally defines seven abstract layers of computer networking, our scope falls within the *physical layer* (layer-1). An example of a

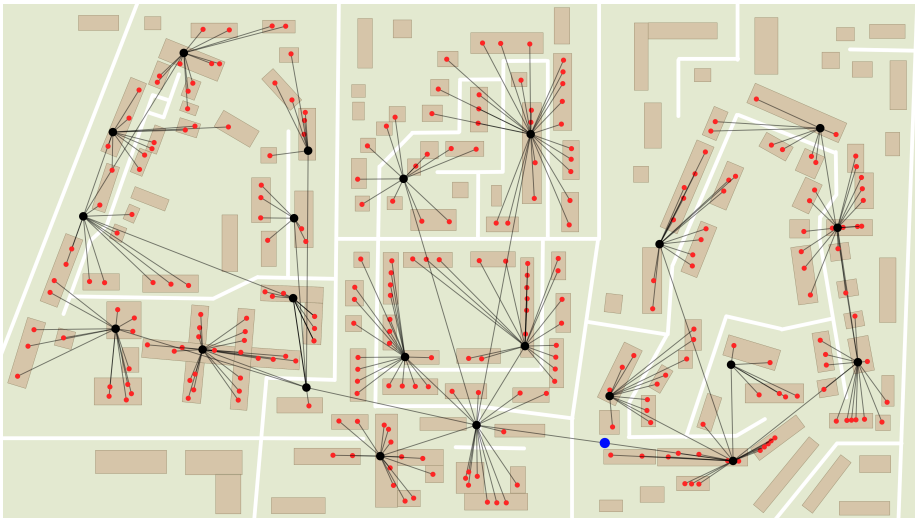


Figure 2.1: An example of a real life last-mile network. Red nodes are customers, black are splitters and blue is the CMC (root node). The cable connections are pictured as euclidean distances, but in reality they mostly lie alongside roads. For confidentiality reasons, the map behind the network is completely artificial, but bears resemblance with the real housing layout. The background is included to show how the network topology is shaped by the housing and road layout. The topology is also pictured without the background in Suppl. S2.

last-mile network is depicted in Figure 2.1.

In telecommunications networks we distinguish between two signal transmission types, namely downstream and upstream-transmissions. Downstream refers to data being sent from a network service provider to a customer, and vice versa for upstream. Each customer modem collects data about the transmission quality. This data includes, but is not limited to; modulation error, corrected code-words error ratio, latency, downstream power, etc. These metrics reflect the state of the network as errors propagate in the direction of the data stream. As an example, if a cable in the network has a loose connections, the quality of the downstream data will decrease for all customers whose connection to the CMC goes through this cable. Ideally, the modem data only reflects transmission noise, however, background noise that carries no useful information will inevitably also always be present in modem data.

While the backbone grid typically is quite well-documented, the topology of the last-mile network is frequently more dubious. This may have several causes: first of all, cables were mostly laid down in a time before digitalization, so much of the documentation is in paper-form and not accessible for daily planning. Furthermore, frequent mergers of telecommunication companies mean that some

information may have got lost. Finally, changes are often registered in a non-systematic manual way, meaning that several versions of the same topology documentation may exist [85, 27].

Instead of a time-consuming documentation, we will use data-mining techniques to identify the most probable cable topology of a network. For this purpose we will use data from the customer modems which measures the state of the *downstream-transmissions* at leaf-level, as well as the geographical location of all nodes, including customers, the CMC, and cable splitters. Transmission data is rarely available from internal devices such as amplifiers and cable splitters, either because these devices are passive and consequently unresponsive or because the cost of probing or storing historic data is too high[98, 85, 18].

We hypothesize that; the more similar the transmission signal is for two modems and the closer they are to each other in Euclidean distance, the more cable connections they probably share. The motivation for the latter term is that telecommunication companies in general try to minimize cable and digging costs, hence shorter cable connections are preferred.

Given our hypothesis, we choose the **objective** of our solution method to constitute of two terms: the first term minimizes the length of the total cabling of the topology which corresponds to the sum of the Euclidean distance of the edges in the spanning tree solution, and the second term minimizes the amount by which the signal transmission quality changes throughout the tree. Although we use Euclidean distances, any metric can be used. For instance, if it is known that the cables follow roads, then actual road-distances or Manhattan distances can be used.

The second term in the objective deserves further clarification. The logic behind the term originates from the field of phylogenetic inference in biology—the practice of inferring the true evolutionary relationships between a set of observations such as species—as the problem of reconstructing the history of related species is comparable to the problem of reconstructing the history of related signal transmissions.

In the evolution of species, a common ancestor evolves into multiple species over times through a series of evolutionary inherited changes. Similarly, the original state of the down-stream transmission signal in a last-mile network (at CMC level) changes as it travels through different parts and components of the network before reaching each customer. The structure of our problems thus naturally lends itself to phylogentic principles.

There exists various categories of evolutionary inference methods, each with their own advantages and disadvantages, with the most well-studied categories being: parsimony-, distance-, likelihood-, and Bayesian-methods. Common for all approaches is the limitation that there are no guarantees that the true topology will be the most probable/likely based on the input data. We refer the

inexperienced reader to [3] for a light introduction to the terminology and general concepts of phylogentic reconstruction and seasoned phylogeneticists to [44] for an extensive overview of the theory and algorithms or [97] for a more compact review.

Likelihood-, and Bayesian-methods are considered to be some of the more sophisticated reconstruction approaches, as both methods are based on explicit evolutionary (substitution) models and have gained popularity for their ability to incorporate various different aspects of the evolutionary process[39, 71]. However, this is also what makes these models unsuited to our application as we have very little understanding of the features and processes of evolution of our data.

Distances-based methods rely on constructing a symmetric distance-matrix that describe the pairwise dissimilarity between all observations (in our case modems). While distances-based methods are computationally fast, much time-dependent information is lost when compressing the available data into singular pairwise distance measures[39]. This makes Distances-based methods unsuited for problems with low sequence similarity[3], which is the case for the modem data we have at hand.

This leaves us with the maximum parsimony method, the oldest and most applied phylogentic method[3]. Although its popularity has seen a decline in recent years due to its simplicity and inability to incorporate sophisticated evolutionary models, maximum parsimony is nevertheless a good choice for our application as we cannot formulate an appropriate evolutionary model for our data at this stage.

Under the maximum parsimony criterion the number of evolutionary changes from the ancestor to its descendants is minimized. Thus, the criterion assumes that the simplest and most probable explanation for the differences in the observed data of the descendants, reflect reality. Take a street with ten customers connected to the same cable splitter as an example. Say that nine of the ten customer experience abnormal signal quality at time t. Then, it is more likely that there is a singular fault in the network *above* the cable splitter, affecting the whole street of customers, rather than nine customers experiencing abnormal signal quality independent of one another at the exact same time. Now recall that the second term of our objective *"minimizes the amount by which the signal transmission quality changes throughout the tree"*? This term is equivalent to the maximum parsimony criterion.

To quantify the amount by which the signal transmission quality changes, a discrete representation of the modem data is needed. Moreover, the transmission data must be reconstructed for all intermediate nodes (cable splitters and CMC). This way, the number of data mutations (equivalent to evolutionary changes) on each edge in a proposed tree can be counted using the Hamming distance. Figure

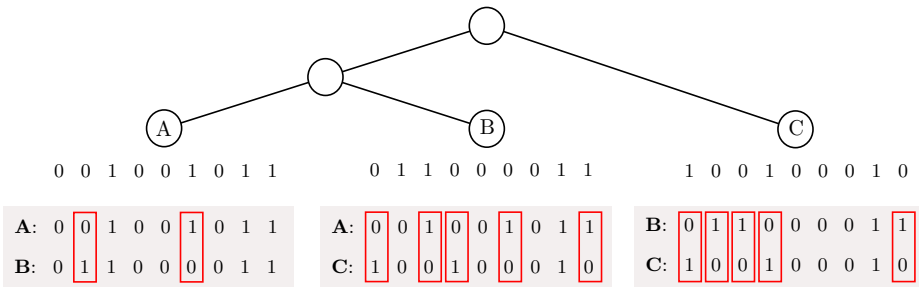


Figure 2.2: An example of a discrete binary representation of the modem data in a network with three customer nodes; A, B, and C, and two internal nodes. For each customer, the data sequence contains nine time steps. We observe that the Hamming distance, $h()$, between each pairwise customer is; $h(A, B) = 2$, $h(A, C) = 5$, $h(B, C) = 5$, as highlighted with red boxes. Thus, according to our hypothesis, customer A and B would be closer to one another (as pictured) than they would to C.
The pictured reconstruction is indeed optimal with respect to the maximum parsimony criterion.

2.2 illustrates how the Hamming distance can be used to compare pairwise discrete data sequences.

Although we employ a well-studied method from computational biology, we do not make the same general assumptions. As opposed to phylogenetics, in telecommunication networks homoplasy is not rare, data mutations are not irreversible along each branch, and symmetry is not tied to speciation rates (a measure of how many new species appear in an interval of time) but rather occur unintentionally and is thus random. In biology it is also commonly assumed that speciation events always produce two ancestors from one, meaning *binary* trees are predominantly considered. However, in our case we cannot impose such a constraint on the internal nodes, since a splitter typically have multiple connections. For the customer modems we though maintain the assumption that they have no children and represent all leaf nodes.

Lastly, it is important to point out that if the Euclidean distance term in the objective is removed, then focusing on the parsimony score alone would lead to an exponential number of equivalent mappings of the logical tree to the physical nodes. See Figure 2.3 for an illustration of equivalent logical trees.

The main **contributions** of this paper are:

- We present a data-mining approach inspired by phylogenetic algorithms used in Biology.
- The classic *minimum mutation tree problem* is extended to also include geographical information about the location of nodes. We consider a bi-objective problem minimizing both parsimony distance between two nodes,

and the Euclidean distance between nodes. The two objectives are merged using a scaling factor.
- We develop a *Variable Neighborhood Search* algorithm for finding the most parsimonious tree. The algorithm makes use of three graph-inspired neighborhoods.
- We give an algorithmic description of the maximum parsimony method by Hartigan [59] and analyze its time complexity.
- We present a vectorized implementation of the maximum parsimony method by Hartigan [59] that makes it possible to analyze large data-sets in reasonable time.
- We simulate modem events for synthetic network topologies and one real-life network topology and present a detailed computational study.

The overall results are promising, showing that we can reconstruct the topology of a last-mile network with few miss-predictions even for large networks with up to 30% data background noise.

The paper is **organized** as follows: The following section presents background information and provides an overview of related works. In Section 2.3 we formulate the problem in more details, explaining how data is collected at modem level and how transmission noise propagate through the network. Furthermore, we introduce the terminology used in the sequel. The section ends with an explanation of how to efficiently calculate the minimum parsimony score of a given tree. In Section 2.4, we describe a *variable neighborhood search heuristic* for optimizing the tree topology, as well as three different greedy construction algorithms. Section 2.5 presents the real-life and synthetic data instances used, both with respect to topology and modem data. Section 2.6 describes the computational experiments, where we first focus on perfect data to verify the

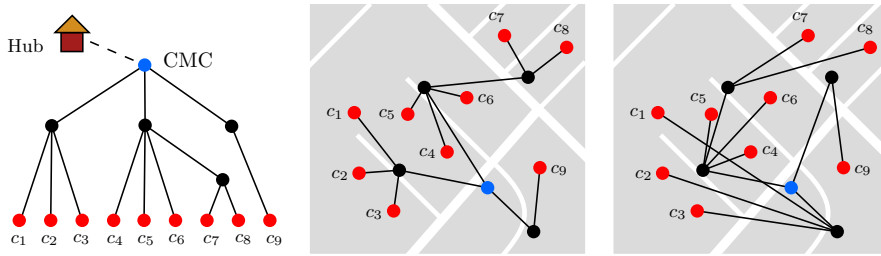


Figure 2.3: An illustrative example of how a logical tree (A) has exponentially many equivalent mappings to physical nodes, with respect to the number of inner nodes. Here, two such mappings are illustrated in (B) and (C). The transmission source node (CMC) is marked with blue, the cable splitters are marked with black, and the customers with red.

method, then study large scale problems of realistic sizes with different levels of background noise, and lastly compare the results to the real-life instance. Section 2.7 concludes the paper.

2.2 Background

The parsimony problem was first introduced by Camin and Sokal in 1965 [17] as a method for phylogenetic tree reconstruction. Many variations of the problem have since been explored and proven NP-hard [52, 38, 37] in many of its versions. The core difficulty of the problem lies in the task of having to evaluate all possible tree toplologies before the best can be selected with respect to a chosen criterion. Within the family of parsimony methods, efficient dynamic programming algorithms exist for scoring a given topology. The most established method is perhaps Fitch' algorithm [48] which was the first parsimony method for inferring ancestral states in binary trees. Fitch' algorithm is widely established within biology, where it is common to assume that all nodes are binary since genetic mutations are rare. Hartigan [59], independently of Fitch, developed an independent method which is applicable to general trees, but simplifies to Fitch' algorithm in the binary case. In telecommunication networks, most nodes are non-binary, and hence we apply the more general algorithm of Hartigan. It is, however, less studied, and mathematically analyzed.

Both Fitch' and Hartigan's algorithm solves the *Unweighted Rooted Small Parsimony Problem* in which it is assumed that the cost of mutating a data point from one state to another is the same for all possible states. The Sankoff's algorithm offers a solution to the weighted problem in which a cost is associated with each possible mutation [138]. Since our study will only considers binary data states, we have omitted the adaptation of variable mutations costs. Sankoff's algorithm could however readily replace that of Hartigan's in our methodology.

The maximum parsimony criterion bears strong resemblance to the *minimum evolution* (ME) criterion, a distance-based approach first proposed by[73]. Where maximum parsimony minimizes the number of evolutionary mutations, minimum evolution minimizes the total length of the tree topology based on preassigned edge weights. Despite the maximum evolution criterion being superior to that of maximum parsimony from a theoretical and statistical consistency-standpoint[136], the estimation of appropriate edge weights is still dependent on model parameters, typically substitution rates. Furthermore, in our problem we are also interested in inferring signal data at the internal nodes, since we speculate this can help us identify the source location of faults in a future project. This is possible with a maximum parsimony approach but not with ME. Catanzaro [19] presents a good overview on Minimum Evolution methods up until 2008.

In the literature, exact as well as heuristic [61, 96] solution approaches for inferring phylogenies have been developed. However, heuristic approaches are most frequently adapted with exact methods typically focusing on solving special cases of the general problem [2, 137] or small instances[71]. The main difficulty with a MIP-based approach is the size of the model—in particular the amount of binary (or integer depending on problem and the formulation) variables. Here, we refer to the set of 'optimal data states' which must be decided for each time step for all internal cable splitter nodes. Moreover, the problem is of a volatile nature, where the change of a single edge propagates upwards in the network and in the worst case affect all states between the edge and the root, giving rise to a significant change in the network parsimony score.

Within computational biology, successful MIP-based approaches for large input data rely heavily on the ability to reduce the model size. Characteristics such as *q-near-perfect phylogeny*—limiting the number of times a data-character can mutate to q, or *minimal homoplasy*—a character state shared by a set of observations (taxa) but not present in their common ancestor, make reductions of up to several orders of magnitudes possible in either the number of constraints or decision variables [149]. These types of characteristics exist in evolutionary biology where mutations are rare and typically unique, unlike in telecommunication networks. Thus, previously explored reduction techniques are unlikely to be successful within our scope. To the best of our knowledge no exact methods exists which can provide efficient solutions to problems of the same magnitude as ours.

The perhaps most related problem to ours is *Network Tomography* a relatively new field within computer science formally introduced in 1996[161]. Network tomography problems typically operate in the scope of layer-2 and layer-3 networks[68] such as Ethernet applications or path determination for network packets. Two of the key differences that set network tomography problems apart from our is the nature of the data involved and topology characteristics.

In network tomography active end-to-end (source-to-receiver) probing is used to measure packet loss and delays, while we in our problem rely on passive measurements at the customer nodes over a longer time horizon. The data obtained from active probing lends itself much more naturally to accurately estimate model parameters, resulting in numerous studies applying a wide range of phylogentic methods[28, 99, 27].

Within the related scope of *path determination* the overall logical network topology is typically known, and the task is to find out the routing topology for packets, which is a subset of the logical network. [29] considers a multi-source network and uncovers packet paths by merging the logical tree topologies obtained from considering only one source at a time when probing all receiver nodes.

Our problem also bears resemblance to the *Minimum Weighted Tree Reconstruction* (MWTR) problem, as studied by e.g. [51]. Here, a pairwise distance matrix between the set of leaf nodes is known, and the task is to find a tree and associated edge weights by introducing additional (internal) nodes, with the goal of minimizing the sum of all edge weights. The edge weight determination is complicated by the fact that the sum of edges on each unique path between all pairwise leaf nodes must be at least as much as the distance given by the pairwise distance matrix.

The paper [85], is to our knowledge, the study that comes closest to ours. The problem setting is the same; i.e. a layer-1 last-mile-network with assumed unknown cable topology, in which all intermediate nodes are unresponsive, and with the only difference being that the modem data is collected through active probing at carefully selected times when higher-level devices suggest some degree of network failure. The main objective of [85] is to infer *failure groups* (groups of modems likely to experience the same failures) to be used for localizing failures, identify failure root causes and suggest potential misconfigurations in records of any existing topology. A unsupervised learning algorithm based on non-negative matrix factorization is presented and results are based on both simulated and real network data in the form of a discrete binary representation that is identical to the data representation we will use in this paper.

Lastly, [85] attempt to incorporate the geographic location of nodes—just as we do in this paper—by including the longitude and latitude positions as two additional λ-scalable entries in the feature vector belonging to each node. However, the authors speculate that this approach will be hard to tune in practice.

2.3 Problem statement

In this section we formally define our problem and the corresponding objective function. Furthermore, we describe our dynamic programming approach for calculating the Hamming cost of a given tree. Finally, we show how the Hamming costs can be calculated using vectorized calculations.

2.3.1 Problem

Let $V = \{1, \ldots, n + n'\}$ be the set of nodes in a last-mile network T, where V can be split into two subsets $V_L = \{1, \ldots, n\}$ and $V_I = \{1, \ldots, n'\}$. V_L is the set of customer modems (*taxa*) to be analyzed, which corresponds to the *leaf nodes* in the network and V_I is the set of *internal nodes* (root and cable splitter nodes). Every node $v \in V$ is associated with an input of longitude and latitude coordinates that describe its location.

Let $M = \{1, \ldots, m\}$ be the studied time horizon. Every node $v \in V$ is, in addition to its location, associated with a sequence of observations (*characters*)

$\sigma_v = \{c'_{v1}, \ldots, c'_{vm}\}$. The data sequence is given as input for the leaf nodes (customers), while σ_v is unknown and thus variable for the internal nodes. The observations correspond to various metrics about the quality of the transmissions received by the customer modems at certain times.

A character c_{vk} can take on different values (*states*) in the *alphabet* Ω. In our case we have chosen two states $\Omega = \{A, B\}$, where A means that the modem data metric was normal at time $k \in M$, while B means the metric was abnormal. All of the following results can easily be generalized to a more fine-grained set of states Ω.

For an edge $(u, v) \in T$ between node u and $v \in V$, let $h(u, v)$ be the *Hamming distance* between sequence σ_u and σ_v. Recall that the Hamming distance between two sequences of equal length refer to the number of positions at which the sequences differ. Assume that a candidate last-mile network, i.e. an evolutionary tree T, is given as input and we want to calculate the minimum number of state-changes $z(T)$ (hereinafter *parsimony score*) needed to explain the data observed in leaf nodes of the tree. Then,

$$z(T) = \left\{ \begin{array}{lll} \min & \sum_{(u,v)\in T} h(u,v) & \\ \text{s.t.} & c_{uk} = c'_{uk} & \text{for} \quad u \in V_L,\, k \in M \\ & c_{uk} \in \Omega & \text{for} \quad u \in V_I,\, k \in M \end{array} \right\} \tag{2.1}$$

Since the characters are independent for each time slot $k \in M$ we can evaluate the time slots one by one, and successively add the cost of state-changes at time k for all M. This means that we can calculate $z(T)$ as the following sum

$$z(T) = \sum_{k \in M} z_k(T) \tag{2.2}$$

where the parsimony score for a single time slot $z_k(T)$ is given by

$$z_k(T) = \left\{ \begin{array}{lll} \min & \sum_{(u,v)\in T} h(c_{uk}, c_{vk}) & \\ \text{s.t.} & c_{uk} = c'_{uk} & \text{for} \quad u \in V_L \\ & c_{uk} \in \Omega & \text{for} \quad u \in V_I \end{array} \right\} \tag{2.3}$$

Let $\mathcal{T}$ be the set of all spanning trees, defined on the fixed set of nodes $V = V_L \cup V_I$. The *maximum parsimony problem* is to find a spanning tree $T \in \mathcal{T}$ in which the taxa are the set of leaf nodes, and the states in the interior nodes are chosen such that the following is minimized

$$\min_{T \in \mathcal{T}} z(T) \tag{2.4}$$

Since the number of possible trees are of magnitude $O(V!)$ for V nodes, an exhaustive search is not possible for problems of realistic size.

2.3.2 Objective function

As explained in our hypothesis in the introduction, the objective should not only consider signal similarity between nodes, but also the geographical distance. Let $d(u, v)$ be the physical distance between nodes $u, v \in V$. This can be the Euclidean, Manhattan, or actual street distance. For a given tree $T \in \mathcal{T}$ the cabling distance cost of the tree is

$$w(T) = \sum_{(u,v)\in T} d(u, v) \tag{2.5}$$

In our work we use a weighted sum of $z(T)$ and $w(T)$ to guide the search. Since the two objectives are of different scale (z is measured in Hamming distance, while w is measured in geographical distance) we first normalize the two terms individually, before computing the weighted sum. Let T' be the tree found by a greedy construction algorithm (see Section 2.4.2) and $\alpha \in [0, 1]$ a scaling factor. Then, our objective can be written as

$$\min \quad \alpha \frac{w(T)}{w(T')} + (1 - \alpha)\frac{z(T)}{z(T')} \tag{2.6}$$

In Section 2.6 we will study how the choice of α affects the solution, and propose proper values.

2.3.3 Dynamic programming

Since every time step $k \in M$ can be considered independently in $z_k(T)$ as given in (2.3), we will in the following consider a single character position.

For a node $v \in V$ let T_v be the subtree of T rooted in v. The corresponding subproblem is

$$z'(T_v) = \left\{ \begin{array}{lll} \min & \sum_{(u,w)\in T_v} h(c_u, c_w) & \\ \text{s.t.} & c_u = c'_u & \text{for} \quad u \in V_L \cap T_v \\ & c_u \in \Omega & \text{for} \quad u \in V_I \cap T_v \end{array} \right\} \tag{2.7}$$

Moreover, let the subproblem $z''(T_v, \omega)$ be the problem $z'(T_v)$ in which we have fixed the character at the root node v to $c_v = \omega$. With D_v defined as the set of direct descendant nodes of v and $\pi \in \Omega$ the variable character of a descendant $w \in D_v$, we can write up a recursive formulation of (2.7) as follows: for internal nodes $v \in V_I$ we have

$$z''(T_v, \omega) = \left\{ \min_{\pi\in\Omega} \sum_{w\in D_v} \left(h(\omega, \pi) + z''(T_w, \pi)\right) \right\} \tag{2.8}$$

and for leaf nodes $v \in V_L$ we have

$$z''(T_v, \omega) = h(\omega, c'_v) \tag{2.9}$$

As the optimal character assignment ω for a leaf node $v \in V_L$ is exactly its observed input character c'_v, (2.9) always evaluates to zero.

The running time of this algorithm is $O(E \cdot \Omega^2)$ and space complexity $O(V \cdot \Omega)$ since all values of $z''(T_v, \omega)$ must be stored. Here, E refers to the set of edges in T.

Hartigan's algorithm

Hartigan [59] observed that instead of calculating all values of $z''(T_v, \omega)$, one only needs to keep track on all the alternative states that result in the same optimal objective value of $z'(T_v)$. In other words, let R_v be defined as:

$$R_v = \{\omega \in \Omega : z''(T_v, \omega) = z'(T_v)\} \tag{2.10}$$

Instead of calculating the actual costs in (2.8) we can just choose the state $\omega \in \Omega$ that occurs most frequently in the sets R_v, where $(v, w) \in T_v$.

The Hamming cost at node v is defined as:

$$H_v = \sum_{w \in D_v} h(c_v, c_w)$$

For the base case $v \in V_L$, we can set:

$$\begin{aligned} R_v &= \{c'_v\} \\ H_v &= 0 \\ z'(T_v) &= 0 \end{aligned} \tag{2.11}$$

To evaluate recursion (2.8) we consider the nodes in T from leaves to the root. For a given node v we first calculate the frequency of each state $\omega \in \Omega$

$$f_v(\omega) = |\{w \in D_v : \omega \in R_w\}| \tag{2.12}$$

Let $F_v = \max_{\omega \in \Omega} f_v(\omega)$ be the most common frequency of a state. Then we get the recursion:

$$\begin{aligned} R_v &= \{\omega \in \Omega : f_v(\omega) = F_v\} \\ H_v &= |D_v| - F_v \\ z'(T_v) &= H_v + \textstyle\sum_{w \in D_v} z'(T_w) \end{aligned} \tag{2.13}$$

The logic behind the expression for $z'(T_v)$ is that the Hamming distance is increased by one for each state-mutation, i.e. for each node $w \in D_v$ that does not have a state in common with parent node v. Thus, choosing the optimal

state(s) of v among the most frequent state(s) of the descendants, R_v, yields the smallest addition in Hamming distance H_v.

It can be seen that the time complexity of (2.12) and (2.13) is $O(\Omega \cdot D_v)$ for every node $v \in V$. This means that the overall time complexity is $O(\Omega \cdot \sum_{v \in V} D_v)$, which is equivalent to $O(E \cdot \Omega)$ since the network is a tree.

The space complexity is $O(V \cdot \Omega)$ since for every node v we must store a set R_v of size up to $|\Omega|$.

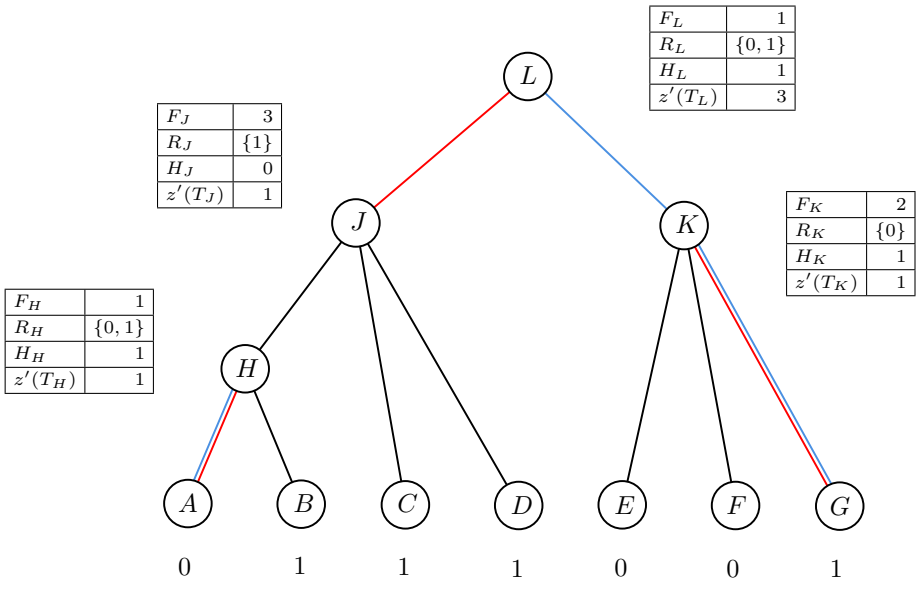
Since we only considered a single state $k \in M$ in the above recursion, we need to repeat the above for all time steps $k \in M$ to find the total parsimony score $z(T)$ as given by (2.1). This means that Hartigan's algorithm has time complexity $O(M \cdot E \cdot \Omega)$ and space complexity $O(M \cdot V \cdot \Omega)$.

Example

We now demonstrate how Hartigan's recursive algorithm works on a small example topology with seven customers for a single binary data point in time. The input data point for each customer is:

Modem v	A	B	C	D	E	F	G
Character c'_v	0	1	1	1	0	0	1

On the topology below we show the intermediate calculations for all nodes from leaves to root, corresponding to Hartigan's algorithm (2.11)-(2.13).



The minimum number of state-mutations is $z'(T_L) = 3$. This optimal score can be obtained through two combinations of character states selected from the sets R_v, $v \in V_I$. On the topology above, we have colored the edges on which a state mutation occurs *red* or *blue*, corresponding to selecting either state '0' or '1' in R_L, respectively.

Although the presented methodology returns the minimum number of mutations $z'(T_v)$, it is important to note that the definition of the optimal state set R_v in (2.13) only yields a subset of the states that all result in the same optimal objective value. Hartigan [59] presents the full definition for the optimal state set, detecting all optimal combinations of character states. Since only the minimum mutation score is needed in our work, we can simplify the definition of R_v.

Fast vectorized Parsimony

To efficiently implement Hartigan's algorithm, we need to represent the sets $R_v \subseteq \Omega$ in equation (2.10). A space-efficient way is to use a bit-mask. Each bit position corresponds to a state $\omega \in \Omega$, where a value of 1 means that the state is part of set R_v. In our case the size of the alphabet Ω is two, so we can use a single byte to represent a set R_v. Equation (2.12) now counts the number of bits at each position, and the recursion (2.13) selects the bits with highest frequency.

The recursion lends itself well to a vectorized representation. For a given node $v \in V$, assume that each character c_{vk} corresponding to time step $k \in M$ can be represented by a byte, then we can compute the recursion (2.13) for 64 characters (time steps) in parallel using standard AVX-512 instructions [67]. As we will see, this makes it possible to analyze large trees in very reasonable time.

2.4 Solution method

Although we presented an efficient algorithms for calculating the parsimony score of a given tree in the previous section, we must still search among all possible trees to find the most parsimonious one. Therefore, we present a variable neighborhood search heuristic, using three graph-based neighborhoods, to explore the solution space. To ensure a good initial solution we also present three construction heuristics based on minimum spanning trees and clustering of related nodes.

2.4.1 Variable Neighborhood Search heuristic

Our solution method is a *Variable Neighborhood Search* (VNS) [93] heuristic with three types of neighborhood functions. The heuristic follows a classic structure:

first, an initial solution is generated; second, one of the three neighborhood functions is selected at random and used to alter the current solution; third, the solution change is evaluated and either accepted or rejected based on an acceptance criterion which considers both the total cabling and the network parsimony score; fourth, the second and third steps are repeated until the algorithm is terminated. Pseudo-code for the heuristic is shown in Algorithm 2.1. We briefly discuss the possibility of extending the approach to an ALNS [133] in Section 2.6.

Algorithm 2.1 VNS heuristic

```
1: input: V nodes, their location, and data for v ∈ V_L
2: repeat
3:     x ← GreedyConstruction();
4:     repeat
5:         select neighborhood function f ∈ {N1, N2, N3} with equal probability;
6:         x' ← f(x);
7:         if accept(cost(x'), cost(x)) then
8:             x ← x';
9:     until stop criterion is met
10: until number of restarts is met
11: return best found x
```

2.4.2 Initial solution

We consider three different approaches for constructing an initial solution and evaluate their performance individually as well as in the context of the entire solution procedure. Each of the three construction methods consists of two phases as described below.

2.4.2.1 Minimum Spanning Tree (MST)

This heuristic is based on the assumption that a MST is a fair approximation of the true topology given the costs associated with cabling.

In phase one, a directed MST rooted in the predefined root node (CMC), is build over the internal nodes $v \in V_I$. Here, we employ Kruskal's algorithm [75] where the cost of an edge, $d(w, u)$, is the Euclidean distance from node w to u.

In phase two, the remaining leaf nodes $v \in V_L$ are each connected to the closest splitter-node in the MST. The complexity of Kruskal's is $O(V_I^2 \log V_I)$ and connecting the leaves can be done in $O(V_L \cdot V_I)$.

2.4.2.2 Greedy Spanning and Parsimony Tree (GSPT)

We consider a greedy spanning tree where the cost between each pair of nodes now consists of both the Hamming distance as well as the cable length, as opposed to only the cable length.

In phase one, all leaf nodes are attached to a splitter. First, the leaf nodes (customers) are sorted in ascending order based on their shortest distance to a splitter-node. The leaf nodes are then connected to a splitter-node one-by-one in the order of the sorted list, using a cost function which determines the distance and increase in parsimony score of adding node u to v:

$$g(v,u) = \beta\, d(v,u) + (1-\beta)\frac{h(v,u)}{|M|} \tag{2.14}$$

Here, $\beta \in [0,1]$ is a scaling factor to be determined. The leaf nodes are attached in the sorted order due to the assumption that the closer a customer is to a splitter, the more likely it is that they are directly connected. The first customers attached to a splitter will create the initial *majority vote* in terms of data states, which will influence the subsequent customer assignments. Thus, the more accurate the first customer assignments are, the better. The complexity of phase one is $O(V_L \cdot V_I \cdot M)$, where the modem sorting can be done in $O(V_L \log V_L)$ and the modem insertion can be done in $O(V_L \cdot V_I \cdot M)$.

In phase two, the internal nodes are connected in a spanning tree manner as well. We consider all possible directed edges between all pairs of internal nodes using the cost function (2.14) for the assignment of node v as the parent of u. The initial edge costs can be found in $O(V_I^2\, M)$. In each iteration the cheapest edge is added and subsequently the parent node of the inserted edge is updated with respect to its cost and string of optimal states. A new edge is added until all nodes are connected by a tree rooted in the predefined root. The iterative process can be done in $O(V_I\,(V_I \cdot M + \log V_I))$ when using a min-heap for the edge costs. This amounts to $O(V_I^2\, M)$ for all of phase two.

2.4.2.3 Cluster Spanning and Parsimony Tree (CSPT)

Instead of greedily assigning customers to splitters one by one, we first cluster the customers together based on their similarity in location and data, before building a greedy parsimony- and distance-based spanning tree from the clusters.

In phase one, $C = V_I - 1$ clusters are initialized, each using the data and location of a modem selected at random. Applying k-means clustering, all the leaf nodes $u \in V_L$ are assigned to their nearest cluster $i \in C$ using the cost function $g(i,u)$ from (2.14), where i refers to the location and data means of cluster i. The cluster means are then updated and the leaf nodes reassigned. This process is repeated until the assignment does not change. Assuming the number of iterations is constant, the complexity of the k-means clustering is

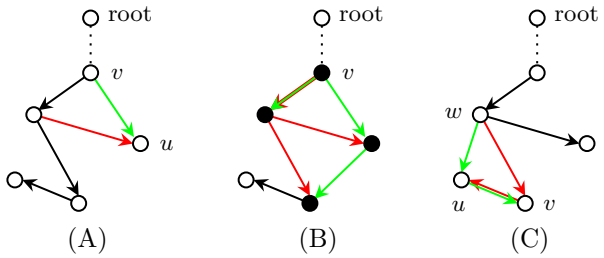


Figure 2.4: An illustrative example of each of the three neighborhood functions. Black edges are unchanged, green edges are new, and red edges are removed. (A) *Random swap*. (B) *Sub-tree shuffle* with sub-tree size $q = 4$ marked by black nodes. (C) *Parent-child swap*.

$O(V_L \cdot V_I \cdot M)$. Finally, each of the C clusters are assigned to one of the $V_I - 1$ splitter-nodes by minimizing the sum of cabling. The cluster assignment can be done in $O(V_L \cdot V_I)$, totaling the complexity of phase one to $O(V_L \cdot V_I \cdot M)$.

In phase two, the internal nodes are connected. The process is identical to phase two in GSPT. Since k-means clustering is sensitive to the cluster initialization points, CSPT is repeated V_L times and the best tree according to (2.6) (where T' is the solution from the first repetition) is returned as the initial solution. With repetitions, the whole of CSPT runs in $O(V_L^2\, V_I\, M)$.

2.4.3 Neighborhoods

To give the heuristic flexibility to move around in the search space, we consider three neighborhood functions: *random swap*, *sub-tree shuffle*, and *parent-child swap*. *Random swap* can move both internal and leaf edges, whereas *sub-tree shuffle* and *parent-child swap* only affect internal edges. *Random swap* affects a single edge, where *sub-tree shuffle* and *parent-child swap* always affect a subset of linked edges. A *parent-child swap* can occur within a *sub-tree shuffle*-move—given a set of lucky choices—but we found that including *parent-child swap* as a separate neighborhood function, was beneficial to the heuristic. All neighborhood moves can be performed in constant time—excluding objective evaluation.

2.4.3.1 Random swap

Random swap replaces an existing edge with a new random edge. Given the choice of a new edge is random, any edge can be removed or created. Thus, this neighborhood allows the heuristic to explore most of the solution space.

First, a node $u \in V$ and a node $v \in V_I$ are selected at random. Second, a new directed edge from v to u is added. Node u now has two parent nodes

and to rectify the solution, the edge to the former parent is removed. If a *random swap* disconnects the network, the move is reverted and replaced by a new *random swap*. Figure 2.4 (A) illustrates a *random swap* where the new green edge replaces the red edge.

2.4.3.2 Sub-tree shuffle

Sub-tree shuffle selects a random sub-tree of size q and shuffles its edges to form a new sub-tree. Since an improvement in the objective value (2.6) could require multiple edges to change in unison, the search extent of *random swap* alone is limited. *Sub-tree shuffle* thus extends the search capability of the heuristic.

First, a node $v \in V_I$ is selected at random to represent the root of a sub-tree. Second, the size of the sub-tree $q \in [3, 5]$ is selected at random. For the selection of the nodes composing the sub-tree, we consider all nodes $u \in V_I$ which are direct descendants of a node within the sub-tree—initially only the children of root v. Nodes are selected at random from this set until the sub-tree is of size q; if a sub-tree of size q cannot be build, a new *sub-tree shuffle* move is performed. The edges in the sub-tree are removed and replaced by $q - 1$ new randomly selected edges which form a new sub-tree still rooted in v. The subtree size q is chosen in the range of $[3, 5]$ because a subtree with less than three nodes only contain one edge, and because for larger subtrees it becomes increasingly more unlikely that the new randomly reassembled subtree is an improvement due to the exponentially increasing number of possible subtree reconstructions.

Figure 2.4 (B) illustrates a *sub-tree shuffle* move where a subtree of size $q = 4$ is selected (black nodes) and the initial subtree (red edges) are replaced by a new subtree (green edges).

2.4.3.3 Parent-child swap

Parent-child swap randomly selects a node u and makes it the parent of its current parent node, thus reversing the parent-child roles. This neighborhood function is a special case of the *sub-tree shuffle* where $q = 3$ and where both the initial and reconstructed subtree must have a depth of two. Through experiments it was observed that a *parent-child swap* move frequently could improve the objective, and by including it as an independent neighborhood function, better solutions were obtained through less iterations on average.

First, a node $u \in V_I \setminus \{\text{root}\}$ is selected at random. The parent-node of u is denoted v, and the parent of v is denoted w. Next, the directed edge e_{vu} is flipped such that conversely u becomes the parent of v. To preserve the tree structure, the edge e_{wv} is replaced by an edge e_{wu}, such that w is now the parent of u. Figure 2.4 (C) illustrates the described *parent-child swap* move.

2.4.4 Solution evaluation and acceptance

A candidate solution is either accepted or rejected based on the objective function (2.6). We use a greedy approach in which a new solution is only accepted if it improves the current solution. A simulated annealing acceptance criteria has also been explored, but yielded no improvement compared to the greedy criteria. We discuss this in more details in Section 2.6.

As calculating the parsimony score is based on a leaf-to-root dynamic programming procedure, only a subset of nodes will be subject to a potential state change, when one or more edges in the solution are changed. This set includes all nodes attached to either a new edge or a removed edge, as well as all of the ancestors of these nodes. In the worst case a new solution evaluation will still have to reevaluate the character states at all internal nodes. However, on average this approach is faster than recalculating the tree score from scratch.

2.5 Instances

We consider 15 synthetic network topologies, denoted by the letters A-Q, and one real-life network from Denmark denoted by the letter R (Figure 2.1). Each topology is associated with five instances of varying degree of background noise, totaling 80 instances. An instance is generated in three stages: first, the complete network topology, second, the modem data, i.e. transmission noise, third, background noise. We use two methods for generating the network topologies while the method for simulating data—both transmission and background noise—is the same for all instances. Although the real-life instance R corresponds to a real last-mile network topology, the data in the real-life instance is also simulated. That is because using real life downstream data would require extensive preprocessing such as; parameter selection, filtering, data transformation into discrete events, etc. Thus, we consider the task of preparing the input modem data as outside the scope of this work.

The following assumptions hold for all synthetic instances and are based on trends observed in real telecommunication networks:

- Every splitter node is the immediate parent of at least three customer nodes.
- The root node is not the immediate parent of any customer nodes.
- No cables are crossing.

An overview of the instance enumeration used throughout the paper as well as key instance parameters are shown in Table 2.1. The 15 simulated topologies are visualized in Suppl. S1 and the real life topology is visualized using the same format in Suppl. S2.

Below we present the two network simulation methods as well as the approach for

Table 2.1: An overview of the 16 network instances. The instances consist of 15 unique synthetic networks (denoted by letters A-Q) and one real network (denoted by letter R) each containing five variations of $0-40\%$ background noise (BN). Here, $|V_L|$ is the number of costumers, $|V_I|$ the number of splitters incl. the root, and NT the network topology simulation type. All instances contain 3200 binary data points per costumer. The primary sorting of the synthetic instances are based on the number of costumer and the secondary sorting is based on the level of background noise.

Inst.	A0	A1	A2	A3	A4	B0-B4	C0-C4	D0-D4	E0-E4	F0-F4	G0-G4	H0-H4
BN	0	10	20	30	40	...	...	...	...	...	...	...
$\|V_L\|$	100					125	150	175	200	225	250	275
$\|V_I\|$	15					14	18	16	25	23	30	27
NT	2					1	2	1	2	1	2	1

Inst.	J0-J4	K0-K4	L0-L4	M0-M4	N0-N4	P0-P4	Q0-Q4	R0-R4
BN	...	...	...	...	...	...	...	...
$\|V_L\|$	300	325	350	375	400	450	500	311
$\|V_I\|$	33	27	38	30	42	44	50	24
NT	2	1	2	1	2	2	2	Real

simulation modem data, i.e. transmission noise, as well as random background noise.

Network topology method 1 (NT1)

First, the network of splitters is grown *organically* outwards from the root. One to three splitters are randomly placed in an annulus around the root and then directly connected by an edge. For each new splitter $v \in V_I$ we place and connect zero-three new splitters at random in a predefined sized section of an annulus around v furthest away from the parent edge of v. This procedure is repeated until a desired amount of splitters is placed and connected. Second, a random number of customer nodes is placed and connected in a circular area around each splitter node, summing to a predefined number of customer nodes $|V_L|$. With this approach, the vast majority, but not all, of the customer nodes will be connected to their closest splitter reflecting real-life. Figure 2.5 (a) illustrates a network topology generated using NT1.

Network topology method 2 (NT2)

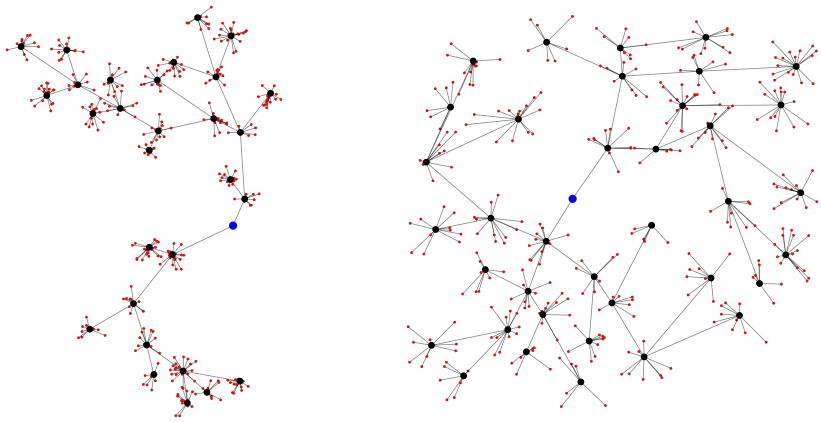
First, V_I nodes are placed in random locations representing the splitters and the root. One node is selected as the root and added to a list. The first node in the list is removed and assigned as the parent node of the either one, two, or three closest splitter nodes chosen at random. The newly connected nodes are added to the list and the procedure is repeated until all splitters are connected. Second, the customer nodes are added in the same way as described for Network

topology method 1. Figure 2.5 (b) pictures a network topology generated using NT2.

Modem data

All 3200 modem data points are initialized with a 0-bit. At a data-collection rate of 15 min (used by the infrastructure owner in Denmark), 3200 data points corresponds to roughly a month of data. For each time step $k \in M$ we select a random sub-set of internal edges $E'_I \in E_I$ of a size in $[2, V_I/10]$ and simulate that an abnormal event occurred on these edges. Such an event could be a glitch in a faulty cable connection, a consequence of road maintenance, or similar. Consequently, the transmission noise received by all modems in the sub-tree below an edge $e \in E'_I$ will be intensified and we flip the bit to represent this change at time k. On average, the zero to one bit ratio becomes approximately three to one for all customers in an instance using this procedure, meaning approximately 25% of the data points correspond to an abnormal transmission denoted by a 1-bit. By only introducing events on internal edges, modems that share the same parent node will have identical data.

Background noise is introduced by flipping a random set of bits for each customer. For a level of 20% background noise the size of the random set is $0.2\,|M|$ and is chosen anew for each customer.



(a) Network M (NT1) **(b)** Network L (NT2)

Figure 2.5: (a) pictures a network generated with NT1 and (b) pictures a network generated with NT2. NT1 grows a tree outwards from the root, splitter node by splitter node, whereas NT2 first randomly places all splitter nodes and then subsequently connects them.

2.5.1 Solution validation

There are two apparent ways in which we can validate the topology predictions made by our proposed method.

1. We can calculate the objective value (2.6) of the true topology and find the gap between this value and the objective value of the predicted topology.
2. We can compare the edges in the true topology to the edges in the predicted topology and count how many differ.

Although the first approach is considered standard practice within operations research, this measure cannot be used alone. This is because it is not guaranteed that the true topology will yield the lowest objective value out of all possible tree topologies—especially not for high levels of background noise. This is especially problematic when tuning the scaling factor α in the objective (2.6). For an α-value of one, a MST topology will yield the best possible objective value leading to a negative solution gap. Although this solution is good with respect to the solution gap, it might be very different from the true topology we want to uncover. For this reason, we use a validation method based on the second approach to tune the algorithm and to infer the overall prediction quality.

Rather than simply counting the number of wrong edges in a prediction, we evaluate a candidate solution $\hat{T} = (V, \hat{E})$ by quantifying the deviation γ. This process differentiates between two types of solution edges:

$$\gamma = \frac{|E_I - \hat{E}_I|}{|E_I|} + \frac{|E_L - \hat{E}_L|}{|E_L|} \tag{2.15}$$

Here, E_I and E_L denote the set of internal and leaf edges, respectively, such that $E_I \cup E_L = E$ and $E_I \cap E_L = \emptyset$. If the candidate topology is identical to the true topology, the deviation γ will be zero, and if all edges in the candidate topology are different from the true topology, γ will equal to two. Given that T is a general tree, γ penalizes a wrongly predicted internal edge more than a leaf edge. This is reasonable since a wrongly predicted internal edge affects a subset of customers, whereas a leaf edge affects only one customer.

For an edge to be considered correct, both its endpoints as well as its direction must be identical to an edge in the true network topology.

2.6 Computational experiments

The computational experiments are divided into three parts. First, we establish the best case effectiveness of the proposed heuristic by considering the synthetic instances without background noise only. Second, we evaluate how sensitive the heuristic is to background noise which is inevitable in real-life data. Third, we compare the quality of the synthetic results to that of the real instance. The two

Table 2.2: Each construction method ranked from one to three based on their deviation γ for each of the 15 synthetic instances without background noise. GSPT yields the best results on average.

		Instances															
Method	β	A0	B0	C0	D0	E0	F0	G0	H0	J0	K0	L0	M0	N0	P0	Q0	Avg
MST	-	3	3	2	3	2	3	3	3	3	3	3	2	3	1	3	2.67
GSPT	0.2	1	1	3	1	1	1	1	2	1	1	1	1	1	3	1	1.33
CSPT	0.3	2	1	1	1	3	2	2	1	2	2	2	3	2	2	2	1.87

Table 2.3: The deviation γ for each of the 15 synthetic instances without background noise using the three construction methods only. GSPT yields both the best average rank (table 2.2) as well as the best average γ .

		Instances							
Method	β	A0	B0	C0	D0	E0	F0	G0	H0
MST	-	0.52	0.478	0.543	0.64	0.478	0.5	0.694	0.687
GSPT	0.2	0.143	0	0.782	0	0.418	0	0.379	0.303
CSPT	0.3	0.213	0	0.235	0	0.607	0.167	0.547	0.24

Instances							
J0	K0	L0	M0	N0	P0	Q0	Avg
0.685	0.522	0.693	0.432	0.786	0.642	0.777	0.61
0.56	0	0.414	0.172	0.146	0.738	0.43	0.30
0.635	0.323	0.461	0.478	0.473	0.668	0.599	0.38

< 0.1
< 0.2
< 0.35
< 0.5
< 0.7
< 1.0
≤ 2.0

first parts follow the same general structure: an analysis of the three proposed construction heuristics followed by an analysis of the construction heuristics in combination with the neighborhood search, i.e. the full heuristic.

All tests are conducted with 200,000 neighborhood moves repeated for three restarts on a 2.6 GHz Intel Core i5 PC with 16 GB of RAM, in less than five minutes. All scaling factors introduced in Section 2.3.2 are tuned using 11 values in the interval $[0, 1]$ with a step size of 0.1. All tuning tests and results are included in Supplement S3. S4, and S5.

2.6.1 Method validation

Results discussed in the following sections are only based on the 15 synthetic instances with no background noise. This is to verify that the proposed method in fact works. Moreover, once background noise is added, we can quantify how much it affects prediction accuracy, as the results presented in this section serve as a best-case baseline for predictions.

2.6.1.1 Construction Heuristics

In order to compare the three proposed methods for generating an initial solution (MST, GSPT, and CSPT), the optimal value of the scaling factor β in (2.14) must be determined for GSPT and CSPT, respectively. First, all combinations of instances and potential values of β are computed for both methods. Then, for each instance, every value of β is assigned a rank from $1-11$ based on their deviation γ (see Suppl. S3 Table S-9). The β value with the lowest rank when averaging over all 15 instances is selected. For GSPT we select $\beta = 0.2$ and for CSPT $\beta = 0.3$.

Table 2.2 reports the rank of each of the three tuned construction heuristics based on their deviation γ. On Average, GSPT yields the best rank. Table 2.3 reports the actual deviation values γ for which GSPT returns an average of 0.30. This corresponds to 7% of all edges in a GSPT solution being wrong on average; 25% of the internal edges and 5% of the leaf edges. In four of the instances GSPT finds the true network topology.

2.6.1.2 Full algorithm

There are three possible variations of the full neighborhood search heuristic—one for each construction method. For each variation we tune the best value of the scaling factor α in the objective (2.6) based on the ranking approach explained above (see tuning in Suppl. S4 Table S-9). We select $\alpha = 0.5$ when initializing the full algorithm with MST, $\alpha = 0.4$ with GSPT, and $\alpha = 0.6$ with CSPT.

Table 2.4 reports the rank of each of the three tuned variations of the full algorithm based on their deviation γ. On Average, using the MST construction heuristic yields the best rank and is thus ruled the best version of the algorithm for instances without background noise. We denote this version of the algorithm $Algo^0_{opt}$. Table 2.5 shows that $Algo^0_{opt}$ returns the true topology for twelve of the fifteen instances and near-perfect prediction for instance C0 and N0. The worst prediction performance of $Algo^0_{opt}$ occurs in instance J0 with a deviation $\gamma = 0.193$ corresponding to 5/32 internal edges and 11/300 leaf edges being wrongly predicted.

We turn to the solution gap values to improve our understanding of the results. The two near-prefect predictions, C0 and N0, both correspond to a small negative solution gap greater than -0.3%, i.e. a better solution than the true solution. Thus, considering new neighborhoods, an ALNS approach or a new acceptance criteria will not lead to better predictions for these instances. Instead, the negative solution gaps suggest that the objective (2.6) is not a perfect analogy for topology prediction, since a perfect analogy would yield the best possible objective for only the true topology. Instance J0 returns a positive gap of 6%. Here, two sets of *sibling*-customer nodes of size around 5 are each

Table 2.4: Each variation of the full heuristic ranked from one to three based on their deviation γ for each of the 15 synthetic instances without background noise. Each variation uses the optimal β value in the construction phase. The variation which uses a MST initialization yields the best results on average.

		Instances															
Construct	α	A0	B0	C0	D0	E0	F0	G0	H0	J0	K0	L0	M0	N0	P0	Q0	Avg
MST	0.5	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1.07
GSPT	0.4	1	1	3	1	3	1	1	1	3	1	3	1	1	2	2	1.67
CSPT	0.6	3	3	2	3	1	3	3	3	1	3	2	3	3	3	3	2.60

Table 2.5: The deviation γ for each of the 15 synthetic instances without background noise using the three variations of the full search heuristic. An initial MST solution yields both the best average rank (Table 2.4) as well as the best average deviation $\overline{\gamma}$.

		Instances							
Method	α	A0	B0	C0	D0	E0	F0	G0	H0
MST	0.5	0	0	0.059	0	0	0	0	0
GSPT	0.4	0	0	0.427	0	0.158	0	0	0
CSPT	0.6	0.143	0.287	0.393	0.118	0	0.166	0.62	0.077

Instances							
J0	K0	L0	M0	N0	P0	Q0	Avg
0.193	0	0	0	0.024	0	0	0.02
0.341	0	0.333	0	0.024	0.417	0.081	0.12
0.062	0.311	0.098	0.191	0.073	0.525	0.635	0.25

< 0.1
< 0.2
< 0.35
< 0.5
< 0.7
< 1.0
≤ 2.0

connected to the wrong splitter along with five internal edges being misplaced. We speculate that either multiple leaf edges or a combination of leaf and internal edges would have to change within the same neighborhood move, for the objective value—specifically the parsimony score—to decrease. Either of such described moves do not exist within the scope of the three current neighborhood functions.

An interesting observation is that the MST construction heuristic on its own yields significantly worse predictions with respect to the deviation γ than both GSPT and CSPT as seen in Table 2.3, but in the context of the full algorithm the MST construction is best. Although MST is the worst of the three construction methods with respect to internal edge prediction, MST is the best of the constructors at predicting leaf edges; note that this is one-to-one correlated with the percentage of customer connected to their closest splitter (97.6% on average for all instances). This suggests that the neighborhood search is more proficient at correcting internal edges than leaf edges.

2.6.2 The impact of background noise

Results discussed in the following sections are only based on the 60 synthetic instances with non-zero background noise (see Table 2.1 for reference). Here, we aim to explore to what extend our proposed method could work for real transmission data where background noise is inevitable. We repeat the analyses performed in Section 2.6.1.

2.6.2.1 Construction heuristics

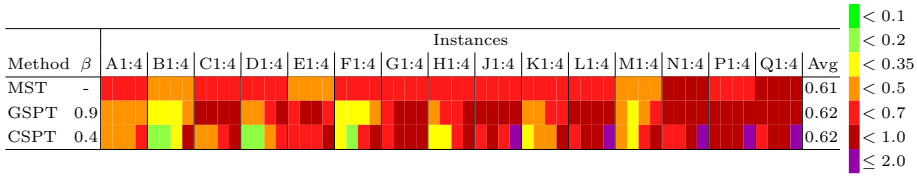
When considering the construction heuristics individually, we now find that the best value of the scaling factor β in (2.14) is 0.9 and 0.4 for GSPT and CSPT, respectively (see Suppl. S3 Table S-9), as opposed to 0.2 and 0.3 for the instances with no background noise from before. Without background noise, the optimal scaling factor for GSPT of 0.2 favored the parsimony over the distance contributions in (2.14). With background noise, the trend has changed and GSPT almost solely focuses on the cable distances.

Table 2.6 reports the rank of each of the three tuned construction methods based on their deviation γ. On Average, MST yields the best rank. When the construction methods were ranked for data without background noise, the MST method ranked the worst of the three methods. Table 2.7 reports the actual deviation values for which MST returns an average of $\gamma = 0.61$, about double the score obtained with GSPT for instances without background noise. This corresponds to 8% of all edges in a MST solution being wrong on average; 58% of the internal edges and 2.4% of the leaf edges. When only considering

Table 2.6: Each construction method ranked from one to three based on their deviation γ for each of the 60 synthetic instances with background noise. MST yields the best rank on average.

		Instances															
Method	β	A1:4	B1:4	C1:4	D1:4	E1:4	F1:4	G1:4	H1:4	J1:4	K1:4	L1:4	M1:4	N1:4	P1:4	Q1:4	Avg
MST	-	3 3 3 2	3 3 3 2	2 2 1 1	3 3 3 1	1 1 1 1	3 3 2 2	1 1 1 1	3 3 3 1	2 1 1 1	3 3 3 1	2 1 1 1	3 3 2 1	3 2 1 1	1 1 1 1	2 1 1 1	1.85
GSPT	0.9	2 2 1 1	2 2 2 1	3 3 3 2	2 2 2 3	3 3 3 2	1 2 1 1	3 2 2 2	2 2 2 2	3 3 3 2	2 2 2 2	3 3 2 2	1 2 1 2	2 3 3 2	3 3 2 2	3 3 2 2	2.18
CSPT	0.4	1 1 2 3	1 1 1 3	1 1 2 3	1 1 1 2	2 2 2 3	2 1 3 3	1 3 3 3	1 1 1 3	1 2 2 3	1 1 1 3	1 2 3 3	2 1 3 3	1 1 2 3	2 2 3 3	1 2 3 3	1.95

Table 2.7: The deviation γ returned by each of the three constructions methods individually for each of the 60 synthetic instances with background noise. The average deviation γ is almost the same for all methods, however the prediction quality of CSPT is more fluctuating.



the average deviation γ over all instances, all three construction methods have similar performance of $\overline{\gamma} \in [0.61, 0.62]$, however, the prediction quality of CSPT is fluctuating more.

2.6.2.2 Full algorithm

The best scaling factor in (2.6) for the three variations of the full neighborhood search heuristic is found to be $\alpha = 0.1$ using both a MST and a GSPT construction and $\alpha = 0.3$ for a CSPT construction. This is surprising since the optimal value of α was around 0.5 for all variation of the full algorithm with respect to the instances without background noise. We hypothesized that given the data quality decreases when adding background noise, the algorithm would have to depend more on the cable lengths. However, when background noise is added, the parsimony score of a network increases substantially. Since the objective is normalized by the initial solution, the second term of the objective is consequently divided by a substantially larger factor. Therefore, α has to be decreased to make contributions from improvement in parsimony visible.

Table 2.8 reports the rank of each of the three tuned variations of the full algorithm based on their deviation γ. On Average, using the MST construction heuristic yields the best rank and is thus ruled the best version of the algorithm for instances with background noise, just as it was for instances without background noise. We denote this version of the algorithm $Algo^1_{opt}$. Table 2.9 shows that $Algo^1_{opt}$ returns the true topology for 16 of the instances and near-perfect prediction for all instances with 20% or less background noise. All results for

Table 2.8: Each or the three variations of the full algorithm ranked from one to three based on the deviation γ for each of the 60 synthetic instances with background noise. A MST initial solution yields the best results on average.

		Instances														
Construct	α	A1:4	B1:4	C1:4	D1:4	E1:4	F1:4	G1:4	H1:4	J1:4	K1:4	L1:4	M1:4	N1:4	P1:4	Q1:4	Avg
MST	0.1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 2	1 2 2 2	1 1 1 1	1 1 2 1	1 1 1 1	2 1 1 1	1 1 1 3	1 1 1 1	1 2 1 1	1.15
GSPT	0.1	1 1 2 1	1 1 1 1	1 1 1 2	1 1 1 1	1 1 2 2	1 1 1 3	1 1 2 1	1 1 1 1	1 1 2 3	1 1 1 3	1 1 1 3	1 1 2 2	1 1 2 1	1 1 2 2	2 1 2 3	1.38
CSPT	0.3	3 3 3 3	3 3 3 3	3 3 3 3	3 3 3 3	3 3 3 3	3 3 3 2	3 3 3 3	3 3 3 3	3 3 3 2	3 3 3 2	3 3 3 2	3 3 3 3	3 3 3 2	3 3 3 3	3 3 3 2	2.90

Table 2.9: The deviation γ for each of the 60 synthetic instances with background noise using the three variation of the full algorithm. A MST construction yields both the best average rank (Table 2.8) as well as the best average deviation $\overline{\gamma}$. As the GSPT with $\alpha = 0.1$ closely resembles a MST, this constructer yields similar results as MST.

		Instances														
Construct	α	A1:4	B1:4	C1:4	D1:4	E1:4	F1:4	G1:4	H1:4	J1:4	K1:4	L1:4	M1:4	N1:4	P1:4	Q1:4	Avg
MST	0.1	0 0	0 0		0 0		0 0	0	0		0	0	0		0	0	0.20
GSPT	0.1	0 0	0 0		0 0		0 0	0	0		0	0	0 0		0		0.22
CSPT	0.3																0.57

< 0.1
< 0.2
< 0.35
< 0.5
< 0.7
< 1.0
≤ 2.0

instances with 30% background noise are also decent with all deviation values γ being under 0.35. For 40% background noise the predictions are no longer of useful quality.

For only two of the 60 instances $Algo^1_{opt}$ produces a positive solution gap, namely 0.24% for instance H2 and 0.04% for instance Q3. In instance H2, 4/26 internal edges are incorrectly predicted and in instance Q3, 6/49 internal edges and 2/500 leaf edges are incorrectly predicted. For all remaining instances the solution gap of $Algo^1_{opt}$ is in the range $[-1\%, 0\%]$.

For the instances with background noise it is clear that the prediction accuracy of the algorithm is dependent on the starting solution. The variation of the full algorithm which uses CSPT as an initial solution results in poor predictions for the majority of the instance. Although all three construction methods yield a similar average deviation $\overline{\gamma}$ individually (see Table 2.7), both GSPT and MST return initial solutions with high accuracy with respect to leaf edge prediction (recall GSPT favors the cable distance given $\beta = 0.9$) as compared to CSPT. This further supports the hypothesis that the neighborhood search is more proficient at correcting internal edges rather than leaf edges.

2.6.2.3 Convergence study

Both $Algo^0_{opt}$ and $Algo^1_{opt}$ converges well within the upper limit of 200,000 iterations, with an average of 97% of all accepted moves occurring within the first

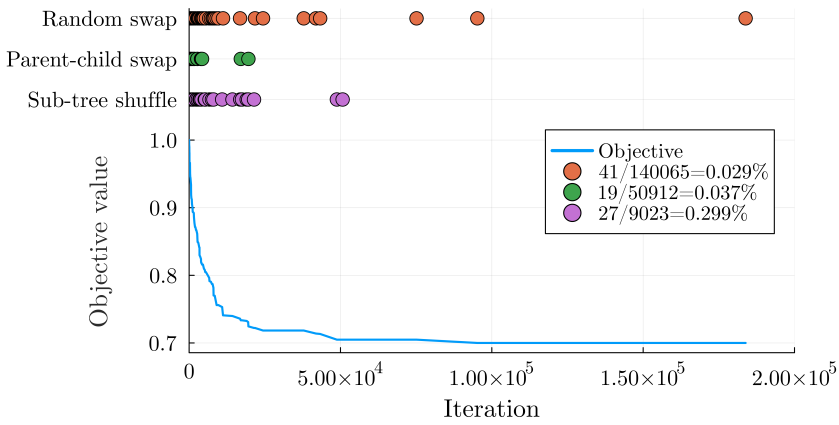


Figure 2.6: A convergence study shown for $Algo^0_{opt}$ when run on instance N0. Above the plot illustrating the evolution of the objective value, each neighborhood function is plotted at the iteration when a move was accepted by the algorithm and thus led to an improvement in the objective value. The format of the legend is *accepted/selected*, i.e. the fraction of successful moves for each neighborhood function.

50,000 iterations. In all iterations *Random swap* is selected with a probability of 70%, *Parent-child swap* with a probability of 25%, and *Sub-tree shuffle* with a probability of 5%. All neighborhood functions play a significant role in all instances considered. Figure 2.6 illustrates a convergence study for one of the instances.

In general, we observe that there is a positive linear correlation between the number of accepted neighborhood moves and the size of the network. Consequently, smaller networks tend to converge faster than larger ones.

When considering the same network topology, the addition of more background noise generally reduces the number of accepted neighborhood moves. However, we do not find any clear correlation between the speed of convergence and the background noise level of an instance.

2.6.3 Real-life instance

Lastly, we examine if the effectiveness of the proposed methodology translates to the real-life instance R0-R4 (see Figure 2.1). This is done using the best version of the full algorithm, $Algo_{opt}^{0/1}$, the variant where an MST solution is the starting point of the heuristic search. Since the distance and parsimony score of a candidate solution are normalized individually in the objective, it should not matter that the distance scale of the real life instance is different to the synthetic instances. However, as the real life instance is given in geographic coordinates which have been converted and scaled to Cartesian coordinates, the scaling factor α in the objective (2.6) is still tuned for good measure. The tuning results are shown in Suppl. S5 Table S-9.

The optimal value of α is found to be 0.1 for both the instance without background noise (R0) and the instances with background noise (R1-R4). Earlier, an α-value of 0.5 and 0.1, respectively, were found optimal for the synthetic instances. A lower fraction of customers are connected to their closest splitter in the real-life instance compared to the synthetic instances. This could explain

		Instances					
Construct	α	R0	R1	R2	R3	R4	Avg
MST	0.1	0.045	0.091	0.188	0.383	0.788	0.30

< 0.1
< 0.2
< 0.35
< 0.5
< 0.7
< 1.0
≤ 2.0

Table 2.10: The deviation γ for each of the five variations of the real-life instance using $Algo_{opt}^{0/1}$ with an optimal α-value of 0.1. The average deviation $\overline{\gamma}$ is slightly higher for the real-life instances as compared to the synthetic instances, but overall a similar trend is observed with respect to the background noise level.

why it is better to assign less weight to the distance contribution in the objective for the real-life instance as compared to the synthetic instances in the no background noise case. As a perfect instance with no background noise is a very special case, we accept that the tuning of such an instance seems more sensitive to the underlying topology.

Table 2.10 shows the results of running $Algo_{opt}^{0/1}$ with $\alpha = 0.1$ on all five instances of the real topology R. Good solutions are obtained for instance R0-R2 with one, two, and six out of a total of 333 edges being wrongly predicted, respectively. The solution quality decreases significantly when introducing 30-40% noise (instance R3 and R4) where 14 and 48 edges are wrongly predicted, respectively. The observed level of correctness closely resembles the results using $Algo_{opt}^{0/1}$ on the synthetic instances as seen in Table 2.5 and 2.9.

2.7 Conclusion

We have presented an algorithm for reconstructing the topology of a telecommunication network, using time series data as well as physical distances for guiding the search. The results are very promising since we are able to reconstruct the network with a very high precision even for instances having up to 30% background noise.

The algorithm may be used in several telecommunication applications, where documentation of the topological structure has been lost, or changes are sparsely updated. Indeed the results are so promising that it could be more efficient to just calculate a tree based on the most recent data, than to maintain topological information, given the assumption that it is possible to filter out background noise levels to less than 30% for real transmission data.

One weakness of the proposed method, is the ability to handle *empty splitter nodes*, i.e. splitters that have no direct customer descendant, only other splitter nodes. *Empty splitters* are not common in telecommunication networks and were not present in any of the topologies presented in this paper. Figure 2.7 illustrates an example of why an *empty splitter* can cause prediction errors in the context of our proposed solution methodology.

Tests have been performed on synthetic data resembling how signals are transmitted in a data network. A next step will therefore be to try the approach on real-life data. For this purpose it will likely be necessary to develop filtering methods that are able to translate signal changes into discrete events. Furthermore, each modem is measuring several parameters associated with the data transmission. It would be interesting to study which of these parameters contain the most useful information for reconstructing the topology.

Although the developed tools were designed for telecommunication networks,

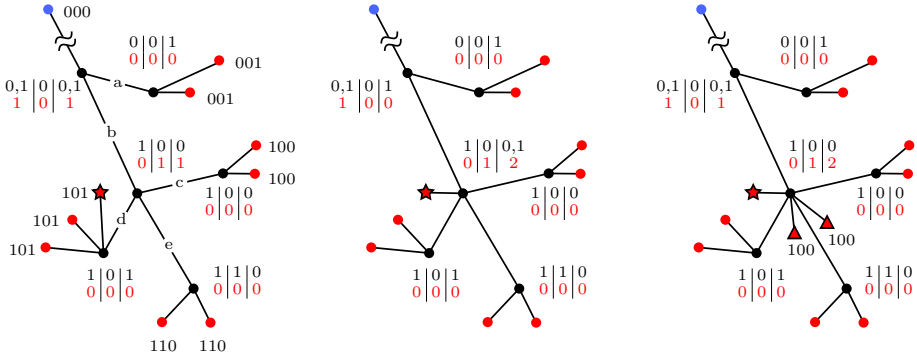


Figure 2.7: An example of why splitter nodes (black) without customer (red nodes) children (hereafter *empty splitters*), pose a potential issue to the proposed algorithm. (A) illustrates the true topology and modem data of the example. The example has three data points and no background noise. Initially at CMC level (blue node), the data is 000. A disturbance occurs on edge b at time $t = 1$, on edge e at $t = 2$, and on both edge d and a at $t = 3$, resulting in the modem data pictured. Using Hartigan's algorithm (2.11)-(2.13), the total Parsimony score is four. Here, the black values represent the optimal state(s) in each node, at each time, and the red values are the corresponding costs. In (B), the star-shaped customer is moved to the *empty splitter*. This improves the objective value (2.6) as this customer is closer in distance to the *empty splitter*, and since the total Parsimony score remains the same by shifting costs from above to below edge b. In (C) we show that solution (B) would worsen the Parsimony score to five, had the *empty splitter* been non-empty. This is shown by introducing two new customers (triangular-shaped). When every splitter has two or more customer children, the modem data from these will often be enough to establish a majority for the true state in the parent splitter, depending on the background noise level. This holds under the assumption that any splitter has at most three splitter children.

they may also have applications in biology or archaeology, where the physical location or time-period of some observations is known. Adding the differences in physical locations or the differences in time periods as a second criterion in the optimization may lead to more correct phylogenetic trees. Furthermore, the physical location or point in time of intermediate nodes may bring a deeper understanding of the evolutionary process.

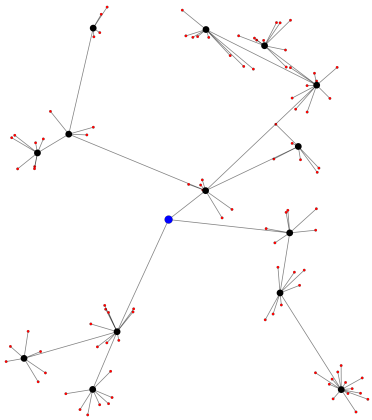
Acknowledgments

This project was supported by Innovation Fund Denmark, project number 0224-00055B, GREENFORCE.

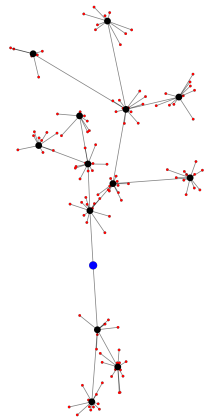
Appendices

S1 Synthetic instance topologies

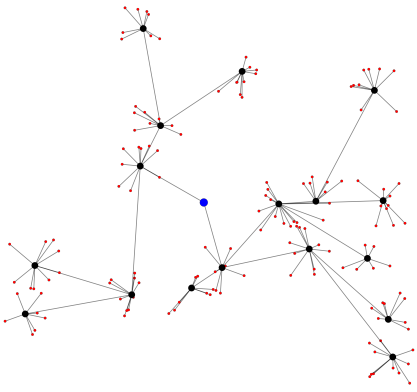
The 15 synthetic instance topologies introduced in Table 2.1 are pictured below.



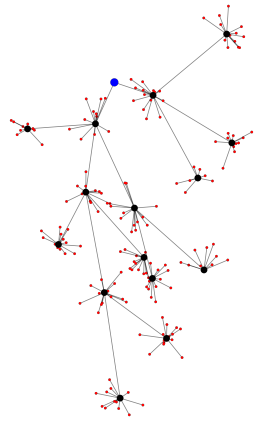
Instance A0-A4



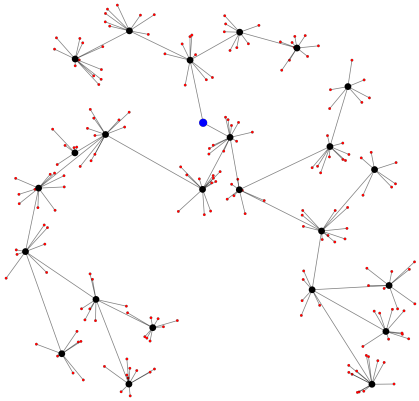
Instance B0-B4



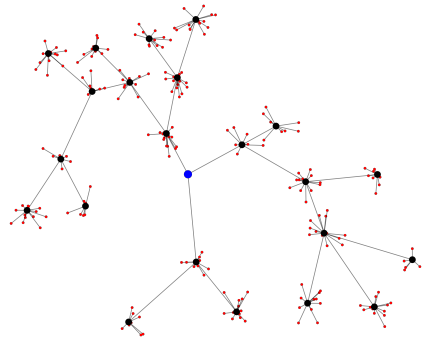
Instance C0-C4



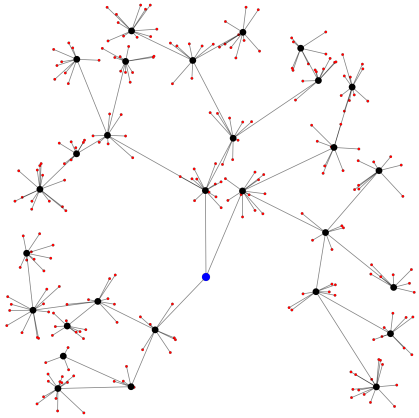
Instance D0-D4



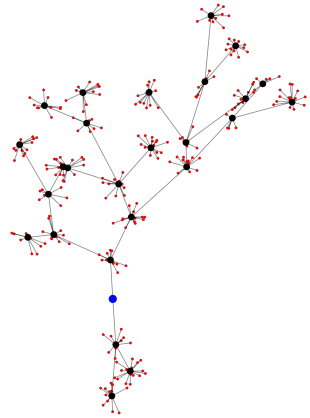
Instance E0-E4



Instance F0-F4



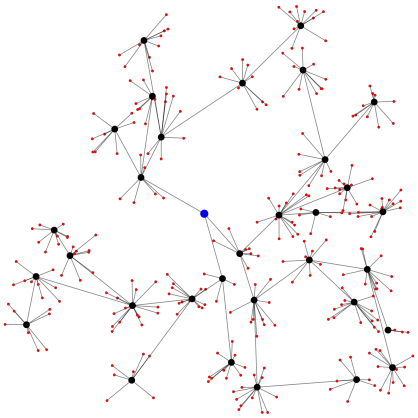
Instance G0-G4



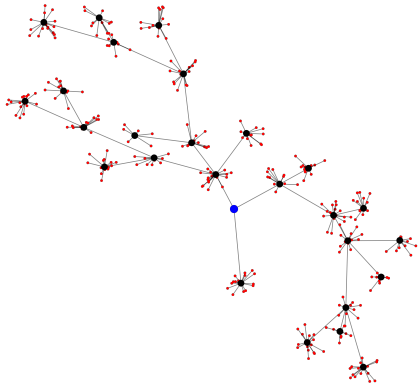
Instance H0-H4

S2 Real instance topology

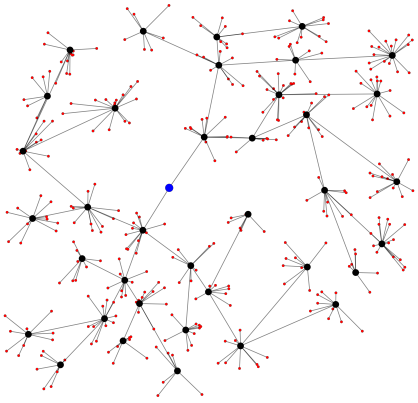
The real instance topology introduced in Table 2.1 is pictured below.



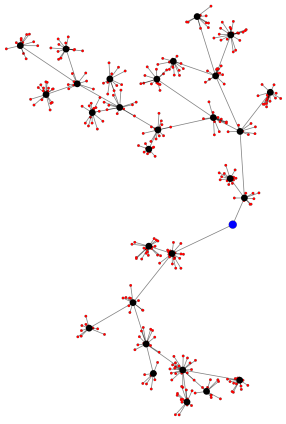
Instance J0-J4



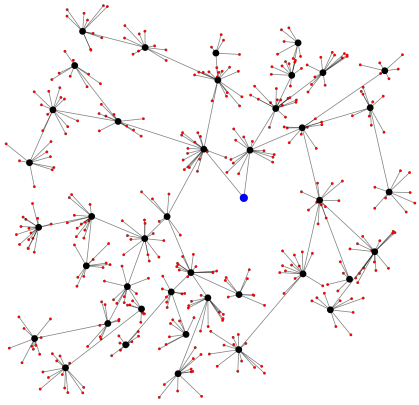
Instance K0-K4



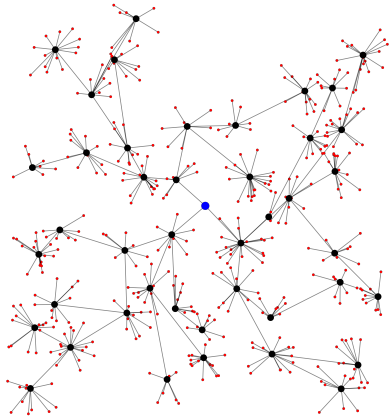
Instance L0-L4



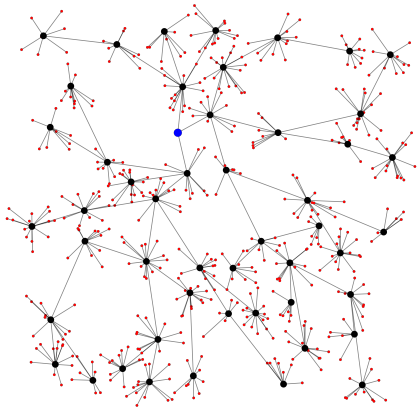
Instance M0-M4



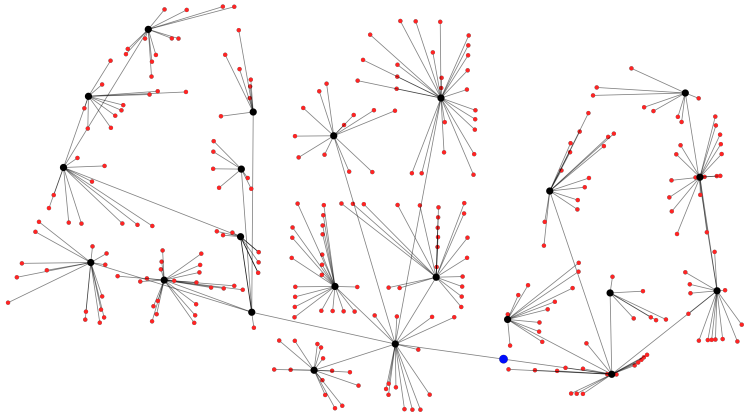
Instance N0-N4



Instance P0-P4



Instance Q0-Q4



Instance R0-R4

S3 Construction heuristics β-tuning for the synthetic instances

β	Instances (with GSPT)																																																																										Avg[1]	Avg[2]
0	11	11	11	11	11	11	11	11	11	11	2	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	10.40	11.00
0.1	1	10	10	9	10	1*	10	9	10	10	11	10	10	10	10	1*	8	10	9	10	6	10	9	9	10	1*	10	10	10	10	2	10	10	10	10	6	10	10	8	10	5	10	10	10	10	1*	10	10	10	10	4	10	10	10	10	1	9	8	9	10	1	10	10	10	10	10	10	10	10	10	3	10	10	10	10	3.60	9.78
0.2	2	9	9	10	9	1*	7	10	9	8	10	9	9	9	9	1*	9	9	7	9	1	9	10	10	9	1*	9	9	8	9	1	9	9	9	9	6	7	9	10	9	1	9	9	9	9	1*	9	9	9	9	4	9	9	9	9	1	10	10	10	9	2	9	9	9	9	4	9	9	9	9	1	9	9	9	9	**2.47**	9.00
0.3	2	1	8	8	7	1*	7	8	8	6	5	8	8	8	7	1*	9	8	10	8	2	8	8	8	8	1*	8	8	9	8	2	8	8	8	8	6	9	8	7	7	4	8	8	8	8	1*	7	8	8	8	2	8	8	8	8	1	8	9	7	7	3	8	8	8	8	6	8	8	8	7	2	8	8	8	8	2.60	7.80
0.4	4	8	3	7	6	1*	5	6	7	9	9	7	7	7	8	1*	7	7	8	5	3	7	7	7	6	1*	7	7	7	7	2	7	7	7	6	1	8	7	9	8	3	7	7	7	7	1*	8	7	7	7	1	7	7	7	6	4	7	7	7	8	4	7	7	7	7	1	7	7	6	8	4	7	7	7	7	2.67	6.97
0.5	4	5	1	6	8	1*	2	7	5	7	1	6	2	3	6	1*	6	6	6	6	4	6	6	6	7	1*	6	6	6	5	5	6	6	6	7	1	6	6	6	6	2	5	4	6	6	1*	6	5	6	6	3	6	6	6	5	4	6	6	6	5	5	6	6	6	6	2	6	6	7	6	5	6	6	6	6	2.67	5.70
0.6	4	2	5	5	5	1*	9	4	6	5	3	3	1	1	5	1*	3	5	5	7	5	5	2	5	5	1*	4	5	5	6	6	5	5	5	5	1	5	5	5	5	6	6	6	4	5	1*	4	6	5	3	4	2	5	5	7	4	5	5	5	6	6	5	3	2	5	2	5	5	5	5	6	5	5	5	5	3.40	4.70
0.7	7	6	7	4	4	1*	6	1	4	4	3	1	3	6	1	1*	4	1	2	4	7	1	5	4	4	7	3	3	3	3	6	2	4	3	4	4	1	4	4	4	7	1	5	1	4	7	2	4	4	5	7	3	4	4	4	7	4	4	4	4	7	3	2	4	4	7	4	4	4	4	7	4	4	4	4	5.67	3.53
0.8	8	6	6	3	3	8	3	3	2	2	6	2	4	5	2	8	1	3	1	3	8	4	4	3	3	8	1	2	2	4	8	1	1	4	3	5	3	2	3	3	8	4	2	4	2	8	1	1	1	4	8	1	1	3	3	8	1	3	3	3	8	1	1	1	2	9	3	3	3	3	8	1	2	3	3	7.73	2.58
0.9	8	3	4	2	2	9	1	2	1	1	7	5	6	4	4	9	2	2	3	2	8	3	3	2	1	9	2	1	1	1	9	3	3	1	2	9	2	1	1	2	9	2	3	3	3	9	3	2	2	2	9	4	2	2	2	10	2	1	1	2	9	2	5	5	3	8	2	2	2	2	9	2	1	2	2	8.73	**2.32**
1	8	3	2	1	1	10	4	5	3	3	7	4	5	2	3	10	5	4	4	1	10	2	1	1	2	10	5	4	4	2	10	4	2	2	1	10	4	3	2	1	10	3	1	2	1	10	5	3	3	1	10	5	3	1	1	9	3	2	2	1	10	4	4	3	1	5	1	1	1	1	10	3	3	1	1	9.27	2.52

β	Instances (with CSPT)																																																																										Avg[1]	Avg[2]
0	11	11	11	11	11	8	10	10	9	10	4	6	11	11	11	7	9	8	9	11	8	11	11	11	11	8	11	11	11	11	5	10	10	6	11	5	10	11	11	11	2	7	11	11	11	1	10	11	11	11	6	10	11	11	11	3	11	11	11	11	5	11	11	11	11	1	8	11	11	11	1	9	8	11	11	5.00	10.35
0.1	1	2	9	9	10	1*	1*	8	7	11	1	1	4	10	10	10	5	2	9	7	2	1	2	5	10	9	2	8	9	10	3	1	2	1	10	2	7	9	8	10	6	6	2	6	10	9	1	2	10	10	1	6	3	10	10	9	7	5	10	10	1	3	3	4	10	4	5	3	8	10	6	6	2	2	9	4.33	6.22
0.2	2	2	2	10	4	1*	1*	1	2	8	3	3	1	4	8	1*	1	7	7	8	1	2	1	3	9	5	9	8	10	7	1	1	1	7	1	2	5	7	3	7	4	1	3	4	5	4	6	2	4	8	2	2	4	7	8	9	1	1	8	8	4	2	1	9	7	2	1	4	6	8	3	4	3	1	10	2.93	4.63
0.3	3	1	10	8	8	1*	3	1	3	7	2	2	2	3	1	1*	7	1*	8	4	4	3	6	6	1	5	6	1	8	9	2	3	8	2	8	2	6	4	3	6	1	3	1	1	6	4	5	8	5	9	3	1	1	3	6	5	4	2	5	9	3	1	2	10	9	3	3	1	5	9	2	2	1	4	2	**2.73**	4.45
0.4	5	4	4	3	6	1*	3	1	1	9	6	5	3	1	9	1*	6	2	3	3	7	5	4	1	3	1	5	2	6	8	4	4	4	5	3	6	3	3	2	4	4	2	4	2	8	4	2	4	1	6	4	3	5	1	9	11	6	2	3	7	2	4	4	1	8	5	2	2	3	7	4	3	5	3	8	4.33	**4.00**
0.5	4	4	1	6	6	1*	3	1	5	6	5	4	5	6	5	1*	1	2	1	10	3	9	8	7	8	1	1	2	2	6	6	9	6	3	7	1	8	2	1	8	3	4	5	10	4	7	2	6	9	7	5	4	2	2	2	1	2	7	7	6	6	5	6	5	3	6	4	5	1	6	5	1	6	8	5	3.67	4.78
0.6	5	4	3	7	9	1*	3	1	4	2	7	7	6	2	6	1*	1	2	2	9	5	6	7	7	6	1	3	8	7	4	9	6	5	11	9	9	1	1	5	9	7	5	8	5	9	2	7	7	6	5	7	5	6	9	5	1	2	6	6	5	7	6	5	3	1	7	6	7	3	4	7	5	4	7	1	5.07	5.18
0.7	7	8	8	5	5	9	3	1	11	1	8	7	6	9	6	1*	1	2	6	6	6	8	10	9	7	4	7	5	5	1	7	5	9	8	5	7	4	5	7	1	8	8	7	3	7	3	4	1	2	4	8	7	7	8	7	7	5	4	1	4	9	7	7	7	5	8	7	8	9	5	8	7	7	6	3	6.67	5.63
0.8	8	7	7	1	3	7	8	7	6	5	9	9	8	7	3	8	8	9	4	2	9	7	5	9	5	5	4	6	1	5	8	7	3	4	4	8	2	6	10	5	9	9	6	7	2	8	8	5	8	3	9	8	8	6	3	7	8	8	9	3	8	8	8	6	6	9	9	10	10	3	9	8	9	9	6	8.07	6.17
0.9	8	9	5	4	2	10	9	9	8	3	10	10	9	5	4	9	10	10	5	1	10	4	3	2	4	10	8	4	3	3	10	8	7	9	2	10	9	8	6	2	10	10	9	9	3	10	9	9	3	1	10	9	9	4	4	6	10	10	4	2	10	9	9	2	4	10	10	6	2	1	10	10	10	5	4	9.53	6.03
1	10	9	5	1	1	11	11	11	10	4	11	11	10	8	2	11	11	11	11	5	11	10	9	4	2	11	10	7	4	2	11	11	11	10	6	11	11	10	9	3	11	11	10	8	1	11	11	10	7	2	11	11	10	5	1	4	9	9	2	1	11	10	10	8	2	11	11	9	7	2	11	11	11	10	7	10.47	7.43

*Solution is optimal, $\gamma = 0$

Table S-9: Ranking of deviation γ for GSPT (top) and CSPT (bottom) solutions for all 75 synthetic instances with 11 variations of β. For each instance (column), the β value which yields the best deviation score is ranked 1 and the worst is ranked 11. In each column, the three best and three worst rankings are colored green and red, respectively, and the remainder yellow. Avg[1] is the average rank of all 15 instances without background noise and Avg[2] is the average rank of all 60 instances with background noise.

S4 Complete heuristics α-tuning for the synthetic instances

α	Instances (with MST)	Avg^1	Avg^2
0	1* 1* 1* 2 2 9 9 4 3 11 1 1 1 1 11 7 3 2 2 1 8 4 2 1 11 1* 1* 3 1 11 1* 1* 1* 1 1 6 2 1 2 11 7 2 2 2 11 1* 1* 1* 1 11 4 2 2 2 11 2 3 2 2 11 8 3 2 1 3 7 1* 1* 1 11 7 2 1 2 11	4.67	3.57
0.1	1* 1* 1* 1 1 7 1* 1* 1 1 1 1 2 2 1 5 1* 1* 1 2 3 1 1 2 1 1* 1* 1* 2 1 1* 1* 2 2 11 1* 1* 2 1 7 1 1 1 1 1 1* 1* 2 2 1 4 1* 1 1 1 1 1 1* 1 1 1 1 2 1 4 1* 2 2 1 6 1* 2 1 1	2.53	**1.50**
0.2	1* 1* 3 3 3 1* 1* 2 1 2 1 3 3 3 6 1* 1* 3 3 2 3 1 3 3 10 1* 1* 2 3 2 1* 3 3 3 10 1* 3 3 3 7 1 3 3 3 2 1* 3 3 3 2 4 3 3 3 2 3 2 3 3 2 5 2 3 3 4 1 3 3 3 2 1* 3 3 3 2	1.73	2.95
0.3	1* 4 4 4 4 1* 1* 3 3 2 1 4 4 4 10 1* 3 4 4 6 3 3 4 10 8 1* 4 4 4 4 2 1* 4 4 6 7 1* 4 4 4 3 1 4 4 4 3 1* 4 4 4 4 1* 4 4 4 3 4 4 4 4 5 5 4 4 4 2 4 4 4 4 5 1* 4 4 4 3	1.80	4.13
0.4	1* 5 5 4 4 1* 4 4 5 2 1 5 5 4 9 1* 5 4 4 6 3 5 5 10 8 1* 5 5 4 4 1* 5 4 4 7 1* 5 5 4 3 1 5 5 5 3 1* 5 5 5 4 1* 5 5 5 4 4 5 4 5 5 5 5 5 5 5 1 5 5 5 5 1* 5 5 5 4	1.60	4.87
0.5	1* 5 6 6 4 1* 4 6 5 2 1 6 6 4 6 1* 5 6 6 6 1* 6 5 4 2 1* 6 6 6 4 1* 6 6 4 7 1* 6 6 6 6 6 6 6 6 10 1* 6 5 5 4 1* 6 6 6 5 4 5 6 6 3 1 5 6 6 11 1 6 6 6 5 1* 6 6 5 9	**1.53**	5.62
0.6	7 7 6 6 4 1* 6 6 7 8 7 7 7 4 6 6 7 7 7 6 1* 7 7 4 2 1* 7 7 6 4 1* 7 6 7 5 7 7 6 6 3 5 7 7 6 3 1* 7 7 7 7 4 7 7 7 7 10 7 7 7 6 3 3 7 7 7 6 6 7 7 7 5 5 7 7 7 9	4.33	6.28
0.7	8 8 8 6 4 1* 7 6 7 8 8 7 7 4 2 8 8 8 8 6 7 7 8 4 2 1* 8 7 6 4 8 8 6 7 5 8 8 6 8 1 8 8 8 8 3 8 7 7 7 4 8 8 8 7 6 7 8 8 6 5 3 8 8 8 6 8 8 7 10 8 8 8 7 8	6.60	6.68
0.8	8 9 8 6 4 8 7 9 9 8 9 9 7 4 2 8 9 9 8 6 9 7 8 4 2 9 9 7 6 4 9 8 9 7 2 9 9 9 8 1 9 9 8 8 3 9 9 9 7 3 9 9 8 7 6 7 9 9 9 10 9 9 9 8 6 9 9 9 7 5 9 9 9 9 7	8.67	7.22
0.9	10 9 8 6 4 10 10 10 9 6 10 9 7 4 2 10 10 10 8 4 10 10 8 4 2 10 9 7 6 9 10 10 9 7 2 10 10 10 8 9 10 10 10 10 8 3 10 10 10 10 4 10 10 8 7 6 10 10 10 9 5 10 10 10 8 6 10 10 9 11 4 10 10 10 9 6	10.00	7.82
1	11 9 8 6 4 11 11 11 9 6 11 9 7 4 2 11 11 11 10 8 4 11 10 8 4 2 11 11 11 11 9 11 10 9 7 2 11 11 11 11 10 11 11 11 11 3 11 11 10 10 4 11 10 8 7 6 10 11 10 9 5 11 11 10 8 6 11 11 11 10 3 11 11 10 9 5	10.93	8.30

α	Instances (with GSPT)	Avg^1	Avg^2
0	1* 3 3 2 1 10 6 4 3 11 11 1 2 1 11 7 3 2 2 3 8 3 2 3 11 1* 1* 1* 1 11 1* 4 2 2 11 9 2 2 1 11 6 3 2 1 11 1* 1* 1* 1 5 10 2 1 1 11 6 3 2 1 11 6 3 2 2 5 9 1* 1* 1 11 6 2 1 1 3	6.13	3.60
0.1	1* 1* 1* 1 2 7 1* 1* 1 1 1 1 1 2 2 5 1* 1* 1 2 3 1 1 1 2 10 1* 1* 1* 2 3 1* 1* 1 1 10 7 1* 1 2 1 2 1 1 2 1 1* 1* 2 2 1 6 1* 2 2 10 6 1* 1* 2 2 1 1 1 1 1 1 6 1* 2 2 3 1 1 2 2 2	3.27	**1.83**
0.2	1* 1* 2 3 11 1* 1* 2 1 2 2 3 3 3 1 1* 1* 3 3 1 3 1 3 1 9 1* 1* 3 3 1 1* 2 3 3 5 7 3 3 3 6 2 2 3 3 2 1* 3 3 3 2 6 2 3 3 9 6 2 3 3 1 1 2 3 3 3 3 3 3 3 2 1 3 3 3 5	2.47	2.90
0.3	1* 3 3 3 3 1* 1* 3 3 2 2 4 4 4 8 1* 3 4 4 4 3 4 4 4 7 1* 4 4 4 1 1* 3 4 7 8 1* 4 4 4 6 2 4 4 4 6 1* 4 4 3 3 3 4 4 5 1 1* 4 4 4 9 1 4 4 4 4 4 3 4 4 4 1 1 4 4 4 4 4	**1.53**	4.00
0.4	1* 5 5 3 3 1* 4 4 5 2 2 5 5 4 8 1* 5 4 5 4 3 5 11 11 7 1* 5 5 4 4 1* 5 4 8 7 1* 5 5 4 10 2 5 5 5 3 1* 5 5 5 3 3 5 5 4 2 1* 5 4 5 9 1 5 5 4 2 3 5 4 5 4 1 5 5 5 1	**1.53**	4.93
0.5	1* 5 5 3 3 1* 4 6 5 2 2 6 6 4 8 1* 5 6 6 4 1 6 4 5 1 1* 6 6 6 4 1* 6 6 8 4 1* 6 6 6 4 7 6 6 6 10 1* 6 5 5 6 3 6 6 6 8 1* 5 6 6 3 1 5 6 6 11 2 6 6 5 6 1 6 6 6 8	1.67	5.58
0.6	7 7 5 7 3 1* 6 6 7 8 2 7 6 4 7 6 7 7 6 9 1 6 6 5 1 1* 7 7 6 4 1* 7 6 10 1 4 7 6 6 4 7 7 7 6 4 1* 7 7 5 6 6 7 6 7 3 1* 7 7 6 3 6 7 7 7 10 1 7 8 9 5 7 7 7 7 6	3.47	6.22
0.7	8 8 8 7 3 1* 8 6 7 8 8 8 6 4 3 8 7 8 6 9 7 6 7 5 1 1* 8 7 6 4 8 8 6 4 1 5 8 8 8 2 1 8 8 8 4 8 7 7 8 6 1 8 8 7 3 1* 8 8 6 7 6 8 8 7 9 7 8 10 7 10 8 8 8 7 7	5.20	6.72
0.8	9 9 8 7 3 8 8 9 9 8 10 8 6 4 3 8 9 9 9 9 10 6 7 5 1 9 9 7 6 4 9 8 9 4 1 6 9 9 8 2 9 9 8 8 6 9 9 7 8 6 2 9 8 7 3 9 9 9 9 7 9 9 9 9 6 8 9 11 7 9 9 9 9 9 11	8.27	7.37
0.9	10 9 8 7 3 9 10 10 9 6 7 8 6 4 3 10 10 10 9 4 9 10 7 5 1 10 9 7 6 9 10 10 9 4 6 10 10 10 8 8 10 10 10 8 6 10 9 10 8 10 9 10 8 7 3 10 10 10 9 3 10 10 10 9 6 10 10 7 10 8 10 10 10 10 10	9.60	7.93
1	11 9 8 7 3 11 11 11 9 6 9 8 6 4 3 11 11 10 9 4 11 10 7 5 1 11 11 11 11 10 11 11 11 11 9 11 11 11 11 9 11 11 11 11 6 11 11 10 8 10 11 10 8 7 3 11 11 10 9 3 11 11 10 9 6 11 11 9 11 7 11 11 10 10 8	10.87	8.68

α	Instances (with CSPT)	Avg^1	Avg^2
0	1 1 1 11 11 3 11 11 1 2 11 1 11 11 2 1 2 1 5 1 9 7 8 9 11 7 1 9 1 2 2 6 2 11 11 2 11 1 11 1 9 3 8 8 10 1 10 1 4 11 11 2 4 11 11 1 1 1 1 1 3 4 6 11 11 5 11 3 1 11 1 1 9 5 11	4.47	5.93
0.1	1 1 2 1 1 3 7 1 1 3 1 2 2 2 2 1 1 1 1 2 3 8 8 9 2 7 1 9 1 3 2 6 1 1 9 2 2 1 2 2 5 1 8 8 10 1 9 2 4 10 5 1* 4 1 7 1 2 2 1 2 3 2 6 1 9 1 8 1 2 1 1 2 2 4 7	2.47	3.53
0.2	1 1 2 1 1 3 7 1 3 3 1 2 2 1 2 1 2 3 1 2 3 8 8 1 1 7 1 1 1 3 2 8 6 10 9 2 1 1 2 2 2 2 1 8 6 1 1 2 4 1 5 3 1 1 8 1 2 2 3 2 3 1 1 2 10 1 9 2 11 1 1 5 1 1 7	2.27	3.25
0.3	1 1 2 1 1 6 7 1 3 3 1 4 1 2 2 1 2 3 1 3 8 8 2 2 7 1 1 1 1 2 8 6 2 8 2 2 1 1 2 2 3 3 1 4 1 2 2 4 2 5 4 4 1 3 1 5 2 3 2 3 4 2 3 5 1 10 8 3 1 1 5 2 6 1	2.47	**3.08**
0.4	1 1 2 1 1 6 7 1 3 1 1 4 4 2 2 1 2 3 1 2 3 8 6 8 2 7 1 1 1 1 3 2 1 6 2 1 2 2 5 2 2 2 2 3 11 8 1 1 2 2 4 2 5 4 2 1 1 1 4 2 3 2 1 4 3 4 6 8 2 8 3 1 1 5 6 6 4	2.80	3.20
0.5	1 1 2 1 1 6 1 1 3 3 1 4 4 2 1 1 2 3 10 2 1* 1 6 3 2 1 1 1 1 3 2 2 6 2 5 2 2 5 2 2 6 6 2 2 5 1 11 2 4 2 5 4 3 1 2 1 5 2 3 2 7 4 4 6 7 8 2 8 3 1 1 5 7 6 3	2.93	3.28
0.6	1 1 2 1 1 6 1 1 3 3 1 4 4 2 7 1 2 3 6 2 1* 5 1 4 2 1 1 1 1 3 2 3 6 2 4 1 2 5 2 2 1 6 4 2 1 1 2 11 4 6 1 4 4 1 6 1 5 2 3 2 2 3 5 6 8 8 1 4 3 1 1 5 2 6 2	**1.93**	3.27
0.7	1 1 2 1 1 6 1 1 3 3 1 4 4 2 7 1 2 3 6 2 9 1 2 4 2 1 1 1 1 3 2 4 6 2 3 2 2 5 2 2 7 6 5 4 1 1 2 2 4 6 2 4 4 1 4 1 5 2 3 2 7 4 6 6 1 4 2 5 3 1 1 5 8 6 5	3.07	3.18
0.8	1 1 2 1 1 6 1 1 3 3 1 4 4 2 7 1 2 3 6 2 9 1 2 4 2 1 1 1 1 3 2 4 3 2 2 2 2 5 2 2 8 6 5 4 7 1 2 2 1 5 10 4 4 8 4 1 5 2 3 2 7 9 11 6 1 6 2 6 3 1 1 3 10 6 6	3.80	3.47
0.9	1 1 2 1 1 1 1 1 3 3 1 4 4 2 7 1 10 3 6 2 7 4 2 4 2 1 1 1 1 3 1 8 3 2 6 2 2 5 2 2 10 6 7 4 7 1 2 2 1 6 3 4 4 8 8 11 5 2 3 2 7 10 9 6 1 7 2 7 3 1 10 4 2 2 7	4.27	3.73
1	1 1 2 1 1 1 1 1 3 3 1 4 4 2 7 11 11 0 3 11 2 8 6 2 11 2 1 1 1 11 1 3 2 8 3 2 6 2 2 5 2 2 11 6 10 7 7 1 2 2 1 6 4 4 4 8 8 10 5 2 3 2 11 11 9 5 1 11 7 11 3 1 11 11 11 2 7	5.73	4.65

*Solution is optimal, $\gamma = 0$

Table S-9: Ranking of deviation γ for MST (top), GSPT (middle), and CSPT (bottom) solutions for all 75 synthetic instances with 11 variations of α. For each instance (column), the α value which yields the best deviation score is ranked 1 and the worst is ranked 11. In each column, the three best and three worst rankings are colored green and red, respectively, and the remainder yellow. Avg^1 is the average rank of all 15 instances without background noise and Avg^2 is the average rank of all 60 instances with background noise.

S5 Complete heuristics α-tuning for the real instance

α	Instances R0	R1	R2	R3	R4	Avg^1	Avg^2
0	4	1	2	2	1	4	**1.5**
0.1	1	2	1	1	2	**1**	**1.5**
0.2	2	3	3	3	3	2	3
0.3	2	4	4	4	4	2	4
0.4	5	5	5	5	5	5	5
0.5	6	6	6	6	5	6	5.75
0.6	7	7	7	7	5	7	6.5
0.7	8	8	8	8	8	8	8
0.8	9	9	9	9	8	9	8.75
0.9	10	10	10	10	8	10	9.5
1	11	11	11	11	11	11	11

*Solution is optimal, $w = 0$

Table S-9: Ranking of deviation γ for the five background noise variants of the real instance R, with 11 variations of α in 2.6. For each instance (column), the α value which yields the best deviation score is ranked 1 and the worst is ranked 11. In each column, the three best and three worst rankings are colored green and red, respectively, and the remainder yellow. Avg^1 is the average rank of the single real instance (R0) without background noise and Avg^2 is the average rank of all four real instances with background noise (R1-R4).

Chapter 3

Paper II

Topology Reconstruction in Telecommunication Networks: Embedding Operations Research within Deep Learning

---

Status: Paper is submitted and under peer review for publication in *Computers and Operations Research*

Authors: Tobias Engelhardt Rasmussen[a] · Siv Sørensen[b]
David Pisinger[b] · Thomas Martini Jørgensen[a] · Andreas Baum[a]
[a] Technical University of Denmark, DTU Compute, Kgs. Lyngby, Denmark
[b] Technical University of Denmark, DTU Management, Kgs. Lyngby, Denmark

Keywords: Dynamic programming, Maximum parsimony, Network reconstruction, Phylogenetic trees, Telecommunication networks, Variable neighborhood search

Abstract

We consider the task of reconstructing the cabling arrangements of *last-mile* telecommunication networks using customer modem data. In such networks, downstream data traverses from a source node down through the branches of the tree network to a set of customer leaf nodes. Each modem monitors the quality of received data using a series of continuous data metrics. The state of the data, when it reaches a modem, is contingent upon the path it traverses through the network and can be affected by, e.g., corroded cable connectors.

We train an encoder to identify irregular inherited *events* in modem quality data, such as network faults, and encode them as discrete data sequences for each modem. Specifically, the encoding scheme is obtained by using unsupervised contrastive learning, where a Siamese neural network is trained on a positive (true) topology, its modem data, and a set of negative (false) topologies. The weights of the Siamese network are continuously updated based on a new modified version of the Maximum Parsimony optimality criterion. This approach essentially integrates an optimization problem directly into a deep learning loss function.

We evaluate the encoder's performance on simulated data instances with randomly added events. The performance of the encoder is tested both on its ability to extract and encode events as well as whether the encoded data sequences lead to accurate topology reconstructions under the modified version of the Maximum Parsimony optimality criterion.

Promising computational results are reported for trees with a varying number of internal nodes, up to a maximum of 20. The encoder identifies a high percentage of simulated events, leading to nearly perfect topology reconstruction. Overall, these results affirm the potential of embedding an optimization problem into a deep learning loss function, unveiling many interesting topics for further research.

3.1 Introduction

Broadband technology is one of the most widespread technologies for providing internet access to paying customers. According to CableLabs, one of the leading providers of broadband technology, this technology was the most accessible in both Europe and the US in 2016 [14]. The relatively cheap cost of deployment and continuous innovation suggest that it is going to be an important part of digital infrastructure for many years to come.

Hybrid-Fiber Coaxial (HFC) networks are used to connect users to the internet using coaxial cables in which data is transmitted using radio frequency (RF) signals. Each HFC network is connected to the optical backbone grid through a

local singular conversion node, called a CMC (Coaxial Media Converter), where the signal is converted from optical to electronic form as it transitions from the fiber optic backbone to the coaxial network, and vice versa. Within a given HFC network, each customer is connected to the CMC through a sequence of cable amplifiers and cable splitters arranged in a tree-like structure, allowing the HFC network to cover a local area by gradual branching. In this network architecture, the end connections to the customers' homes are represented by the tree's leaf nodes. The internal nodes are the cable splitters, while the CMC represents the root node. Lastly, the edges in the architecture correspond to cable connections [110].

The RF signal is transmitted using metal-insulated copper wires due to its vulnerability to outside signal interference that would otherwise impair the customer's connection. HFC networks are generally prone to a range of errors, making maintenance an essential part of daily operation. For instance, general degradation can cause one or more customers to have reduced connectivity [173], and weather conditions have been shown to affect the quality of the network [167, 107].

Due to digitization, the complexity of the HFC setup, and time constraints, the infrastructure owner only has partial knowledge of the cabling in most HFC networks, hereinafter referred to as the *topology*. This is problematic since knowing the topology is crucial for network maintenance staff when localizing and resolving network faults. Incomplete topology records lead to a significant increase in both resolution and driving time for the maintenance staff. In their review paper on network monitoring, Lee et al. emphasize that a known and fully updated topology is essential for accurate problem detection [78]. Simaković et al. mentioned that a known topology is needed for monitoring non-intelligent devices, such as amplifiers [143]. Additionally, both Heiler et al. and Simakovic et al. listed a complete topology mapping as one of the data requirements for their approach to root-cause identification and localization of network failures [60, 142], respectively, while Hu et al. planned to utilize the topology for localizing network errors in future work for their HFC anomaly detection algorithm [63]. Due to the scale and the constant changes to HFC networks and digital privacy legislation, it is not viable to manually reconstruct the missing topology data. This makes an automated approach to solving the problem valuable to network owners.

A recent study shows that it is conceptually possible to infer a missing topology of, e.g., an HFC network with high accuracy using discrete time series data collected at modem level only [119]. The authors achieve this by employing a multi-objective approach, incorporating the *Maximum Parsimony* optimality criterion [48] and accounting for the geographic distances between network components. The most *parsimonious* tree topology is the topology that best explains the observed leaf (customer) data in terms of the fewest data muta-

tions. This means that since the RF signal is initially transmitted through a singular root node, one can reconstruct the most probable evolution of the signal throughout the network branches at each time step and count the number of signal mutations [48, 59]. This value is often referred to as the *parsimony score* of a proposed tree topology. The term originates from computational biology, particularly within the field of reconstructing family trees, scientifically known as phylogenetic trees.

The limitation of the above-mentioned study [119] is that its proposed method only works for discrete time series data. Moreover, the observed data must contain distinguishable data mutations such as faults, hereinafter *events*, happening on the network branches over time. Nonetheless, modems predominantly capture continuous time series data, and identifying inherited events within the data and subsequently encoding them into a discrete format is not a straightforward task. As a result, the study [119] focused solely on simulated data.

Inspired by Pisinger & Sørensen [119], our work consequently focuses on extracting significant discrete events from continuous time series data, a task that remains unsolved in this setting. While, theoretically, the parsimony algorithm only necessitates the encoded sequences to be discrete, we consider strictly binary events in this work for simplicity.

Because the parsimony score counts the number of mutations needed to explain the observed leaf data, learning an encoding scheme for the time series using the parsimony score as a loss function is not feasible. No mutations are needed to explain identical data; thus, in this setting, the model will simply encode the same data point for all leaf nodes and arrive at a trivial model, hereinafter *zero-encoder*. Instead, our work proposes using the parsimony score to train a binary event encoder using a contrastive approach [77]. We consider the case where the encoder is constructed using a 1D convolutional neural network architecture, which is then treated as a Siamese network [26]. Hereby, the encoding scheme is similarly applied to all customer modems, meaning a general encoder is learned.

As a preliminary investigation, we aim to analyze the feasibility of the proposed approach using simplified simulated data, where all possible topologies are known, and unique events are simulated on all network edges. We experiment with different data simulation parameters and investigate how the effect of the contrastive approach depends on the time series properties (e.g., length and fault duration) along with network characteristics (e.g., the number of nodes in the network). Moreover, we propose a modified version of the parsimony algorithm to improve the uniqueness of the optimal solution. This modification is based on a new assumption about the state of the data signal at the root-node level. We show evidence that strongly suggests that the modified version leads to better and more unambiguous topology predictions.

The main **contributions** of this paper are:

- An alternative formulation of the tree topology reconstruction problem using leaf node data, also known as the problem of *inferring phylogenies* in Computational Biology. The new formulation adds the assumption that one of the discrete data states is a base-level state representing the root node behavior and indicating an unaffected data point.
- A new algorithm that returns the best possible parsimony score with respect to the maximum parsimony optimality criterion, given the above-mentioned alternative formulation.
- A new conjecture based on computational experiments that states that the true topology, given the new alternative formulation, will always yield a uniquely best parsimony score, provided that at least one isolated distinguishable event occurs on every edge in the tree.
- A novel approach leveraging the synergy between the fields of operations research and deep learning, achieved by integrating an optimization algorithm into a deep learning loss function.
- A contrastive approach for training an encoder that can infer events in continuous time series data that are informative in the new alternative parsimony setting. The end-to-end approach considers the new parsimony algorithm, the true topology of the network, and a set of wrong topologies.
- A stochastic version of the parsimony algorithm to be used with automatic differentiation during the learning stage. This relaxation allows each data point to be represented as a set of probabilities for the different possible states rather than a specific discrete state.
- A method for simulating customer modem time series data under the new parsimony setting, intended for testing the proposed contrastive approach.

The paper is **organized** as follows: In the following two sections, we equip the reader with the relevant theory behind the HFC network setting and the maximum parsimony criterion. Section 3.4 provides an overview of state-of-the-art research related to our problem. In section 3.5 we formally define our problem along with its associated difficulties, and in section 3.6 we present our methodology. Section section 3.7 defines the computational experiments we perform, the evaluation metrics used to assess the computational results, and the data simulation scheme for the experiments. The results of the experiments are presented in section 3.8. Lastly, we discuss the results, the proposed methodology, assumptions, applicability, and future work in section 3.9. Section 3.10 concludes the paper.

Given that this paper caters to both readers interested in operations research as well as deep learning, certain sections might not be of interest to all. Readers with less interest in operations research can skim through section 3.3, subsection 3.5.2 - subsection 3.5.4, and subsection 3.6.1 while readers with less interest in deep learning can skim through section 3.4 and subsection 3.6.2.

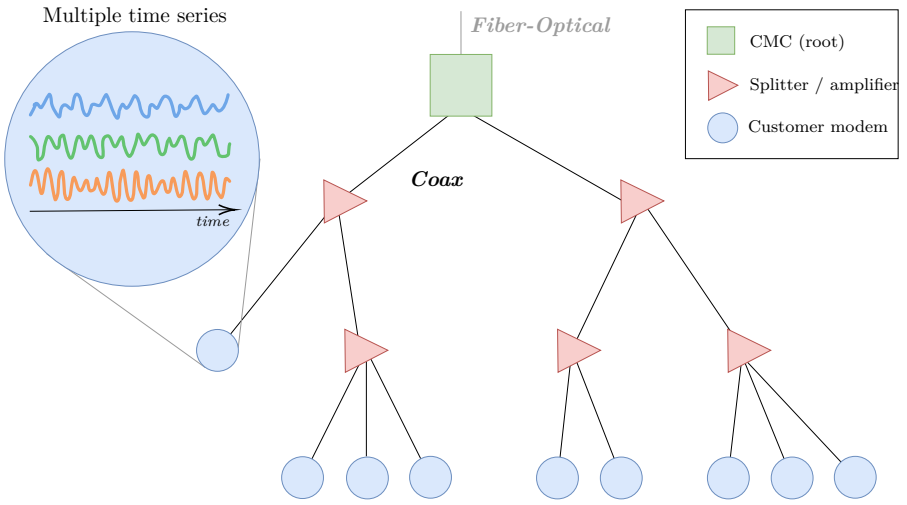


Figure 3.2.1: Illustration of a last-mile HFC network setup. The CMC acts as a local singular node connecting a local network to the backbone internet using fiber-optic technology. Each customer modem is connected to the local CMC through a sequence of amplifiers/splitters (red triangles) linked by coaxial cables. The splitters and amplifiers are usually found close to each other in the same street cabinet. Each customer modem collects a set of performance metrics over time that can be used for monitoring purposes.

3.2 The HFC network

The backbone (country-spanning) network consists of a set of Cable Modem Termination Systems (CMTSs) between which information is transmitted using optical fiber technology in which optical wires are used to transmit data using flickering light [131]. Locally, each CMTS is connected to a set of Coaxial Media Converters (CMCs), also referred to a *fiber nodes*, in which the signal is converted from a fiber signal to an electronic RF signal as part of the transition from the backbone network to the coaxial network, hereinafter *last-mile network*, and vice versa. These CMCs are used as local singular nodes to which all customer modems in a local area are connected. This connection enables data transfer from the backbone internet to the customer modem, i.e., downstream (DS), and the other way, i.e., upstream (US). Each customer modem is connected to a CMC through a sequence of cable amplifiers (amplifying the RF signal) and cable splitters linked by coaxial cables. Usually, the splitters and amplifiers of a last-mile network sit close together in the same street cabinet, making it reasonable to illustrate both the amplification and splitting of the signal by a singular process (node). An illustration of a last-mile HFC network is provided in Figure 3.2.1.

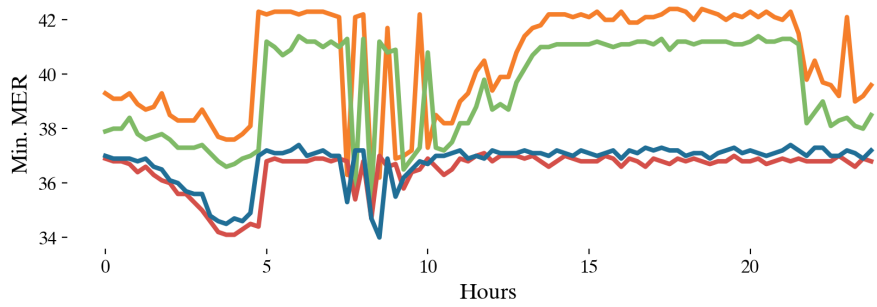


Figure 3.2.2: Example of 24 hours of time series data from four different customer modems consisting of minimum MER values during the last 15 minutes. It appears that there is a baseline signal that all modems adhere to. However, during certain periods, some customers experience signal distortion due to specific events. For example, between approximately hours 7 and 8, the signal for all customers is prone to increased variation with varying degrees. Conversely, during hours approximately 10 to 13 and 22 to 24, only the signal for the green and orange customers is affected.

3.2.1 Time series data

Due to the many potential challenges faced in the daily operation of the HFC network, multiple metrics are sampled regularly from each of the modems in the network. These metrics are specified in the (Data-Over-Cable Service Interface Specifications (DOCSIS) 3.0 [15] and include Modulation Error Ratio (MER), power levels, the number of bits received during the last poll period, the number of corrupted bits, and other metrics that are used to perform surveillance of the quality of the signal over time. Typically, these metrics are aggregated through polling, utilizing statistical methods or summation for each polling period, since continuous real-time monitoring is often impractical due to the significant volume of data that would need to be stored. See Figure 3.2.2 for an example of time series data from four different modems in the same HFC network. These time series consist of minimum MER values during the last polling period (approximately 15 minutes) over a total of 24 hours. MER is believed to be a good indicator of correlation between modems, as it reflects the cumulative effect of the entire traversed path on the DS signal.

3.2.2 Faults and noise propagation

The complexity and vulnerability of the coaxial part of the HFC network, especially, make it prone to degradation and various types of errors. This means that many approaches have been proposed to identify and locate errors in the network [60, 63, 76, 135, 142, 166]. A typical error influencing broadband net-

works is the so-called Common Path Distortion (CPD) caused by connectors that have been affected by stress or corrosion [145]. In general, problems in the HFC network are characterized by being sporadic and difficult to detect due to the many potential root causes. Those include physical properties in the wires, weather factors like temperature and humidity, outside RF interference, electromagnetic interference, and improper customer equipment [124, 125]. Some errors are due to problems in or near the network amplifiers, meaning that all customer modems located topologically beneath a problematic amplifier will be affected in the DS case [176, 49]. An example of this is visible in Figure 3.2.2, where all customers seem to adhere to some baseline signal, but where some or all customers are affected by specific events during periods of time.

3.3 Maximum parsimony

The study of evolutionary relationships, scientifically known as *phylogenetics*, is a heavily researched discipline within biology, dating back to the 1800s when early evolutionists such as Darwin sketched the first evolutionary trees to represent historical relationships among living species [57]. The reconstruction of phylogenetic trees is typically based on genome (DNA/RNA) sequences from a set of living species, sometimes accompanied by some assumed evolutionary model. Here, the living species represent the terminal nodes of the phylogenetic tree, the internal nodes represent extinct ancestors, and the branches depict evolutionary relationships and relatedness.

Numerous methods exist for reconstructing the tree topology [44], where the best choice of approach depends on the available data and evolutionary setting. Maximum parsimony is one of the oldest and most applied methods both praised and criticized for its simplicity and lack of assumption involving underlying evolutionary models [150]. The maximum parsimony principle assumes that the tree which provides the simplest evolutionary explanation is the correct one. Specifically, it aims to minimize the number of necessary data mutations across all branches in the tree, enabling the transformation from a shared ancestral data sequence to each of the observed leaf sequences.

Different variations of evolutionary trees occur in many other problems than biology, one being the reconstruction of coax networks using DS data. In the same way that DNA sequences change over time, the coaxial signal is distorted based on which physical component it encounters on its way from the CMC root node to each customer terminal node.

3.3.1 Finding the most parsimonious tree

Determining which tree is the most parsimonious is NP-hard in many of its variants, including the rooted setting [38, 44]. While calculating the parsimony

score of a particular tree is achievable in polynomial time, what makes the problem difficult is that, in theory, one must consider all possible topologies and their respective parsimony score to determine the most parsimonious tree(s). Given that the number of possible general tree topologies for n nodes is n^{n-2}, also known as Cayley's formula [20], employing a brute force approach quickly becomes infeasible as the tree size increases.

All approaches to computing the best parsimony score of a specific tree are based on dynamic programming, which makes the maximum parsimony approach computationally efficient compared to other tree reconstruction methods [95]. Fitch [48] famously developed the first algorithm for binary trees, Hartigan [59] developed a more general approach not limited to binary trees, and Sankoff's algorithm [138] addresses the weighted variant of the problem, in which a cost matrix is given as input, stating the cost of switching between all possible pairwise data states.

3.3.2 The uniqueness of the most parsimonious tree

Although the aim is to reconstruct the one true topology, no reconstruction method can do so for all imaginable types of data sets. In many cases, the best tree topology will not be unique, and the true tree might not even be amongst the best solutions provided by some chosen reconstruction model [47]. Jussi & Teemu [95] employed a large-scale computational study with simulated phylogenetic data to estimate the probability that maximum parsimony uncovers the true tree topology and concluded that it performs well under a simple data-generating model if the rate of change, i.e. the number of events, is sufficiently small and, crucially, if the length of the data sequences are sufficiently long.

In certain special cases, it has been demonstrated that various reconstruction principles, including maximum parsimony, consistently reveal the true tree topology unequivocally. One such example is based on Buneman's Theorem [11]. Here, in the setting of an unrooted binary tree, if the data in the leaf nodes corresponds to precisely one event occurring on each edge at each time step, then the tree that generated the data will be the unique, most parsimonious tree. Later, Fischer [47] proved this to also be true in the case where an event always occurs on two edges at a time and where this happens to exactly all two pairwise edge combinations.

3.4 State-of-the-art

Even though a known topology is very useful for Internet Service Providers, as previously mentioned, previous research on reconstructing missing topologies is very limited. To the best of our knowledge, the only paper previously published on the matter is from Pisinger & Sørensen in 2024 [119], using the maximum

parsimony criterion, as previously discussed in the introduction. Though the maximum parsimony approach to construct and study phylogenetic trees goes back many years, as described in section 3.3, the work of Pisinger and Sørensen is believed to be the first time this approach has been applied to the reconstruction of HFC-network topologies. The work had very promising results; however, it was only evaluated on discrete simulated data, highlighting the need for a way to extract events from continuous data.

3.4.1 Representation learning in time series

The extraction of discrete events from multiple time series can be understood as equivalent to learning representations of the time series data. This field has only recently gained momentum due to the complexity of the problem and the incomprehensible nature of time series in general, which makes it hard to interpret the outcome and assess the usefulness of the learned representations. Representations are usually learned in an unsupervised manner, in which an encoder is learned with some unsupervised goal in mind.

Recurrent Neural Networks (RNN) have traditionally been used as an encoder for various time series tasks, such as classification and representation learning. For instance, Malhotra et al. used RNNs in an encoder-decoder setting to represent a typical time series behavior and use that to search for anomalies [83], while Malhotra et al. trained a Variational AutoEncoder (VAE) based on RNNs on a set of different datasets to produce a generic time series feature extractor [84].

Others have used different approaches to encode the time series. For instance, Hyvärinen & Morioka [64] divided a time series into several chunks with temporal labels and trained a neural network model that would enable the temporal classification of the time series chunks using logistic regression. Lei et al. used matrix factorization techniques to produce representations that mimic the distances between time series obtained via dynamic time warping [79].

However, recent work suggests that one-dimensional *Convolutional Neural Networks* (CNN) is state of the art in various time series tasks while also being relatively easy to train and interpret [7]. This has resulted in multiple studies using CNNs for time series classification [23, 165], but also for representation learning tasks. Emadeldeen et al. used two types of augmentation based on permutation and scaling, respectively, to learn representations that are contrastive with respect to both time and context [41]. Challu et al. used hierarchical latent factors to represent time series at different scales in a generative model [21]. Lastly, Franceschi et al. used exponentially dilated convolutions and a triplet loss to learn representations that are close to those of sub-sequences of an anchor time series, while far from those of other time series [53].

Nevertheless, all the methods mentioned above and many time series representation methods in general are based on continuous representations, whereas this

work requires discrete representations.

3.4.2 Discrete latent representations

A few studies have been conducted where comprehensible discrete representations were the goal. Van den Oord et al. [160] developed a framework for learning discrete representations based on a continuous embedding space with discrete labels, where the embedding nearest to the encoded input would become the discrete latent representation. This approach, however, was not used on time series. Additionally, the predefined size of the embedding space makes it inflexible to inputs of various sizes that time series are likely to constitute.

A recent work carried out by Fortuin et al. studied discrete representations of time series [50] by applying a Self-Organizing Map (SOM) to the latent space of the encoded values in a VAE setting. This method, however, learns single representations for an input consisting of a sliding window of a time series, but for our work, no assumptions can be made on the length of events. Additionally, this method would not guarantee latent representations that are informative in a parsimony setting.

3.5 Problem Statement

We begin this section by formally introducing the problem. Next, we introduce the most parsimonious tree and show how the parsimony score is calculated. Thirdly, we introduce the issue of obtaining a unique solution, and lastly, we show how the frequency of events in the modem data complicates the problem further.

3.5.1 Problem

Let graph $G = (V = \{N, M\}, E)$ be a general tree corresponding to a last-mile HFC network with a set, N, of internal nodes (cable splitters and amplifiers), a set, M, of leaf nodes (customers), and a set edges, E (cable connections). Given the tree property of the graph, we have that $|E| = |V| - 1$, where in this work, $|\bullet|$ represents the cardinality of a set.

Every modem $m \in M$ is associated with input data $\mathbf{X}_m \in \mathbb{R}^{|F| \times |T|}$ comprising F features, each recorded over a duration of T time steps. The features are various *continuous* metrics describing the quality of the DS signal received by each modem at specific time points.

For each modem, an encoded version of the data is generated using an encoder $f(\mathbf{X})$. The encoded data is represented by a $|F'| \times |T'|$ matrix[1] where each

[1] Where $|F'|$ and $|T'|$ are used to specify the dimensions of the outcome of the encoder,

element is a *discrete* event in the alphabet of binary states $\Omega = \{0, 1\}$. Each state, in theory, corresponds to a physical interpretation, but interpretations can be quite abstract. A simple interpretation could be that state 0 may represent a standard signal, while state 1 may indicate an abnormal signal. All results can readily be generalized to include a larger set of states. We also establish an encoded data sequence $\mathbf{z} \in \Omega^{|B|}$ for each modem, arranging the encoded matrix sequentially in a single dimension, with $B = F' \times T'$ (Cartesian product). Here, $\mathbf{z}$ straightforwardly represents all the encoded features in vectorized form. We denote the stacked collection of all modem sequences in a graph G as $\mathbf{Z} = \Omega^{M \times B}$.

Then, the **aim of this paper** is to develop an encoder, $f(\mathbf{X})$, that, based on the maximum parsimony principle, can transform continuous modem data, $\mathbf{X}$, into a collection of binary data sequences $\mathbf{Z}$. This encoder should ensure that the tree topology, from which the modem data originates, will be the singularly most parsimonious tree with respect to all the encoded modem data sequences, $\mathbf{Z}$.

In this work, we assume, without loss of generality, that all leaf connections are known. Consequently, we only consider the set of tree topologies that emerge from reconstructing the *internal edges* of a last-mile network. We make this assumption based on a first-hand account from Denmark's biggest telecommunication infrastructure owner, TDC NET. Moreover, we also assume that every internal node has at least one child which is a modem (leaf) node. Although this assumption does not hold for all real-life last-mile networks, it is necessary to prevent the emergence of isomorphic topologies resulting from swapping two interchangeable 'modem-less' internal nodes. Lastly, we assume data events occur uniformly distributed across all internal edges and that events propagate undisturbed down through the network.

3.5.2 The most parsimonious tree

The most parsimonious tree is the tree, G, associated with the lowest *parsimony score* $\mathcal{P}(\mathbf{Z}, G)$ based on a set of discrete data sequences, $\mathbf{Z}$; one sequence $\mathbf{z} \in \Omega^{|B|}$ for each modem. Here, the parsimony score of a tree G is given as the sum of the pairwise Hamming distances between all nodes connected by an edge in G. This equates to inferring the data states, z_i, of the internal nodes and thereby minimizing the number of times the root node signal must change through the sequences of nodes in the tree, at every time point, to explain the observed modem data:

$$\mathcal{P}(\mathbf{Z}, G) = \left\{ \begin{array}{lll} \min & \sum_{(u,v)\in E} d_H(\mathbf{z}_u, \mathbf{z}_v) & \\ \text{s.t.} & \mathbf{z}_i \in \Omega^{|B|} & \forall \quad i \in N \end{array} \right\} \tag{3.1}$$

which can be—but does not have to be—of the same dimensions as the input data $\mathbf{X}_m$.

where $(u, v) \in E$ are all edges in G between any two nodes u and v, and d_H, otherwise known as the Hamming distance between two vectors, is defined as:

$$d_H(\mathbf{z}, \mathbf{z}^*) = \sum_{b=1}^{B} \mathbb{1}_H(z_b, z_b^*) \quad \text{where} \quad \mathbb{1}_H(z_b, z_b^*) = \begin{cases} 0 & \text{if} \quad z_b = z_b^* \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

In short, the parsimony score is the sum of all differences in data between any two nodes connected by an edge in G.

The observant reader might notice that $\mathcal{P}(\mathbf{Z}, G)$ is not readily computable, since the data sequences $\mathbf{z}$ are known only for the modem nodes $u, v \in M$. Luckily, polynomial time algorithms exist for determining which sequence(s) $\mathbf{z}$ for each internal node $u, v \in N$ leads to the minimal parsimony score. These algorithms were briefly discussed in section 3.3. Since this work considers general trees with binary data states, Hartigan's algorithm [59] is the most appropriate choice.

Hartigan's algorithm is a dynamic programming approach where each internal node is systematically considered in a bottom-to-top order, determined by a postorder traversal of G. Furthermore, all parsimony algorithms compute the score for each time point in $\mathbf{Z}$ independently before aggregating the score contributions.

For every internal node $v \in V$, the algorithm exclusively assesses the set of direct descendant nodes D_v. Here, the optimal state of node v is determined by the majority state observed in its descendant set, where multiple states can be optimal in the case of a tie. The cost associated with node v reflects the number of descendants assigned a different state than v, each signifying that a data mutation occurs. In other words, the cost corresponds to the number of descendant nodes conflicting with the majority vote.

The parsimony score for the entire tree with respect to a single data point is then obtained by summing the costs of all internal nodes $v \in V$. Alternatively, as this process simultaneously determines the optimal data state(s) of all internal nodes, equation (3.1) presents a different, but equivalent approach, to computing the optimal parsimony score. A formal outline of the procedure for a single time point is provided in the supplementary material 3.B and an example of calculating the optimal parsimony score of a given tree is given in Figure 3.5.1.

3.5.3 Solution uniqueness

In the Maximum parsimony section, we briefly discussed that for unrooted trees, there exist special cases of data generation where the most parsimonious tree is guaranteed to be unique. However, in this section, we demonstrate a counterexample, proving this is *not* the case for rooted trees.

In the unrooted setting, one such special case occurs when exactly one event occurs on every edge at independent points in $\mathbf{Z}$, as discovered by Buneman [11].

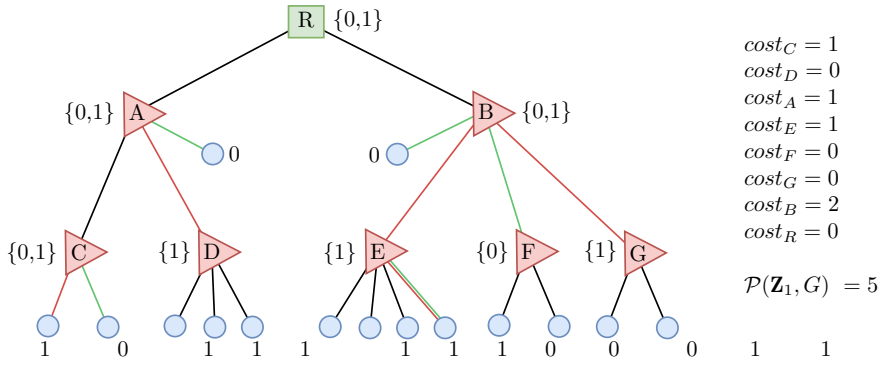


Figure 3.5.1: An example of how to compute the optimal parsimony score for a specific tree G using algorithm 3.3 in which the internal nodes are considered one by one with respect to a postorder traversal C, D, A, E, F, G, B, R. Here, a single time point in the input data sequences $\mathbf{Z}$ is considered. The blue nodes are customers, the red nodes are splitters/amplifiers, and the green node is the CMC. The computed optimal state sets for the internal nodes are listed in curly brackets.

Node C has two children with differing states, resulting in a tie for the majority state. Consequently, the optimal state for C can be either state 0 or 1, denoted by $\{0, 1\}$. Moreover, the cost of node C is $cost_C = 1$, as one child node will always disagree with C, regardless of whether state 0 or 1 is the true state of C. Node D has three children in which the majority state unanimously is state 1. Thus, the optimal state set for node D is 1 and $cost_D = 0$. Node A has three children in which state 0 occurs twice (in node C and the modem node), and state 1 also appears twice (in node C and D). Thus, there is a tie for the majority, and the optimal state set of node A becomes $\{0, 1\}$. Regardless of the true state in node A, one child will always disagree, leading to $cost_A = 1$. The majority state in the children of node E is state 1, but one child disagrees with this state, so the optimal state of E is 1 with $cost_E = 1$. The children of both nodes F and G unanimously agree on the majority state, just as in node D. Node B has four children in which both states 0 and 1 appear twice. Consequently, the optimal state set of node B is $\{0, 1\}$ with a cost of 2, as two children will always disagree with the state of B. Lastly, the root node R has two children, both with an optimal state set of $\{0, 1\}$, meaning the optimal state set of the root is also $\{0, 1\}$ with $cost_R = 0$.

Summing over all node costs gives an optimal parsimony score for tree G of $\Sigma_{v \in N} cost_v = 5$. However, this does *not* mean that G is the most parsimonious tree for the given modem data. Considering the simplified version of the problem where the topology of only the internal edges is unknown, there are still $|N|^{|N|-2} = 262,144$ other possible topology configurations that might yield a better parsimony score.

Two reconstructions of the data states in the internal nodes exist, both of which yield a parsimony score of five. Letting the optimal state be 0 in all nodes with an optimal state set of $\{0, 1\}$, leads to the five data mutations illustrated by the red edges. Oppositely, letting the optimal state be 1 in all nodes with an optimal state set of $\{0, 1\}$, leads to the five data mutations illustrated by the green edges.

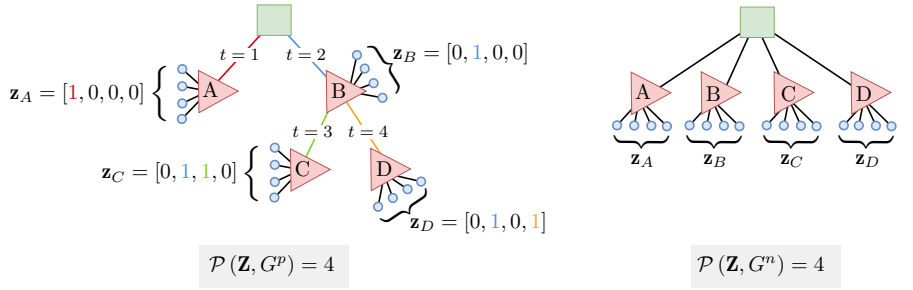


Figure 3.5.2: A counterexample showing that a special data case which results in a unique most parsimonious tree in an unrooted setting [11], does not yield a unique tree in a rooted setting. Here, the left tree is the true tree, G^p, for which an event has been simulated to occur on each internal edge at independent times. Given that there are four internal edges in G^p, the parsimony score of G^p is also four. However, the alternative tree topology to the right, G^n, yields the same optimal parsimony score of four for the same modem data, proving the true tree is not uniquely the most parsimonious.

Figure 3.5.2 (left) illustrates this case for a rooted tree G^p with five internal nodes and modem data sequences of length $|B| = |E| = 4$. At each time point $t \in B$, a new edge e is chosen, and $\mathbf{z}[t]$ is assigned state 1 for all modems below edge e. In the figure, the color of the affected data point corresponds to the color of the edge on which the event was simulated. The resulting modem data, $\mathbf{Z}$, generated by this approach is depicted in Figure 3.5.2. The optimal parsimony score of tree G^p equals the number of simulated events, namely four. However, as shown on the right in Figure 3.5.2, an alternative tree topology G^n also yields an optimal parsimony score of four for the same input data $\mathbf{Z}$, if the root node data is reconstructed as $[0, 1, 0, 0]$.

Although it is unlikely that an encoder can learn to encode a special case of data that generates a unique, most parsimonious tree, it remains problematic that it is not even theoretically possible to do so. While it is improbable for an encoder to learn to recognize and encode a special structure in the data and always produce a unique, most parsimonious tree, the inability to achieve this, even theoretically, is a concern. Therefore, we will introduce a new assumption and propose a new modified parsimony approach in the Methodology section to attempt to circumvent this issue.

Lastly, we also find it important to emphasize that it directly follows from the definition of the parsimony score that some form of inheritable event must occur on *every* edge in a tree G, at least once in the modem input data $\mathbf{Z}$. If an edge $e = (u, v)$ is never affected by an event, the data signal passing through nodes u and v will be identical. Consequently, replacing edge e with $e' = (v, u)$—that is, switching the order of nodes u and v—will result in a new topology, G', indistinguishable from G. Assuming faults occur at random positions in a

network, collecting data over a sufficiently long time period should ensure all edges are affected at least once.

3.5.4 Events

If $|\Omega|$ different data states are observed simultaneously in $\mathbf{Z}$, it will require at least $|\Omega| - 1$ data mutations (events) to explain the data. For example, in the case of binary data, if both states 0 and 1 are observed in different modems at the same time, then the original state of the signal in the root node must have mutated at least once to explain the two states observed at leaf level. This means that if only singular independent events occur at each data point in $\mathbf{Z}$, then there does *not* exist any topology G^n with a better parsimony score than the true topology G^p.

However, if multiple identical events, hereinafter *multi-faults*, occur simultaneously, an alternative tree topology G^n can sometimes outperform the true topology G^p, that generated the modem data. Such an example is illustrated in Figure 3.5.3. Here, at time $t = 4$, the last point in $\mathbf{Z}$, an event occurs on two different edges, where in both cases, the modem data is affected in the same way, i.e., it mutates from state 0 to state 1. As a result, the alternative topology G^n to the right in Figure 3.5.3 is able to group the modems affected by the two simultaneously occurring events under one single edge, (B, D), such that a single event can explain the observed data mutations instead of the two events that occurred in reality.

A set of states more fine-grained than binary could, in theory, resolve the issue of multi-faults. However, in practice, an encoder would then need to learn not only to identify abnormalities in the modem data but to also categorize or at least tell apart different abnormalities. This task is much more challenging and could easily lead to overfitting.

If, in fact, two identical events occur at the same time, i.e., two cables buried next to one another are both cut through, then the maximum parsimony principle fails. Assuming events are not that frequent, it is much more likely that one single event causes a signal mutation in a set of modems rather than two or more identical events occurring simultaneously. The optimal tree, with respect to the maximum parsimony principle, is the tree that can explain the observed leaf data with the *least* amount of mutations, as formalized in (3.3). Therefore, although the left tree in Figure 3.5.3 is the true tree that generated the data $\mathbf{Z}$, according to the maximum parsimony principle, it is more likely that the data originates from the right tree.

Since we in this paper have chosen to work with the maximum parsimony principle, there is not much we can do to combat multi-faults in our methodology. Instead, we have chosen to show the impact on *topology accuracy* (3.9) (defined in a later section) for an increasing amount of multi-faults in the modem data.

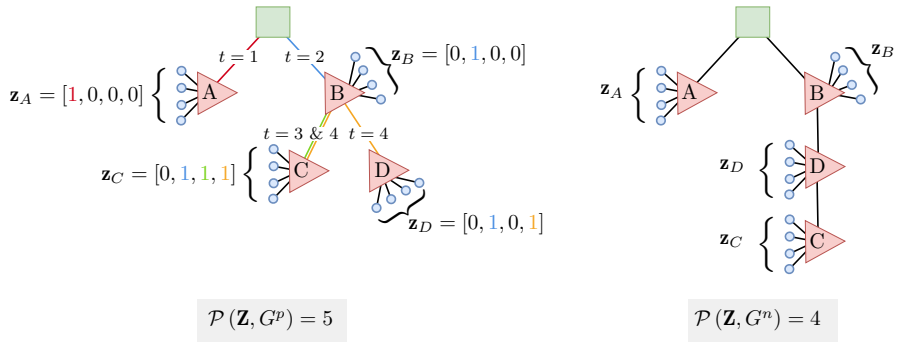


Figure 3.5.3: An illustrative example of the challenge posed by *multi-faults* in a maximum parsimony framework. The network on the left displays the true topology, G^p. On each edge, t indicates when an event occurred on the edge, resulting in a change from the base data state of 0 to 1 in the data sequence of all affected modems. At time $t = 4$, a multi-fault occurs on edge (B, C) and (B, D). The optimal parsimony score of G^p is five: one in each time step with a single fault and two in the time step with a multi-fault.
In the alternative network topology to the right, G^n, the same modem data yields an optimal parsimony score of four, which is better than the score obtained by the true topology. This is because G^n groups all modems affected by the multi-fault under a single edge, (B, D).

3.6 Methodology

In this section, we present our proposed methodology. First, we introduce a new root-node assumption and a modified parsimony algorithm, based on which we also propose a new conjecture. Secondly, we prove that the new modified parsimony algorithm yields the optimal parsimony score given the new root-node assumption. Thirdly, we introduce a contrastive learning approach and a loss function based on the new modified parsimony algorithm; we also introduce a solution to overcoming non-differentiability during the learning phase. Lastly, we introduce a set of neighborhood functions for sampling topologies from the set of all possible topology reconstructions for a given tree size.

3.6.1 Modified parsimony Algorithm

To circumvent the inability to obtain unique solutions, we propose a new assumption, i.e., that the data states in the root node, $\mathbf{z}_{root}$, are known just like the leaf nodes. At all time points, the root node represents some normal *baseline* signal. Even if the signal received by the root node from the backbone grid at some point t is corrupted, any additional faults in the last-mile network will accumulate on top of this initial corruption, acting as the baseline signal at time t.

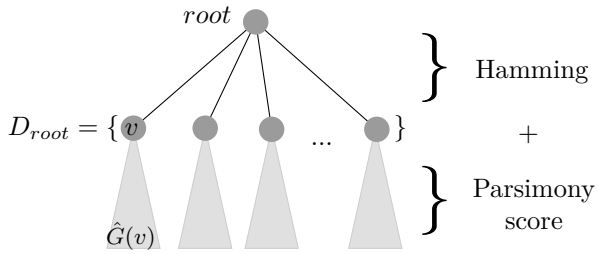


Figure 3.6.1: Illustration to aid understanding of new modified parsimony score and proof of Algorithm 3.1

We propose a new modified parsimony algorithm (algorithm 3.1) identical to the normal parsimony algorithm (see supplementary material 3.B), except for a few steps. Here, when v becomes the root node in the for-loop in line 4, instead of letting the majority state in the children of the root dictate the optimal state of the root, we let the cost of the root be equal to the number of children disagreeing with the presumed input state of the root. This can be formally stated as:

$$\hat{\mathcal{P}}(\mathbf{Z}, G) = \left\{ \begin{array}{lll} \min & \mathcal{P}^{\text{sub}}(\mathbf{Z}, G) + \sum_{v \in D_{root}} d_H(\mathbf{z}_{root}, \mathbf{z}_v) & \\ \text{s.t.} & \mathbf{z}_i \in \Omega^{|B|} & \forall \quad i \in N/\{root\} \end{array} \right\} \tag{3.3}$$

where $\mathcal{P}^{\text{sub}}(\mathbf{Z}, G)$ is the sum of the parsimony scores of all subtrees, $\hat{G}(v)$, rooted in the set of nodes $v \in D_{root}$ which are the direct descendants of the root node. Moreover, $\hat{\mathbf{Z}} \subseteq \mathbf{Z}$ is the set of discrete data sequences belonging to each modem in subtree $\hat{G}(v)$:

$$\mathcal{P}^{\text{sub}}(\mathbf{Z}, G) = \sum_{v \in D_{root}} \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v)) \tag{3.4}$$

Here, the objective of (3.3) minimizes both the aforementioned sum of subtrees as well as the Hamming distance between the root node to each of its direct descendants. A proof demonstrating that Algorithm 3.1 always yields the optimal parsimony score is enclosed in the supplementary material 3.A.

In Figure 3.5.2, the base state of the root was assumed to be zero at all points, and for every simulated event, we let the data mutate from zero to one. Using the modified algorithm 3.1 on this example, assuming $\mathbf{z}_{root} = [0, 0, 0, 0]$, yields a unique most parsimonious score for the true tree to the left in the figure.

Algorithm 3.1 Modified Parsimony Score for tree G with modem data $\mathbf{z}$ for a single time point

procedure MPARSIMONY($G = (V = \{M, N\},\ E)$, $\mathbf{z} \in \Omega^{|M|}$, $\omega_{root} \in \Omega$)
  $\mathbf{z} \leftarrow \left[\mathbf{z},\ \text{NULL}^{|N|}\right]$ ▷ *Expand* $\mathbf{z}$ *with placeholder for internal nodes*
  $\bar{N} \leftarrow$ POSTORDERTRAVERSAL($G[N]$) ▷ *"Bottom-up" sorting of internal nodes*
  for $v \in \bar{N}$ **do**
    $D_v \leftarrow$ set of child nodes of v
    for $\omega \in \Omega$ **do** ▷ *For each character in the set of states,* Ω
      $\phi_\omega \leftarrow$ frequency of $\omega \in \mathbf{z}(D_v)$
    if v is root **then**
      $cost_v \leftarrow |D_v| - \phi_{\omega=\omega_{root}}$
    else
      $\mathbf{z}_v \leftarrow \arg\max_{\omega\in\Omega}(\phi_\omega)$ ▷ *Character state(s) of* v *becomes the most frequent*
      $cost_v \leftarrow |D_v| - \max_{\omega\in\Omega}(\phi_\omega)$ ▷ *Number of children disagreeing with majority*
  return $\sum_{v\in N} cost_v$

3.6.1.1 Uniqueness conjecture

Based on algorithm 3.1, and inspired by the data setting of Buneman [11], we propose the following conjecture:

Conjecture 1 *Given Condition 1 (below), the Modified Parsimony Score algorithm presented in Algorithm 3.1 always yields a unique best parsimony score for the true tree topology.*

Condition 1 *A tree topology $G = (V = \{M, N\}, E)$ with leaf nodes, M; nodes, $M \cup N$; and edges, E, where the root node and each leaf node contains a discrete data string, $\mathbf{z}$ from an alphabet of discrete states, Ω. Every data point in every leaf data string must be inherited from the root through the branch connecting them. If a data point encounters an 'event' through its branch traversal it is modified to a new discrete state $s \in \Omega$, different from the root state. At least one event must occur on each edge of the tree at an independent point in the data string, and no more than one event can occur at the same time. Consequently, the length of each data string, $\mathbf{z}$, must be at least $|E|$.*

The validity of the conjecture will be challenged through rigorous simulated tests described in section 3.7.

3.6.2 Contrastive Learning Approach

Since our approach intends to encode the continuous time series data from the customer modems in the network into a sequence of discrete events, there is no definitive ground truth labeling for the output events. This means that the problem will be unsupervised. In the meantime we intend to learn an encoder, f, to be used on any customer modem individually, thus will only depend on

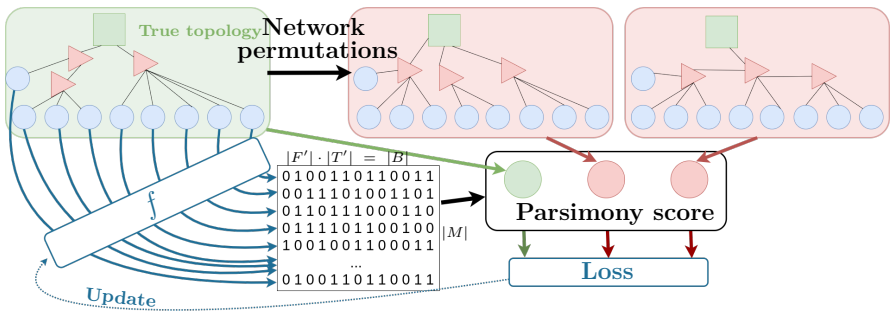


Figure 3.6.2: Illustration of the proposed algorithm. A Siamese network is trained using the parsimony score on a positive (true topology) sample (green) and a set of negative samples (red). The time series from each of the M customers are sent through the same encoder f that encodes continuous time series into an $M \times B$ matrix of binary events. Using this matrix and each of the different topologies, the parsimony scores can be computed. Based on these the loss is calculated and used to update the encoder.

the continuous time series data. We will use a contrastive approach, comparing a positive (true) sample to a set of negative (false) samples, based on the parsimony score (see subsection 3.6.1) to learn an encoder that can extract the most useful events from the time series data in the parsimony setting. This means that we intend the model to encode events with two properties. Firstly, the encoded events should result in the lowest parsimony score when calculated using the true topology (positive sample) as opposed to any other possible topology (negative sample). Secondly, to be able to differentiate between any two topologies, their respective parsimony scores calculated based on the encoded events, should be as different as possible. This could be achieved by minimizing the total parsimony score using the true topology while maximizing a pairwise distance measure between the parsimony scores of any two different topologies. Our contrastive approach is visualized in Figure 3.6.2.

In the following, we describe our approach in the case where the alphabet, Ω, consists of binary events. We choose this for simplicity, not at a loss of generality, as it is straightforward to add an extra dimension for multiple encoded event states. Additionally, the parsimony score is defined for discrete events, not exclusively binary events.

3.6.2.1 Proposed Loss Function

Let $\mathbf{X}_m$ be the $|F| \times |T|$ matrix of time series measurements from the mth modem, that is $|F|$ features measured on $|T|$ occasions. Let f be a Siamese neural network that encodes the multiple time series from each customer modem in an HFC network into a matrix (sequence) of events of size $|B| = |F'| \times |T'|$.

Because the parsimony algorithm treats each data point individually, each of these matrices, $f(\mathbf{X}_m)$, can be flattened and stacked to produce the final encoded event matrix of all modems in the network, $\mathbf{Z}'$, of size $|M| \times |B|$. This matrix is given by:

$$\mathbf{Z}' = \begin{bmatrix} f(\mathbf{X}_1)_1 & f(\mathbf{X}_1)_2 & \dots & f(\mathbf{X}_1)_{|F'|} \\ f(\mathbf{X}_2)_1 & f(\mathbf{X}_2)_2 & \dots & f(\mathbf{X}_2)_{|F'|} \\ \vdots & & & \\ f(\mathbf{X}_{|M|})_1 & f(\mathbf{X}_{|M|})_2 & \dots & f(\mathbf{X}_{|M|})_{|F'|} \end{bmatrix} \quad ; \quad f(\mathbf{X}_m)_j^\top = \begin{bmatrix} f(\mathbf{X}_m)_{j,1} \\ f(\mathbf{X}_m)_{j,2} \\ \vdots \\ f(\mathbf{X}_m)_{j,|T'|} \end{bmatrix} \tag{3.5}$$

where $f(\mathbf{X}_m)_j$ is the jth feature in the encoded events matrix measured for modem m. To make the encoded events binary, f should be chosen such that the output values are in the range $[0, 1]$ allowing for subsequent rounding to a binary value:

$$\mathbf{Z} = \lfloor \mathbf{Z}' \rceil \tag{3.6}$$

Let $\hat{\mathcal{P}}(\mathbf{Z}, G)$ be the modified parsimony score (3.3), and $\hat{\boldsymbol{\mathcal{P}}}(\mathbf{Z}, G)$ the unsummed *vector* of modified parsimony scores with respect to each time point (column) in the encoded binary events matrix $\mathbf{Z}$ using the tree described by $G = (V = \{M, N\}, E)$. We propose the loss function given by:

$$\mathcal{L}(\mathbf{Z}, G^p, \mathcal{G}^n) = \frac{\hat{\mathcal{P}}(\mathbf{Z}, G^p)}{\frac{|E|}{2} \cdot |B|} + \alpha \sqrt{\frac{1}{K(K+1)} \sum_{g_1 \in \mathcal{G}} \sum_{g_2 \in \mathcal{G} \setminus g_1} \frac{1}{\|\hat{\boldsymbol{\mathcal{P}}}(\mathbf{Z}, g_1) - \hat{\boldsymbol{\mathcal{P}}}(\mathbf{Z}, g_2)\|_2^2 + 1}} \tag{3.7}$$

where N is the set of internal nodes, G^p is the true topology, $\mathcal{G}^n = \{G_1^n, \dots, G_K^n\}$ is the set of K negative (sampled) topologies, and $\mathcal{G} = \{G^p\} \cup \mathcal{G}^n$. The first term corresponds to the parsimony score of the true topology and is normalized by the maximum number of data mutations that can occur with respect to the parsimony principle[2]. Meanwhile, it is also natural to normalize the contrastive term by the number of comparisons performed, i.e. the number of contributions in the nested sums; $\frac{1}{2} \cdot K \cdot (K+1)$, and the number of contributions in the sum of squares, $|B|$. Doing this also makes the coefficient α independent of the number of negative samples used in the training loop.

This loss function considers the difference between resulting parsimony scores using all pairwise topology samples, rewarding large score differences and punishing small ones. We calculate these distances in the $|B|$-dimensional space $\left(\mathbb{N}^{|B|}\right)$ allowing for more flexibility in the encoded events.

[2]In the case of binary data states, $|\Omega| = 2$, the maximum number of mutations that can occur is $\frac{1}{2}|E|$. For every internal node in a tree, a mutation occurs for each child node disagreeing with the majority state. The majority will always include at least half of the children, thus a mutation can at most occur on half of all edges. It follows from the same argumentation that if $|\Omega| = 3$, the maximum number of mutations is $\frac{2}{3}|E|$, for $|\Omega| = 4$ it is $\frac{3}{4}|E|$, etc.

3.6.2.2 Overcoming the Non-Differentiability

The modified parsimony algorithm described in subsection 3.6.1 is so far constrained to discrete time series events. Since we use the modified parsimony algorithm directly in our loss function given in (3.7), the rounding operation will be part of the standard forward pass. For the back-propagation algorithm to work, gradients need to be computed with respect to all weights in the network individually [134]—also when utilizing automatic differentiation in e.g. pytorch [109]. Due to its nature, the rounding operation is not differentiable because it is not continuous in its domain.

We overcome this challenge by defining a special forward pass for training in which we let the encoded events take values in $[0, 1]$, thereby representing probabilities. To train the model, we additionally propose a slightly modified version of the parsimony score that allows for continuous events, meaning that the distances in (3.7) will be calculated in $\mathbb{R}_+^{|B|}$. Since all operations will be carried out in the continuous space, this version will allow for the calculation of gradients throughout the full model.

3.6.2.3 Modified Continuous Parsimony Score

The intuition behind the continuous version of the modified parsimony score is to keep track of each event state individually and for each leaf node to encode the probability of the customer modem being in any given event state. We do this by ensuring that the sum of all possible state probabilities sums to one for each customer modem, adhering to the rules of probability theory. The parsimony cost of an internal node will no longer be the number of children disagreeing with the majority, but rather the sum of disbelief (among all the children) in the most probable event state as believed by the children. This means that if every child believes fully in a single (but not necessarily the same) state, the cost will be equivalent to that of the discrete case. Additionally, internal nodes inherit the averaged probabilities for each state from their children, meaning that the degree of belief in an event state will also be inherited and thereby not lost as in the discrete case. The pseudocode for the modified continuous parsimony score is given in Algorithm 3.2. An example of the calculation of the parsimony score for a tree using this probability-based algorithm is given in the supplementary material 3.C

3.6.3 Neighborhood functions

To generate a set of negative samples $\mathcal{G}^n$, we will need a set of neighborhood functions that can permute some tree G into a negative sample G_1^n through a series of neighborhood moves. Figure 3.6.3 illustrates the three types of neighborhood functions used throughout this paper. These neighborhood functions were originally introduced in [119].

Algorithm 3.2 Continuous Modified Parsimony Score for tree G with modem data $\mathbf{z}$ for a single time point

procedure CMPARSIMONY($G = (V = \{M, N\}, E)$, $\mathbf{Z}^* \in \{\mathbf{z} \in [0,1]^{|\Omega|} \mid \|\mathbf{z}\|_1 = 1\}^{|M|}$, $\omega_{root} \in \Omega$)
  $\mathbf{Z}^* \leftarrow \left[\mathbf{Z}^*,\ \text{NULL}^{|\Omega| \times |N|}\right]$ ▷ *Expand* $\mathbf{z}$ *with placeholder for internal nodes*
  $\bar{N} \leftarrow$ POSTORDERTRAVERSAL($G[N]$) ▷ *"Bottom-up" sorting of internal nodes*
  for $v \in \bar{N}$ **do**
    $D_v \leftarrow$ set of child nodes of v
    for $\omega \in \Omega$ **do** ▷ *For each character in the set of states,* Ω
      $\phi_\omega \leftarrow \sum \mathbf{z}^*_\omega(D_v)$ ▷ *Sum of probabilities of character* ω *among children*
      $z_{\omega,v} \leftarrow \frac{\phi_\omega}{|D_v|}$ ▷ *Probability of character* ω *in* v *is average of children's*
    if v is root **then**
      $cost_v \leftarrow |D_v| - \phi_{\omega=\omega_{root}}$
    else
      $cost_v \leftarrow |D_v| - \max_{\omega \in \Omega}(\phi_\omega)$ ▷ *Sum of children's disbelief in most probable* $\omega \in \Omega$
  return $\sum_{v \in N} cost_v$

Random swap inserts a new random edge (v, u) and removes the old parent edge of node u, i.e., edge (w, u) . *Subtree shuffle* selects a random sub-tree $\hat{G}(v)$ with k nodes rooted in node v, and shuffles all the edges in $\hat{G}(v)$ at random. *Parent-child swap* swaps the relationship between two node u and v, where v is the parent of u, such that u becomes the parent of v instead.

3.7 Experiments

In order to accurately assess the performance of our methodology, we have chosen to conduct the following set of experiments, designed to show:

- The legitimacy of our proposed modified version of the parsimony algorithm, i.e. Conjecture 1 (see subsection 3.7.3 and subsection 3.8.1 for experiment details and results, respectively).
- The performance of our approach for small trees, where it is feasible to consider all possible tree topologies during both training and validation. Including the impact of event sparsity/density in the data and the influence of multi-faults, as well as the performance under what is found to be the ideal data conditions (subsection 3.7.4 and subsection 3.8.2).
- The performance of our approach on bigger trees where only a subset of topologies can be considered for computational reasons, as well as the effect of different sampling strategies for generating the considered subset of topologies (subsection 3.7.5 and subsection 3.8.3).
- The performance of a single generalized encoder's ability to encode data for a wide range of tree sizes (subsection 3.7.6 and subsection 3.8.4).

Because this work intends to assess the feasibility of our contrastive approach,

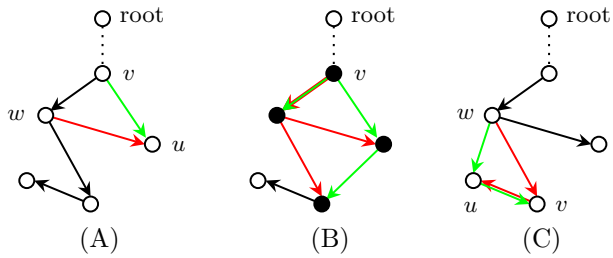


Figure 3.6.3: Examples of the three neighborhood functions used to generate negative samples, G_A^n, G_B^n, and G_C^n, of a topology G. (A) *Random swap*. (B) *Subtree shuffle*. (C) *Parent-child swap*. The green edges are new edges introduced by each neighborhood function, the red edges are edges removed by each neighborhood function. Figure taken from [119].

we will keep both the simulated data and the encoder model simple. For all experiments, we simulate independent training and testing sets each containing a set of instances, where an instance constitutes one true tree topology, a number of customer modems, and modem time series simulated based on the true tree.

We train our model on the training sets and assess the performance on the test sets. Inspired by Franceschi et al. in [53], but originally proposed by van den Oord et al. in [159], our encoder model will use a sequence of one-dimensional convolutional layers with exponentially increasing dilation to handle the time series. Since the topology reconstruction problem is not imminent, our model is not chosen to be causal - oppositely to the work of van den Oord et al.. and Franceschi et al. This means that our model includes data in time both prior and posterior to the prediction point under consideration. After a sequence-to-sequence encoding of the input time series, our model applies an identical fully connected layer to every time point to summarize the output of the convolutional channels into a single vector of output values for each time point. These outputs will act as an event's encoded probabilities after being sent through a sigmoid activation function. Recall that the parsimony algorithm treats every time point independently; hence multiple event vectors can be concatenated to form the input for the parsimony algorithm. An illustration of the architecture is shown in Figure 3.7.1. In the experiments conducted in this study, we opt for a value of 3 for H, a kernel size of 5 for each convolution, 16 channels for each convolutional layer, and eight neurons for the final fully connected layer.

It should also be mentioned that, in the case of binary events, there are two close-to-identical local minima in the encoder parameter space: the one that correctly identifies the presence or absence of events (encoded to be 1s and 0s in the output, respectively), and the one where the two are interchanged. This

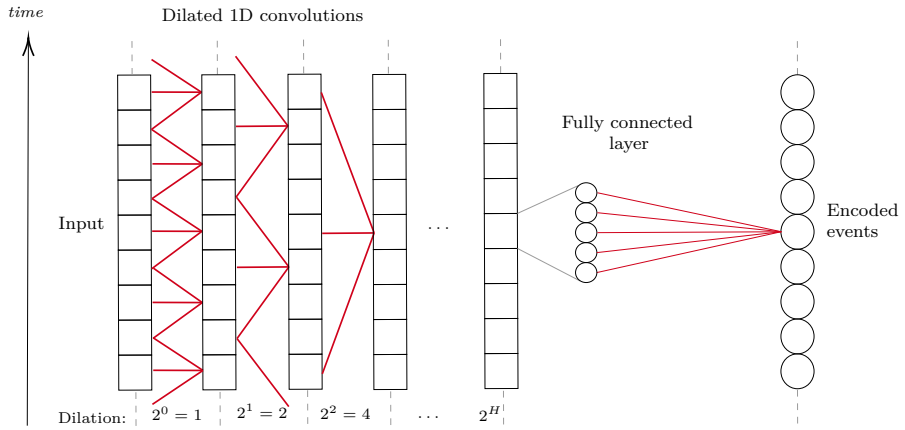


Figure 3.7.1: Illustration of the architecture used in the experiments. The use of dilated convolutions is inspired by Franceschi et al. in [53] and van den Oord et al. in [159], but differs in that a fully connected layer is applied to the output at every time point, retaining the time component in the output. H is the number of convolutional layers (hence also the coarsest dilation) in the architecture.

is not a problem in the traditional parsimony formulation, however, using our alternative formulation of the parsimony score given in (3.3), an interpretation of the output of the encoder is introduced. Since the root-level base signal is assumed to be all 0s, this slightly affects the value of the parsimony score. We get around this by simply checking after training (but still using the training data) which of the two cases provides the lowest modified parsimony score and thereby choose whether or not to flip all outputs of the encoder in the validation stage.

3.7.1 Data simulation

Since the goal of this paper is to assess the ability of the proposed methodology to find inherited events in time series data, we will perform the assessment in a controlled fully simulated setting. Thus, we simulate both the considered tree topologies as well as the continuous time series data. For each observation, we first simulate a true network topology with a predetermined number of internal nodes, N. We sample a number of child modems uniformly in the range from $5 \cdot (|N| - 1)$ to $9 \cdot (|N| - 1)$, both values included, and divide them among the internal nodes, except for the root, making sure that none of these nodes has less than two modems connected to it.

For small trees with less than seven internal nodes, we sample the true topology uniformly from the set of all possible topologies given a tree of that size. For bigger trees, we sample the true topology using a random sampling scheme discussed in the following section.

Lastly, we select the set of negative topologies. For small trees, this is the set of all possible topologies, while for bigger trees, the negative set of topologies will be sampled using one of the schemes discussed in the following.

3.7.1.1 Topology sampling

We define two different sampling schemes, *Smart* and *Random*, for topologies of a given number of internal nodes, $|N|$. The *random sampling* scheme is used for sampling both true topologies and topologies in the negative topology set, whereas, *smart sampling* is only used for sampling topologies in the set of negative topologies.

Random sampling We begin with the set of internal nodes, N, but without any edges connections between them (each of the customer leaf nodes in M has a preexisting parent in N, as described in section 3.5). One node in N is initially marked as the *root*. We then alternate between two ways of adding an edge, (u, v), to the graph until all internal nodes, excluding the root, have a parent, and consequently also a path to the root. In both approaches, we first sample a node $v \in N \setminus root$, that does not yet have a parent when considering the edges in E. The parent node, u, of v is sampled in one of two different ways:

- Sample u from the set of nodes in N for which the addition of the edge (u, v) would not create a cycle in
- Sample u from the set of descendants of the root node or the root node itself, i.e. nodes in N that already have a path to the root node, when considering the edges in E.

Smart sampling A set of smart samples is generated based on a true topology, G^p, each through a series of *neighborhood moves* as described in subsection 3.6.3. For each move, one of the three neighborhood functions is selected with a 45%, 10%, and, 45% probability, respectively. Each smart sample consists of $k \in [1, 3N]$ neighborhood moves chosen with exponentially increasing distance between the number of moves, however, sampled with uniform probabilities. Topologies sampled more than once are discarded, resulting in a naturally skewed distribution of the number of moves[3].

3.7.1.2 Time series simulation

For the time series data, we simulate perfectly distinguishable events by splitting the sequence of time points into several chunks of a predetermined size, τ, each with a start time, t_i, and an end time, $t_{i+\tau}$. For each chunk, we first simulate the presence or absence of an event in the network by sampling from a

[3]The probability of permuting to the same sample multiple times decreases as the number of moves increases.

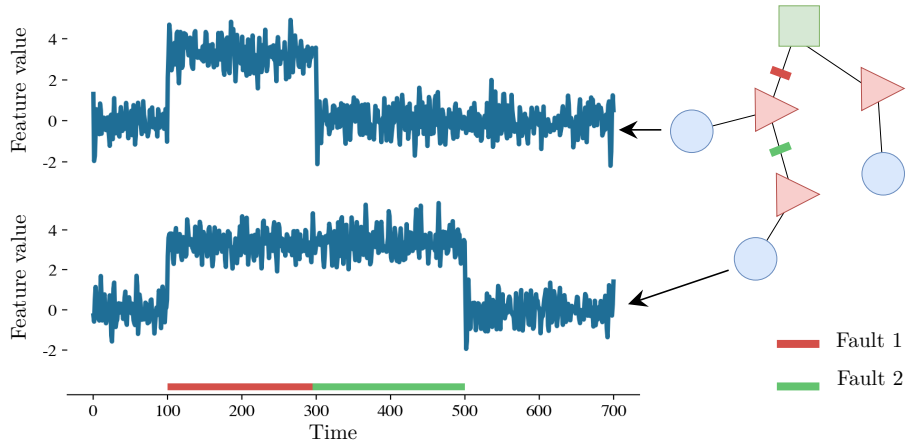


Figure 3.7.2: An example of part of a data instance with a tree of size $N = 4$ shown on the right (only three out of 29 modems are shown). A snapshot of the time series data is shown for two of the modems in which two faults occur on two different edges. These faults cause the time series for the modems to be affected in two different ways, as dictated by the tree topology. The base level signal has a mean of roughly zero.

Bernoulli distribution with a 50% event occurrence probability. In other words, we determine the set of affected time points, T^*. If an event occurs in a given time chunk, we then sample a random edge uniformly from the set of internal edges in the network, and from this edge, we determine the set of modems, M_t^*, affected by the event. Inspired by De Ryck et al. [40], Chang et al. [22], Kawahara & Sugiyama [72], Liu et al.[80], and originally proposed by Yamanishi & Takeuchi [171], we use the AR(2) process [81] as a base signal for our simulated time series data for a modem, m:

$$z_{m,t} = 0.6z_{m,t-1} - 0.5z_{m,t-2} + \epsilon_{m,t}, \qquad \text{where} \qquad \epsilon_t \sim \mathcal{N}(\mu_{m,t}, 0.5^2) \quad (3.8)$$

We choose $\mu_{m,t} = 3$ if modem $m \in M_t^*$, i.e. is affected by a fault at time t. Otherwise, we set $\mu_{m,t} = 0$; i.e. if modem $m \notin M_t^*$, which means that it is *not* affected by a fault at time t. An example of the simulated data is shown in Figure 3.7.2.

3.7.2 Evaluation metrics

To properly assess and compare the performance of our methodology in different settings, we define two accuracy metrics to be used across all of our experiments. One metric relates to our ultimate goal of accurately inferring the true topology of a network, and the other to the end goal of training an encoder that properly learns to identify the true underlying simulated events.

3.7.2.1 Topology Accuracy

We define the *topology accuracy* of an encoder to be the ratio of observations for which the *true topology* (i.e. the topology used to generate the modem time series) leads to the lowest parsimony score out of *all topologies* in a test set $\mathcal{G} = \{G^p\} \cup \mathcal{G}^n$. This means that for a set of observations, $s = 1 \dots S$, where each observation consists of a discrete-valued matrix, $\mathbf{Z}_s$, and a set of topologies, $\mathcal{G}_s$, whereof one is the true topology, G_s^p, the topology accuracy is given by:

$$\begin{aligned} \text{topology accuracy} = & \frac{\sum_{s=1}^{S} \mathbb{1}\left(\mathbf{Z}_s, G_s^p, \mathcal{G}_s^n\right)}{S} \qquad \text{where} \\ & \mathbb{1}(\mathbf{Z}, G^p, \mathcal{G}^n) = \begin{cases} 1 & \text{if} \quad \hat{\mathcal{P}}(\mathbf{Z}, G^p) < \hat{\mathcal{P}}(\mathbf{Z}, g), \quad \forall g \in \mathcal{G}^n \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.9}$$

If $\mathcal{G}$ contains all possible topologies for a given tree size we call it the *true topology accuracy*. This metric, however, quickly becomes cumbersome to calculate as the size of a tree grows, since the number of possible topologies increases exponentially with respect to the tree size as given by Cayley's formula [20]. For this reason, we also define the *estimated topology accuracy*, to be the ratio of observations for which the true topology leads to the lowest parsimony score considering $\mathcal{G}^n$ to be some (sampled) subset of all possible topologies for a given tree size.

3.7.2.2 Event Accuracy

We define the *event accuracy* of an encoder as the mean accuracy of the encoder over a set of observations. This refers to the mean ratio of time points for which the encoder correctly identifies the presence or absence of an event, relative to the true underlying data simulation. This means that for a set of observations, $s = 1, \dots, S$, where each observation consists of a discrete-valued matrix, $\mathbf{Z}_s$, of encoded events and a matrix, $\mathbf{Z}_s^*$, of true underlying simulated events, the event accuracy is given by:

$$\text{event accuracy} = \frac{\sum_{s=1}^{S} \sum_{m=1}^{|M|} d_H\left(\mathbf{z}_{s_m}, \mathbf{z}_{s_m}^*\right)}{S \cdot |M|} \tag{3.10}$$

where M is the set of distinct modems and d_H is the Hamming distance given in (3.2) between two vectors, in this case, the row vectors of $\mathbf{Z}_s$ and $\mathbf{Z}_s^*$, both of length $|B|$. This metric can also be used on a single observation (i.e. a single network) to assess how well the encoder extracts the true underlying discrete events from the continuous time series data.

3.7.3 Uniqueness conjecture experiments

The validity of Conjecture 1 is challenged through a series of rigorous experiments designed to find a counterexample to the claim. The experiments constitute of two parts;

1. *Full* testing for trees with three to six internal nodes.
   It is computationally viable to compute and compare all possible topologies and thus prove that Conjecture 1 holds true up to and including trees of size six.
2. *Partial* sampled testing for selected trees with more than six internal nodes.
   A subset of all possible topologies for trees of size seven to ten as well as fifteen, will be tested against a sampled sub-set of topologies.

We use a tree size of $|N| = 3$ as an example to elaborate on part 1 of the experiment. There exist three unique topologies for a tree of size three. One of these topologies is picked to represent the true topology G^p, and based on G^p the modem data is simulated. As described in Condition 1, we simulate the occurrence of exactly one independent event per edge per time point, i.e. a total of $|E|$. Since G^p in the case of $|N| = 3$ contains two internal edges, the modem data strings also contain two data points. Considering a binary alphabet of discrete states, the root data string is chosen as a string of zero bits, without loss of generality. Then, we let each data point correspond to an event occurring on each of the two edges and flip the bit for the modems affected by each event.

Using the Modified Parsimony Score, as computed by Algorithm 3.1, the parsimony score is calculated for each of the possible topologies using the same simulated data. For Conjecture 1 to hold true, G^p must yield a unique best score out of the three possible topologies. Repeating this process three times, each time letting a new topology represent G^p, proves that Conjecture 1 holds true for all trees of size three.

Similarly, we use a tree size of $|N| = 10$ as an example to elaborate on part 2 of the experiment. A tree of size ten has 100 million possible topology configurations. Consequently, we choose to only let 1000 randomly chosen topologies represent the true topology G^p and simulate modem data for each of them in the same way as described in Part 1.

For each G^p we now *sample* a sub-set of 10,000 topologies, $\mathcal{G}^n$, for which we compute the modified parsimony scores and compare them to the score of G^p. For $\mathcal{G}^n$ to best represent the total set of 100 million topologies, we obtain half of the samples in $\mathcal{G}^n$ by *Smart Sampling* and the other half by *Random Sampling* (see section 3.7.1.1 Topology sampling). This ensures that $\mathcal{G}^n$ includes topologies that are both similar and dissimilar to the true topology G^p.

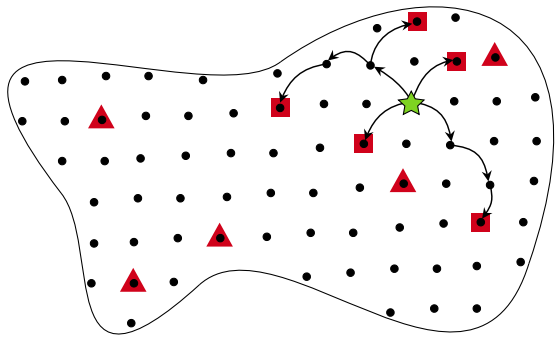


Figure 3.7.3: An illustration of *Random* and *Smart* sampling. The abstract shape illustrates the entire topology space for a network of size $|N|$. Each black dot inside the space represents a unique topology. The green star represents the true topology G^p and the selected set of samples $\mathcal{G}^n$ are marked with red, where triangles correspond to random samples and squares to smart samples. The arrows illustrate the neighborhood moves performed to reach each smart sample.

3.7.4 Training the encoder on the full set of topologies

We evaluate the performance of our proposed methodology when the model takes all possible topologies for a given tree size into account. As previously mentioned, we assume that the last-line amplifier/splitter is known, hence only the topology of the internal nodes will be inferred. In this setting, we test the performance of our approach for trees of varying sizes, however, keeping them small makes it feasible to train and test over multiple repetitions. Simultaneously, we assess the robustness of our methodology as data conditions become sub-optimal. We presume based on Conjecture 1, that the true topology, G^p, leads to the unique lowest parsimony score if the data contains at least one event on every edge in the network and that there is at most one event at a given time point.

For each data set, we simulate 1,000 observations of which half will be used as a test set. In all cases, an observation is given by, firstly; a true topology, G^p, sampled at random, uniformly from the set of possible topologies for a tree of a chosen size. Secondly; a set of negative samples, $\mathcal{G}^n$, is chosen to be the set of all possible topologies excluding G^p. Lastly; an array of time series data for the modems of the network, simulated specifically according to each specific experiment explained in the following.

3.7.4.1 Number of events in the time series data

We simulate ten different data sets for both trees of size four, five, and six, respectively, i.e. a total of 30 data sets. Each of the ten data sets will correspond

to an *average* number of η events per internal edge in E. We fix the chunk size to $\tau = 30$ and simulate time series of length $|T| = 2\tau\eta \cdot |E| = 60\eta \cdot |E|$ for each average number of events, η, in the set $\{1, 2, \ldots, 10\}$. These choices are made to accurately investigate dependencies while balancing running time and good grounds for learning. We do, however, not perform experiments using all ηs for bigger trees due to the infeasibility of computation when trees, thus also their time series, increase in size. We multiply by two to account for the fact that events only occur in a given chunk with a probability of 50% as described in subsection 3.7.1 Data simulation.

3.7.4.2 Introducing multi-faults

We simulate 11 different data sets for both trees of sizes four and five, i.e. a total of 22 data sets. Each of the 11 data sets corresponds to *a proportion of events* in the time series, ρ, for which two faults occur simultaneously on two different internal edges instead of one. We investigate values of ρ in the set $\{0\%, 10\%, 20\%, \ldots, 100\%\}$. For each data set, we fix the chunk length to $\tau = 50$ and the average number of events to $\eta = 10$ to make sure that the encoder has good conditions for learning. These numbers are chosen generously to make the influence of multifaults transparent. This results in a time series length of $|T| = 2\tau\eta \cdot |N| = 1,000 \cdot |N|$.

When simulating an event (after checking if an event is present) in a given chunk as described in subsection 3.7.1 Data simulation, we sample from a Bernoulli distribution with probability ρ of a multi-fault. If a multi-fault should be present, we sample two different internal edges, instead of just one, uniformly from the set of internal edges and modify the time series of the modems beneath both faults accordingly.

3.7.4.3 Performance under ideal data conditions

We consider the case with ideal data conditions (i.e. only singular faults, long time series consisting of long-lasting faults) for our approach and simulate time series under these conditions for trees of size four and five. Already for trees of size six, it is too demanding to train the encoder considering ideal data conditions. This is due to the loss function (3.7), in this case, requiring $6^4 = 1,296$ (see subsection 3.3.1) parsimony evaluations for each forward pass in the training loop, because all possible topologies are considered in the set of negative samples. We fix the chunk length to $\tau = 50$ to be able to computationally evaluate gradients in the process of learning the encoder. We simulate time series with $\eta = 10$ events per internal edge in the tree on average, to allow for accurate modeling. This results in a time series of length $20\tau|E|$. For these results, we exclude the results where the encoder ends up in the pitfall of learning the zero-encoder because it is easy to check if that is the case. This is still a possibility even though we are using a contrastive approach to avoid it.

3.7.5 The influence of sampling

As the size of a network increases beyond six, it is no longer computationally viable to train the encoder on a set of a negative sample $\mathcal{G}^n$ which includes all possible topologies. Realistically, $\mathcal{G}^n$ must therefore be a sub-set representing the full *topology space*. We aim to compare the effectiveness of two different sampling techniques as well as investigate how many samples are needed to represent the full topology space. The effectiveness will be evaluated using the topology accuracy given by (3.9).

The two sampling techniques considered are *Random sampling* and *Smart sampling* which were introduced in subsubsection 3.7.1.1. For larger trees, it quickly becomes impractical to sample a fixed proportion of the full topology space, to represent the negative samples $\mathcal{G}^n$. Instead, we investigate the relationship between the number of negative samples and the size of the trees, denoted as $|N|$. Specifically, we explore the size of $\mathcal{G}^n$ within the range from 1 to $3|N|$, both values included, as this range is computationally feasible for trees of up to approximately size 400.

The experiment includes trees of size six, seven, nine, and fifteen, where size six has the unique benefit that both the *true* and *estimated* topology accuracy (3.9) can be computed and consequently compared. Additionally, for size six it is feasible to also *train* the encoder letting $\mathcal{G}^n$ include all possible topologies, making this a baseline for evaluating the effect of letting $\mathcal{G}^n$ be a sampled subset of all possible topologies. This is, however, computationally very heavy. For trees of size five and smaller, the size of the topology space is at most 125, and consequently, $3|N|$ constitutes a significant fraction of the full space, making such results non-generalizable. This is why we only consider trees of size six or bigger.

To avoid sampling bias, each experiment for each tree size will be conducted on 500 different true topologies G^p. We compare the two sampling methods using the same set of true simulated topologies and time series, hence only the set of negative samples are simulated differently for the two methods. Both sampling methods are tested on the same test using the estimated topology accuracy defined in (3.9). A test set constitutes an equal mix of both random and smart sampling as described in subsubsection 3.7.1.1. An illustration of the full experiment setup is shown in Figure 3.7.4. We set the length of the data chunks to $\tau = 30$ and the length of the time series to $|T| = 12\tau|N|$ for a mean number of events per edge of six.

3.7.6 A generalized encoder for multi-sized trees

Since HFC networks rarely have the same size, i.e. the same number of amplifiers/splitters, we check the ability of our approach to learn an encoder when the test samples stem from observations with trees of different sizes. We simulate

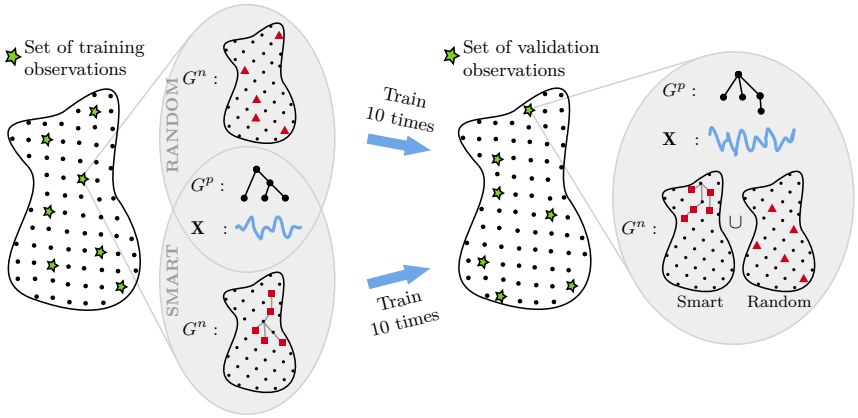


Figure 3.7.4: The data generation scheme for the sampling influence experiment. Both sampling strategies are trained on the same set of true topologies and time series, hence only the set of negative samples vary between the two datasets. Both strategies are validated based on the estimated topology accuracy using the same validation set. Each observation in this set consists of a mixture of both random and smart samples, as described in subsection 3.6.3.

500 training samples each consisting of a randomly sampled true topology, G^p, with a number of internal nodes, $|N|$, in the range 4 to 20, both values included. The set of negative topologies, $\mathcal{G}^n$, will be based on $3|N|$ sampled topologies, sampled using the smart sampling scheme, as described in subsubsection 3.7.1.1. For the test set, we simulate 500 observations based on the same method, however, letting the set of negative topologies be based on 500 samples, sampled using a mixture between smart and random sampling, to ensure that the full topology space is well represented.

In this case, we simulate fewer customer modems, simulating the number in the range from $2 \cdot (|N| - 1)$ to $5 \cdot (|N| - 1)$, ensuring that all internal nodes, except for the root, each have at least two adjacent modems. We fix the chunk length to $\tau = 30$ and simulate on average $\eta = 10$ events per internal edges, such that the modem time series have lengths $|T| = 2 \cdot 10 \cdot 30 \cdot |N| = 600|N|$.

3.8 Results

3.8.1 Uniqueness conjecture experiments

Table 3.8.1 shows the results of the tests designed to disprove Conjecture 1. For tree sizes from three and up to and including seven, a full test—where all possible topologies have been tested against all possible topologies—was conducted. In all cases, the true topology leads to a unique lowest modified parsimony score.

Table 3.8.1: Results of tests designed to find counterexamples to Conjecture 1. All tests failed to disprove the conjecture.
Here, $|N|$ is the tree size, $\Psi(|N|)$ the number of possible topologies of size $|N|$ (Cayley's formula), *# Tested* the number of unique topologies tested as being true (i.e. used to simulate data from) followed by the corresponding fraction of $\Psi(|N|)$ given in parentheses, *# Tested against* the number of unique topologies each true topology is tested against, *Successful* the number of tests which successfully disproves Conjecture 1, and *Proved* marks which test fully prove Conjecture 1 by exhaustive search.

$\|N\|$	$\Psi(\|N\|)$	# Tested (%)	# Tested against	Successful	Proved
3	3	3 (100.00)	*All*	0	✓
4	16	16 (100.00)	*All*	0	✓
5	125	125 (100.00)	*All*	0	✓
6	1,296	1,296 (100.00)	*All*	0	✓
7	16,807	2,004 (11.92)	*All*	0	
8	262,144	1,002 (0.38)	10,000	0	
9	4,782,969	882 ($1.84 \cdot 10^{-2}$)	10,000	0	
10	100,000,000	786 ($7.86 \cdot 10^{-4}$)	10,000	0	
15	1,946,195,068,359,375	528 ($2.71 \cdot 10^{-11}$)	10,000	0	

For reference, case $|N| = 7$ took around 12 hours to compute using six parallel threads on an *Intel Xeon E5-2620 v4 (2.1 GHz)* CPU.

For the remaining cases in Table 3.8.1, only a subset of all topologies was tested against a subset of topologies. However, all tests still show that the true topology leads to a uniquely best parsimony score.

Given the mix of a *random* and a *smart* sampling technique, the subset tests still provide compelling evidence for Conjecture 1 for larger trees. Random sampling ensures a wide range of differing topologies is considered while smart sampling ensures topologies similar to the true topology are also considered. Thus, the full *topology space* is well represented in the sampled sub-set tests.

3.8.2 Training the encoder on the full set of topologies

Using the full set of possible topologies as the negative samples in our contrastive approach gives the model the maximum amount of information possible and, hence serves as a baseline for the best possible performance.

Figure 3.8.1 shows the true topology and event accuracy, respectively, of our contrastive approach as a function of the mean number of events per internal edge simulated in the time series. Figure 3.8.1a shows a clear trend with the performance increasing as the information (number of events) in the simulated

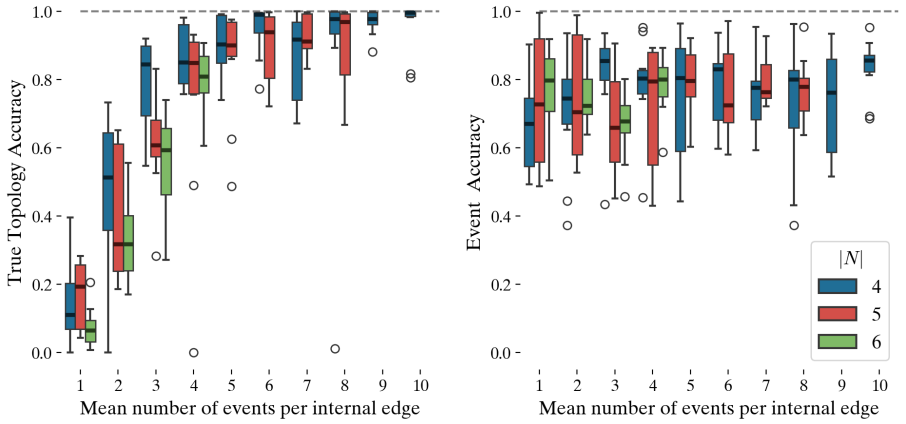


Figure 3.8.1: Performance of the proposed approach as a function of the number of events generated in the simulated data. Reported using a grouped boxplot of ten repeated experiments for each number of events per internal node and tree size combination. Each repetition is randomly initialized with a new encoder. Results are truncated at the point where we did not find it feasible to train due to memory computation constraints (length of time series, thus also computational demand increases with the number of events).

time series increases. It also shows that the topology accuracy converges at 100%, affirming the potential of our contrastive approach in learning an encoder that can extract inherited events from time series.

Interestingly, from Figure 3.8.1b it seems that the trend of the event accuracy as a function of the mean number of events is rather constant. Moreover, the increased topology accuracy cannot be directly correlated to a simultaneous increase in the event accuracy. In other words; it seems that the encoder does not need to extract all the events in the time series to learn an informative encoder with the ultimate goal of being able to do accurate topology reconstruction.

Additionally, Figure 3.8.1a also shows that, even though the approach works well most of the time, there is still a possibility of the encoder learning the trivial case; the zero-encoder, resulting in a topology accuracy of 0 (recall that the topology accuracy is based on the true topology being uniquely best). This is, however, not believed to be problematic, since it is easy to check if this is the case after training.

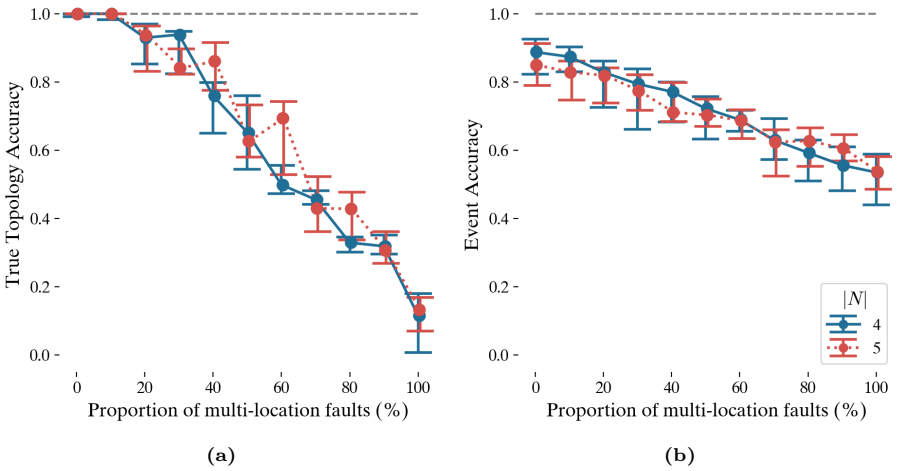


Figure 3.8.2: Performance of the proposed approach as a function of an increasing amount of multi-faults, i.e. faults that happen on multiple edges at the same time. Results are reported as the median and 50% confidence interval (CI) of ten experiments, each with a new and randomly initialized encoder.

3.8.2.1 Introducing multi-faults

We test the robustness of our approach with respect to the challenge posed by *multi-faults* as introduced in subsection 3.5.4. Figure 3.8.2 shows the performance of our approach as a function of the proportion of simulated events that are multi-faults, i.e. indistinguishable events happening simultaneously at two locations in the topology, affecting the descendant modems of both locations. From Figure 3.8.2a it seems that our approach is somewhat robust to multi-faults up to a level of approximately 20-30%. The figure also suggests a similar robustness for trees of sizes four and five, respectively, due to the curves showing close-to-identical trends. In Figure 3.8.2b a seemingly negative trend is visible that ends at approximately 50% (corresponding to a random encoder) when all the generated faults are multi-faults. It should be mentioned that the number of simulated events in this case is rather high (ten events per internal node), to allow for higher granularity in the tested proportions.

3.8.2.2 Performance under ideal data conditions

Table 3.8.2 shows the results for trees of sizes four and five in an ideal setting without multi-faults and with time series containing the information necessary to learn an optimal encoder, based the experiment in Figure 3.8.1. These results have been reported as the mean and standard deviation of 20 repetitions,

Table 3.8.2: Results for trees of size four and five when all possible topologies are used as negative samples in the contrastive approach. The results are reported as the mean and standard deviation of 20 repetitions where a new random initialization of the encoder is performed. The length of the chunks, τ, is chosen to be 50, to allow for accurate modeling. Because it is easy to check, zero-encoders have been excluded from these results which means that experiments have been repeated until 20 encoders have been learned that did not end up in this pitfall.

$\|N\|$	Topology accuracy	Event accuracy
4	98.27 % ($\pm$3.01)	86.16 % ($\pm$9.82)
5	97.92 % ($\pm$3.36)	82.88 % ($\pm$10.13)

however, removing samples in which the zero-encoder was learned.

We observe that the accuracy is very high in both cases. This suggests that the learned encoder successfully generated representations that, in almost all instances, resulted in the topology from which the data was sampled, having the lowest parsimony score among all possible topologies.

Similar conclusions as the ones made for Figure 3.8.1 can be drawn; i.e. the event accuracy stagnates at a sub-optimal level of approximately 85%, which is inconsequential to the topology accuracy which is near perfect at a level of approximately 98%.

3.8.3 The influence of sampling

The topology accuracy for trees of sizes 6, 7, 9, and 15 is depicted in Figure 3.8.3. Here, the negative set of topologies $\mathcal{G}^n$, utilized during training the encoder, is a sampled subset. Assuming the results from Table 3.8.2 can be generalized to larger trees, we know that it is theoretically possible to train an encoder that delivers a topology accuracy of around 98%. Since this result was obtained when training and testing on the full set of possible topologies, it serves as a baseline when evaluating the influence of sampling.

The figure furthermore compares the topology accuracy when sampling $\mathcal{G}^n$ during the training stage using either a fully *Smart* or fully *Random* sampling scheme. Moreover, Figure 3.8.3 shows how the number of samples used when training the encoder, i.e. the size of $\mathcal{G}^n$, affects the topology accuracy.

It is immediately clear that Smart sampling outperforms Random sampling in all four experiments in Figure 3.8.3. Moreover, for $|N| = 6$ there is a positive correlation between the true topology accuracy and the number of samples in $\mathcal{G}^n$. This trend is not apparent for the remaining values of $|N|$ considering the estimated topology accuracy. However, it is important to note that these results contain an increasing level of uncertainty. For $|N| = 6$, the full topology space

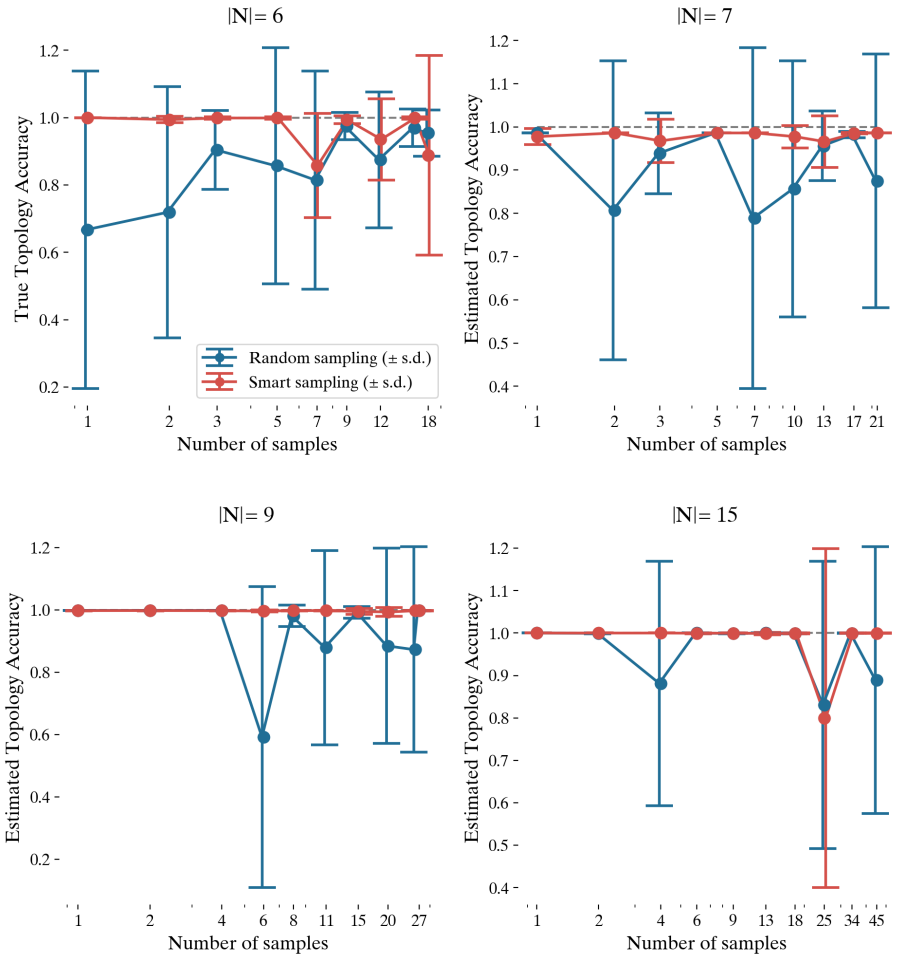


Figure 3.8.3: The above results illustrate how the *topology accuracy* of trees of size 6, 7, 9, and 15 is influenced by each of the two sampling strategies considered as well as the sample size. Each data point represents an average of ten repetitions of training and validating the encoder on 500 observations, respectively, where each observation is based on a unique combination of a true topology, G^p, and modem data. See Figure 3.7.4 for an illustration of the data generation scheme. The whiskers on each data point denote the standard deviation. The x-axis (log scale) shows the number of samples in $\mathcal{G}^n$ in the range from 1 to $3|N|$, used to train the model in each observation. The trained encoders are validated on a validation set containing 500 observations each consisting of both random and smart samples. Zero-encoders have been excluded from the results.
For tree size $|N| = 6$, the true topology accuracy is computed, whereas the remainder of the results are based on the estimated topology accuracy. In all experiments, employing the smart sampling scheme for sampling the negative set of topologies used in training outperforms the random sampling scheme.

is small enough for us to compute the *true* topology accuracy, i.e. where the test set contains all possible topologies of size six. For the remaining experiments, it is only possible to compute the *estimated* topology accuracy, i.e. only comparing the true topology to a subset of possible topologies during testing. Since the size of the test set compared to the full set of topologies dramatically decreases for increasing tree sizes, consequently, so does the validity of the estimated topology accuracy.

Despite this evaluation uncertainty, we have attempted to sample a fair test set of topologies using both Smart and Random samples. In Figure 3.8.3, however, the topology accuracy appears to improve for increasing tree sizes. This suggests that the estimated topology accuracy is optimistic for bigger trees, as the full variation of the exponentially increasing topology space might not be captured by the fixed-sized sampled test set. However, an alternative explanation is that the topology accuracy improves for bigger trees due to an increase in the event accuracy, which we found to improve by approximately 10% from the $|N| = 6$ case to the $|N| = 15$ case.

Overall, the smart sampling scheme during training results in consistently high topology accuracies concerning both the size of the tree and the number of training samples, $\mathcal{G}^n$. Moreover, this is achievable with considerably fewer computational resources.

3.8.4 A generalized encoder for multi-sized trees

Lastly, Table 3.8.3 shows the results of training and testing a generalized encoder on sets of trees of varying sizes up to 20. Although one would expect this learning task to be more challenging, the results in the table suggest otherwise. Here, zero-encoders have been excluded from the results enabling a fair comparison with results in Table 3.8.2. However, it must be noted that in Table 3.8.2 the true topology accuracy is reported, whereas in Table 3.8.3 the topology accuracy is partially estimated.

As compared to the results of training and testing an encoder under the best possible conditions (see Table 3.8.2), the event accuracy improves from approximately 85% to more than 90% in the case of the generalized encoder. As the event accuracy is never an estimation and thus a reliable measure, this suggests that when the encoder is trained on only a single tree size, it is prone to overfitting.

Despite the increased event accuracy, Table 3.8.3 shows that the generalized encoder obtains only a slightly improved topology accuracy of approximately 99% as compared to the 98% observed in Table 3.8.2. Here, it is important to mention that this is only an *estimated* topology accuracy, since for larger trees, the test set only includes a sampled subset of all possible topologies.

Table 3.8.3: Results for generalized encoder. Results, especially event accuracy are better than for when an encoder is only trained on one tree size.

Topology accuracy	Event accuracy
99.89 % (±0.23)	93.44 % (± 5.87)

3.9 Discussion

Overall, we find the results of this preliminary study of our proposed methodology promising. We demonstrated that a topology prediction accuracy of around 99% and an event accuracy of up to 93% are attainable for simplified time series data.

Additionally, we designed a set of experiments that: studied the robustness of the methodology with respect to various data assumptions; investigated performance under ideal data conditions; considered the impact of sampling; and investigated how generalizing the methodology impacted performance. We found the performance of the encoder to be best when trained on a set of trees of various sizes as opposed to one single size (see Table 3.8.2 and Table 3.8.3). When investigating the influence of multi-faults, we found that as long as multi-faults do not constitute more than 20% of all data events, the topology accuracy does not worsen by more than 10% (see Figure 3.8.2). Furthermore, we found that the minimum number of events needed to achieve the best possible topology accuracy depended on the size of the tree. As events were simulated to occur on random edges, this result intuitively makes sense, as for larger trees it becomes progressively less likely to sample each unique edge at least once due to the increasing total number of edges.

As trees grow in size, the proposed methodology is only able to consider a set of training and test topologies constituting a diminishing fraction of the complete topology space. We analyzed two sampling schemes and found the *smart* sampling scheme based on neighborhood moves to be best performing during training (see Figure 3.8.3). In testing, we let the topology test set be a mix of both *smart* and *random* samples. We computed the *true* topology accuracy for trees of size 6, where the full topology space contains 1,296 topologies, as a function of using various sizes of sampled training sets. The *estimated* topology accuracy of larger trees was computed as a function of the same training scheme, allowing for a credible comparison between the *true* and *estimated* topology accuracy as well as the two sampling schemes (see Figure 3.7.4).

We experimentally proved Conjecture 1 to be true for tree sizes up to and including six and used the same training and testing sampling scheme as above to challenge the validity of the conjecture for larger trees (see Table 3.8.1).

Moreover, the 99% topology accuracy obtained in other experiments, meaning a unique best topology was almost always found, adds to the plausibility of Conjecture 1 although it remains to be formally proved.

For this preliminary study, data events were simulated to be simple and distinguishable and to occur on a single edge at a time (except for when investigating the influence of multi-faults). While real modem data is not this ideal, the encoder should still be able to identify the positions within the data where the behavior is suitable with respect to the contrastive loss function. Useful data behavior does not have to be actual events that modify the signal in a simple way as visualized in Figure 3.2.2, but can also be an underlying correlation structure, e.g. the same general behavior but scaled differently, or more complex events such as local instantaneous spikes or cyclic behavior. The identification of this type of data behavior could also prove useful for purposes other than topology reconstruction, such as anomaly detection or root cause analysis in all sorts of data sources not limited to HFC data.

As modems collect various data metrics at every time point, some metrics may not be informative on their own but could be valuable collectively. Our proposed encoder can be readily adapted to handle multiple time series inputs on different scales, producing multiple latent variables. This collection of latent variables could form an informative ensemble. Additionally, the parsimony algorithm is not restricted to binary outcomes, as earlier mentioned, allowing for the inclusion of multiple states to represent varying degrees of errors. This does, however, impose the challenge of relating the different states to one another when determining mutation costs, meaning that strong domain knowledge of the errors would be needed to properly do so.

Due to the proposed loss function including a minimization problem building on a bottom-up (poster-order) traversal of the internal nodes of the network, the evaluation of this is computationally expensive and not readily improvable by e.g. the use of GPUs. This is, however, not a problem for the ISPs (Internet Service Providers) as they would only need to train a generalized model once to infer all the missing topology. After this point, the loss function does not need to be evaluated again.

For the proposed methodology to give good results for ISPs, one of the main challenges would be to obtain a descriptive and extensive dataset that could be used for benchmarking. In training the model, the true topology of a given last-mile network would need to be known and as mentioned earlier, many ISPs often only partially know their own topologies. In the case of TDC NET; while the whole topology is only known for a small set of last-mile networks, it is far from all networks in which nothing is known. This means that ISPs could use the known part (and corresponding modem time series) of a network to train the encoder model, and then use the trained model to infer the missing part of the topology.

One of our main assumptions in our work is that every internal node needs to have at least one modem child. While this assumption might not hold for real networks, ISPs could train an encoder using only the parts of a set of real networks in which this assumption holds. Additionally, deployment of the full approach would mean using the encoded events to search for the optimal topology including geographic distances using e.g. the algorithm proposed by Pisinger & Sørensen in [119]. This means that the assumptions do not necessarily need to hold true for good deployment results in real life.

Our proposed loss function aims to simultaneously minimize the parsimony score of the true topology while maximizing the dissimilarities between distinct topologies. However, it may be oversimplified to solely focus on maximizing the distances between any two pairs, as similar topologies could yield small maximum distances. Therefore, incorporating a more sophisticated loss function that also considers the degree of similarity among all pairwise sets of topologies could prove beneficial.

Although we obtain promising results, there is still the possibility of the encoder ending up in the pitfall of learning a zero-encoder, which, in some cases, is sensitive to weight initialization. Even though it is easy to check if the encoder ends up in this pitfall, it could also be addressed by doing unsupervised pretraining of the weights of the network, making this less likely.

3.10 Conclusion

We propose a novel contrastive approach to performing discrete encoding of continuous time series while confirming the feasibility of integrating an optimization algorithm into a deep learning loss function. We present an encoder, f, tailored for HFC networks, which takes a set of continuous time series as input and generates binary sequences, implicitly learning data events in order to solve the topology reconstruction problem. Our approach leverages a contrastive loss function, incorporating a modified algorithm for computing the *parsimony score*. The modified parsimony score is designed to enable unique optimal solutions and we introduce a conjecture that states this is possible for all trees. We prove the conjecture for trees with up to six internal nodes and provide compelling evidence for bigger trees. To address the non-differentiability of binary sequences during training in the deep learning context, we also propose a continuous variant of the new modified parsimony algorithm.

Our results are promising, achieving an estimated accuracy of 99.89 % in topology reconstruction and an accuracy of 93.44 % in data event identification for a generalized encoder considering trees of multiple sizes. Additionally, we propose a sampling scheme to ensure comprehensive coverage of the full topology space, providing a solid foundation for training an encoder and generating accuracy

estimates for larger trees.

As this work is only a preliminary pilot study, the natural next step will be to evaluate the methodology in a more realistic setting, potentially using real modem data. Nevertheless, our results still confirm the potential for infrastructure owners to infer missing topology using continuous time series from the customer modems on the network.

Although our proposed methodology was designed around a telecommunication framework, its applicability may extend to other fields. For instance that of phylogenetic inference in biology where despite genetic material being inherently discrete, the uncertainty and ambiguity in the data may justify adopting a continuous perspective. E.g. by considering the probability of each of the four nucleotides found in DNA occurring at each position in a genetic sequence.

Overall, our work highlights the advantage of integrating an optimization problem in a deep learning setting rather than using each of the methods individually. Despite the increased complexity introduced by this approach, its relevance remains significant, particularly for models requiring only a single training phase.

Acknowledgement

This project was supported by Innovation Fund Denmark, project 0224-00055B, GREENFORCE. David Pisinger was funded by ERC-2022-ADG, project 101093188 DECIDE.

Appendices

3.A Proof of Algorithm 3.1

Firstly, Algorithm 3.1 only modifies how the parsimony score is calculated in the top layer of a tree, namely when calculating the cost of the root node with respect to its immediate children. By letting the children of the root node be represented by the set D_{root}, then every subtree rooted in a node $u \in D_{root}$, must be optimal. This follows directly from the proof of the original parsimony algorithm since algorithm 3.1 calculates the cost of these subtrees in exactly the same manner.

Now, it remains to be proven that it is optimal to pay the cost of the root-node assumption in the top layer of the tree, and that it cannot be cheaper to move the top-layer data mutations further down the tree.

This part of the proof can be split up into two cases. For every node $u \in D_{root}$ we have that either:

- Case 1: the optimal state set of node $u \in D_{root}$ includes the state of the root node.
- Case 2: the optimal state set of node $u \in D_{root}$ does *not* include the state of the root node.

It is straightforward to prove that Case 1 is optimal. Since the state of the root is included in the optimal state set of node v, then there is no data mutation on edge ($root$, v), and the cost associated with this edge is then zero. Since mutation costs must be positive, a cost of zero in the top layer cannot be improved. Moreover, we have already argued that the subtree rooted in node v is optimal, and consequently also not improvable. In conclusion, in Case 1, algorithm 3.1 will compute the best possible parsimony score.

In Case 2, the state of the root node is *not* included in the state set of node v. As a result, algorithm 3.1 pays a cost of one for letting a data mutation occur on edge ($root$, v). The total cost of this part of the tree, according to algorithm 3.1, is then; $1 + \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$. The question is then, if this can be done cheaper by letting the mutation occur somewhere else in the subtree $G(v)$. i.e. obtain a cost $< 1 + \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$. This could potentially be achieved in two ways:

i) cost is $\leq 0 + \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$, i.e. the cost of edge ($root$, v) disappears while the cost of the subtree rooted in v stays the same or becomes less.
ii) cost is $\leq 1 + \Big(\mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v)) - 1\Big)$, i.e. the cost of edge ($root$, v) remains one, while the cost of the subtree rooted in v is decreased by at least one.

In i), if the cost of edge $(root, v)$ must disappear, then it means that the state of the root node must also be the state of node v. But given that we are in Case

2, then the root node state is *not* in the *optimal* state set of v. Consequently, when the state of v is forced to equal the root node state, then the total cost of the subtree rooted in v must be suboptimal, i.e. $> \mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$, since at least one additional mutation must occur in the subtree rooted in v.

In ii), the cost of edge $(root, v)$ remains one—the state of v can be chosen freely—but consequently the subtree rooted in v must be less than $\mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$. Since $\mathcal{P}(\hat{\mathbf{Z}}, \hat{G}(v))$ is already optimal, this is impossible. This concludes the proof.

■

3.B Original Parsimony Algorithm

The formal method of calculating the parsimony score for a single time point using the original algorithm is given in Algorithm 3.3.

Algorithm 3.3 Parsimony Score for tree G with modem data, $\mathbf{z}$, for a *single time point*

procedure PARSIMONY($G = (V = \{M, N\},\ E)$, $\mathbf{z} \in \Omega^{|M|}$)
  $\mathbf{z} \leftarrow \left[\mathbf{z},\ \text{NULL}^{|N|}\right]$ ▷ *Expand* $\mathbf{z}$ *with placeholder for internal nodes*
  $\bar{N} \leftarrow$ POSTORDERTRAVERSAL($G[N]$) ▷ *"Bottom-up" sorting of internal nodes*
  for $v \in \bar{N}$ **do**
    $D_v \leftarrow$ set of child nodes of v
    for $\omega \in \Omega$ **do** ▷ *For each character in the set of states,* Ω
      $\phi_\omega \leftarrow$ frequency of $\omega \in \mathbf{z}(D_v)$
    $\mathbf{z}_v \leftarrow \arg\max_{\omega \in \Omega} (\phi_\omega)$ ▷ *Character state(s) of* v *becomes the most frequent*
    $cost_v \leftarrow |D_v| - \max_{\omega \in \Omega} (\phi_\omega)$ ▷ *Number of children disagreeing with majority*
  return $\sum_{v \in N} cost_v$

3.C Calculation example using the continuous algorithm

An example of the calculation of the continuous parsimony score of a tree is provided in Figure 3.C.1.

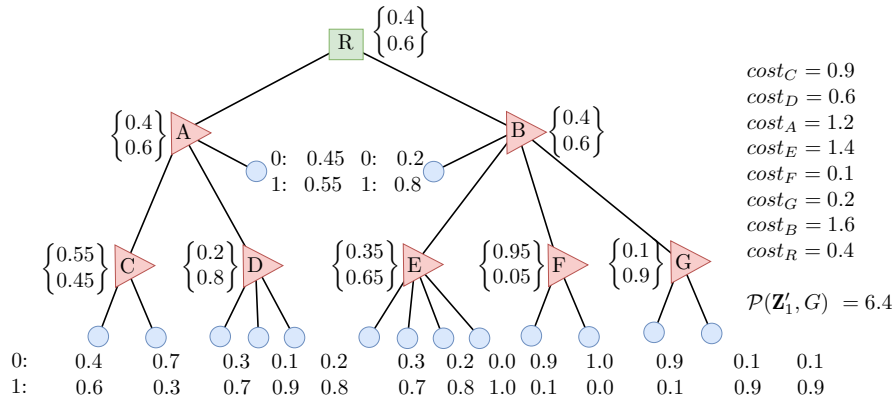


Figure 3.C.1: An example of how to compute the optimal parsimony score for a specific tree G using algorithm 3.2. Here, a single time point in the input data sequences $\mathbf{Z}'$ is considered. This data consists of probabilities of each of the two states in the alphabet for each customer (blue nodes). The red nodes are splitters/amplifiers, and the green node is the CMC. The computed probabilities of each state for the internal nodes are listed in curly brackets. The optimal parsimony score for the example tree is 6.4. The cost in each node corresponds to the sum of the probabilities for the least probable state as believed by the children. For instance, in node C, the most probable state is 0, hence the cost becomes the sum of probabilities of state 1 as observed in the children; $0.6 + 0.3 = 0.9$.

Chapter 4

Paper III

Fault Estimation in Telecommunication Networks: A Maximum Parsimony Approach

---

Status: Working paper

Authors: Siv Sørensen[a] and David Pisinger[a]
[a] DTU Management, Akademivej, Building 358, DK 2800 Kgs. Lyngby

Keywords: Maximum Parsimony, fault detection, likelihood estimation

This paper builds on some of the ideas and methods introduced in Paper I and serves as a **preliminary** proof-of-concept for an approach to obtaining the data input assumed to be available in Paper IV.

The **aim** of this paper is to demonstrate that—in addition to using the *maximum parsimony* principle to reconstruct *last-mile* telecommunication networks as shown in Paper I and Paper II—the same principle can be applied to infer the *likelihood* of a component in a last-mile network being faulty.

4.1 Introduction and motivation

Recall that the maximum parsimony optimality criterion—in its original context—corresponds to reconstructing an evolutionary tree by minimizing the number of evolutionary changes, or mutations, required to explain the observed differences between a set of species. The equivalent problem in a last-mile telecommunication network is reconstructing the network cabling by minimizing the number of data mutations required to explain the observed differences in downstream modem data. In an ideal world, the data quality of all modems within the same network would be identical, but this is rarely the case. Minor variations in modem data are often caused by imperfections in the network equipment or the way it is installed, while significant variations usually suggest underlying issues or faults that disrupt a customer's connection. A maximum parsimony approach relies on such data variations, known as *events*, to be effective. In Paper II, we presented preliminary evidence supporting the extraction of events from continuous time-series modem data is possible, leading us to hypothesize that:

The maximum parsimony principle can be used to localize or detect faults in last-mile telecommunication networks.

This hypothesis could benefit several processes, including, but not limited to:

Fault resolution When a customer reports a connection issue, a technician is dispatched to resolve it. The fault causing the customer issue could be located either on the customer's premises or anywhere along the direct cable connection, including any intermediate components, leading back to the Coaxial/Cable Media Converter (CMC) node connecting the last-mile network to the backbone grid. Typically, the fault-finding process is done through trial and error and involves visiting multiple points along this path until the fault is identified. This process consumes an unnecessary amount of resources with respect to both driving and occupying a technician's time.

Predictive maintenance By identifying potentially faulty components in the last-mile network before a customer complaint arises, service providers can perform proactive maintenance, which could lead to improved customer

satisfaction. Predictive maintenance tasks are generally less urgent than actual fault tickets, and this flexibility can be exploited in the planning phase. For example, predictive maintenance can be scheduled when a technician is already in the vicinity, reducing driving time and in the long run, potentially, all together mitigating the resource consumption associated with future fault tickets.

The main **contributions** of this paper are:

- We present a recursive algorithm for label reconstruction under the *Maximum Parsimony* criterion.
- We propose a fast, novel cross-disciplinary framework for fault prediction and localization in telecommunication networks based on modem data.
- We demonstrate that the framework accurately identifies the correct fault location within its top two predictions in 90% of the time, across a diverse set of instances with varying difficulty.
- We highlight the framework's practical potential when combined with routing strategies, leading to increased customer satisfaction, time savings, and reduced travel for both fault localization and predictive maintenance tasks.
- Our findings opens up new research opportunities for integrating fault localization uncertainty into routing problems, with significant potential for advancements in areas like the Technician Routing and Scheduling Problem (TRSP).

The paper is **organized** as follows: Section 4.2 highlights the additional information that the *maximum parsimony* principle can offer apart from the parsimony score of a tree. Next, section 4.3 presents the proposed framework. In section 4.4, the setup of the computational experiments is explained and preliminary results are reported. Section 4.5 concludes the paper.

4.2 Underlying information in the parsimony score

Assuming the topology of a last-mile network is known, the maximum parsimony principle can be used for more than simply calculating the minimum (best) *parsimony score* (2.1) of the network, i.e. the least amount of data mutations needed to explain the observed differences in downstream modem data. Based on the parsimony score, all possible *label assignments* that lead to the minimum parsimony score, can be reconstructed. This essential corresponds to reconstructing the character (data) states in the internal nodes that minimize the total mutations. These label assignments are also referred to as *minimum mutation fits* [59]. Figure 4.2.1 illustrates all minimum mutation fits for an example topology considering a single data point.

By analyzing the internal label reconstructions and identifying where mutations

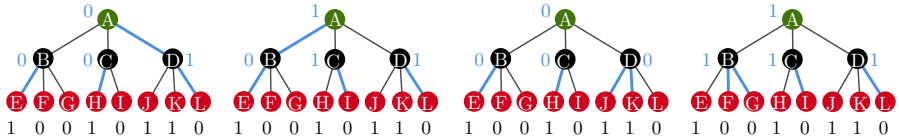


Figure 4.2.1: An illustration of all possible *label assignments* (given in blue) that lead to the minimum parsimony score for the network and modem data pictured. Each label assignment corresponds to a best possible parsimony score of four, meaning four data mutations occur (marked with blue edges). The red leaf nodes illustrate customer nodes, the black nodes cable splitter/amplification nodes, and the green node the root node (CMC).

(or changes) occur on the tree's edges, one can infer which parts of the network are most likely to have experienced a fault based on the observed modem data. As an example, say that customer node L in Figure 4.2.1 reports an issue with their connection. As the modem data corresponds to data metrics about the downstream data quality, the fault must occur somewhere on the path from L to the root node A, given by the path: {(A,D)(D,L)}, where (i, j) is an edge. In three of the possible label reconstructions, the network data mutates on edge (D,L), and in one reconstruction, it mutates on edge (A,D). This suggests that edge (D,L) is most likely the source of the error experienced by node L.

In the above, we cannot distinguish between whether a fault occurs on a cable (i, j) or node j because both components are arranged in a sequence where one follows directly after the other. Therefore, we will, for simplicity, refer to the likelihood of a node j being faulty in the remainder of this chapter, but implicitly, this includes the possibility that the cable (i, j) leading to node j may also be the source of the issue.

4.3 Framework

Let $T = (V, E)$ be a last-mile telecommunication network with V nodes and E edges. Moreover, let $z_k(T)$, as in (2.3), define the minimal parsimony score for a network T at a single point in time k, based on the observed modem data at that instance in time. We can then define $\mathcal{L}_k(T)$ as the set of all possible label assignments that yield a parsimony score of $z_k(T)$ for the network T considering modem data at time k. Additionally, let c^l_{vk} denote the label of node v at time k for a given label assignment $l \in \mathcal{L}_k(T)$, where a label can take on different values (*states*) in the *alphabet* of states; $\Omega = \{0, 1\}$. Here, the label in a leaf node (customer) is always equal to the observed modem data state. All of the following results can easily be generalized to a more fine-grained set of states Ω.

Based on the above, we define the *score* of a node $v \in V$ at time k as:

$$\delta_{vk} = \sum_{l \in \mathcal{L}_k(T)} 1_{\left(c^l_{vk} \neq c^l_{uk}\right)} \tag{4.1}$$

where u is the parent node of v, and $1_{(\text{condition})}$ is the indicator function, which returns 1 if the label of the parent node u of v is different from the label of v, and 0 otherwise. Essentially, the score δ_{vk} counts how many times a data mutation (label change) occurred in the edge above node v across all label assignments in $\mathcal{L}_k(T)$.

Since interference from a fault can be sporadic, it is reasonable to assume that considering modem data over a longer period of time rather than at a single instance could be advantageous. For example, if a customer reports an issue at time k_2, modem data from time k_1 to k_2 could be considered, where k_1 is a prior point in time. We therefore define an additional *score* function that sums all scores over a period of time:

$$\delta_{vk_1k_2} = \sum_{k=k_1}^{k_2} \delta_{vk} \tag{4.2}$$

This time-aggregated score helps identify intermittent faults that might not be detectable at a single instance in time, improving the robustness of the fault detection framework.

Moreover, given the uncertainty associated with the data recorded by customer modems—due to background noise, differences in equipment, etc.—as well as the uncertainty associated with encoding the time-series data (see Paper II), we propose to extend the set of label assignments, $\mathcal{L}_k(T)$. Instead of only including the label assignments that strictly correspond to the minimal parsimony score, $z_k(T)$, we include all label assignments that yield a parsimony score less than or equal to the minimal score $z_k(T)$ plus a slack value. Formally, we define:

$$\mathcal{L}^{\text{slack}}_k(T) = \{l \mid \text{parsimony score of } l \leq \lceil z_k(T) \times (1 + \mathit{slack}) \rceil\} \tag{4.3}$$

where technically, the *slack* can be any value from zero to infinity. Realistically, a good slack value would mostly be in the interval $[0, 1]$, such as 0.1, allowing label assignments that deviate up to 10% from the best parsimony score.

Additionally, a specific label assignment $l \in \mathcal{L}_k(T)$ (or $\mathcal{L}^{\text{slack}}_k(T)$) is defined as:

$$l = \{c^l_{vk},\, v \in V\} \tag{4.4}$$

where c^l_{vk} is a distinct label for node $v \in V$ and where the collection of labels $\{c^l_{vk},\, v \in T\}$ together form a minimum mutation fit in $\mathcal{L}_k(T)$.

However, by introducing slack, some less likely label assignments become feasible. Figure 4.3.1 illustrates two label assignments for the same network T,

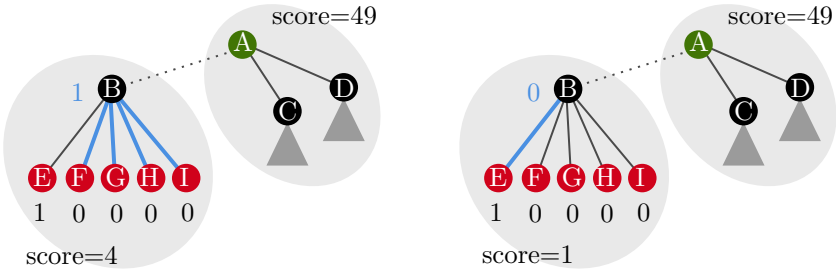


Figure 4.3.1: An illustrative example of why label assignments $l \in \mathcal{L}_k^{\text{slack}}(T)$ can be unlikely even when considering low slack values. The label assignment pictured to the right $l \in \mathcal{L}_k(T)$ is optimal and forms a minimum mutation fit for T with a parsimony score of $z_k(T) = 50$. The label assignment pictured to the left has a parsimony score of 53, which is only 6% more than $z_k(T)$. However, all of the slack has been accumulated in the sub-tree rooted in node B, making the label assignment in this part of the network unlikely.

where the label assignment on the right corresponds to a minimal parsimony score of $z_k(T) = 50$. As an example, say we allow a 10% slack when computing all label assignments in $\mathcal{L}_k^{\text{slack}}(T)$, then the label assignment on the left becomes valid with a parsimony score of 53. However, all the allowed slack has been concentrated in the sub-tree rooted at node B, making the label assignment in this part of the network unlikely. To avoid such label assignments, we introduce a constraint for each label c_{vk}^l that limits how *sub-optimal* a label of a specific node can be.

Let $f_v(\omega)$ denote the *frequency* of the label $\omega \in \Omega$ among the children of node v, where the set of children is denoted by D_v. Moreover, let F_v represent the *frequency* of the most common label(s) in D_v. Then:

$$f_v(c_{vk}^l) \geq F_v - S \qquad \forall \text{labels } (c_{vk}^l) \text{ in } l, \forall v \in V, l \in \mathcal{L}_k^{\text{slack}}(T) \tag{4.5}$$

This constraint ensures that the frequency of the chosen label c_{vk}^l is not significantly lower than the frequency of the most common label among the child nodes of v, controlled by the slack parameter S, which should be selected based on the average number of descendant nodes a node in T has.

Using node B in Figure 4.3.1 to illustrate the above constraint, label 0 is the most frequent label in the set of children D_B, with a frequency of $f_B(0) = F_B = 4$. As an example, choosing $S = 2$ would mean that only labels with a frequency of two $(4 - 2)$ or higher would be allowed. Since the frequency of label 1 is $f_B(1) = 1$, the left label assignment pictured in Figure 4.3.1 would not be allowed in the set $\mathcal{L}_k^{\text{slack}}(T)$.

4.3.1 Generating the set of possible label assignment

The only challenging part of the above framework is computing the set of label assignments $\mathcal{L}_k^{\text{slack}}(T)$. Once this set is known, the score for each node—as given by (4.1)—can readily be calculated.

In the paper by Hartigan [59] that first describes and proves how the minimal parsimony score for a general tree can be computed, a method for obtaining all the equivalent label assignments is also very briefly described. Below, in Section 4.3.1.1, we present and elaborate on this method. Afterward, in Section 4.3.1.2, we explain how the method can be adapted such that the label assignment set $\mathcal{L}_k^{\text{slack}}(T)$ can be computed while adhering to the constraint of only allowing individual node labels that are at most S worse than the majority label as given by (4.5).

4.3.1.1 Hartigan's label assignment method

Hartigan [59] proposes a two-pass method. Since the label of each leaf node is predetermined by the data input, the method only considers the set of internal nodes $V^I \subset V$.

first-pass: The first-pass considers each of the nodes $v \in V^I$ one-by-one, in a bottom-up manner, e.g., by a post-order traversal. For each node, an upper label set, $VU(v)$, and a lower label set, $VL(v)$, is computed.
$VU(v)$ is the set of labels that minimize the number of mutations when considering the set of direct children of v, D_v. The $VU(v)$ set consists of the most frequent labels observed in the child nodes, as these are the labels that would minimize the number of changes required. Specifically, $VU(v) = \{\omega \in \Omega \mid f_v(\omega) = F_v\}$. Similarly, $VL(v) = \{\omega \in \Omega \mid f_v(\omega) = F_v - 1\}$, which is the set of the second most frequent labels.

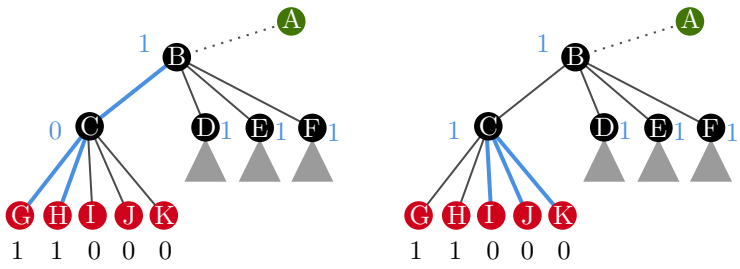


Figure 4.3.2: An illustrative example of why the lower level set, VL, is relevant when computing all label assignments $l \in \mathcal{L}_k(T)$. For node C, $VU(C) = \{0\}$ with $F_v = 3$, which means that $VL(C) = \{1\}$. The two label assignments pictured illustrate that because the optimal label in node B (parent of C) is $c^l_{Bk} = 1$, then there exists a minimum mutation fit for both $c^l_{Ck} = 0$ and $c^l_{Ck} = 1$.

Without going into too much detail with regards to the underlying proof of the correctness of Hartigan's approach, Figure 4.3.2 gives some intuition as to why considering the upper label sets $VU(v)$ is not enough when reconstructing all possible label assignments $\mathcal{L}_k(T)$.

The first-pass procedure corresponds to Algorithm 4.1.

second-pass: Hartigan [59] does not directly describe a procedure for the second-pass of his approach but instead proposes a theorem that makes it possible to find all minimum mutation trees. Let ω_u be the label of node u in an optimal label assignment $l \in \mathcal{L}_k(T)$, and let $V(v, \omega_u)$ be the possible values taken by node v—given v is a child of u—when $c^l_{uk} = \omega_u$. Then:

$$V(v,\omega_u) = \begin{cases} \{\omega_u\} & \text{if } \omega_u \in VU(v) \\ \{\omega_u\} \cup VU(B) & \text{if } \omega_u \in VL(v) \\ VU(v) & \text{if } \omega_u \notin VU(v),\ \omega_u \notin VL(v) \end{cases} \tag{4.6}$$

We propose a recursive-like approach with backtracking for the second-pass as given by Algorithm 4.2 where (4.6) corresponds to lines 14-20. The structure and the correctness of the algorithm are presented below.

Algorithm 4.1 first-pass: Computing the sets $VU(v)$ and $VL(v)$

▷ **Initialization:**
1: **for** each $v \in V^I$ **do**
2: Initialize $VU(v)$ and $VL(v)$ as empty sets
3: **for** each leaf node $v \in V^L$ (customer) **do**
4: Initialize $VU(v) = \{c_{vk}\}$ (observed data) and $VL(v) = \emptyset$
▷ **Post-order traversal to compute** $VU(v)$ **and** $VL(v)$**:**
5: **for** each $v \in V^I$ in post-order traversal of T **do**
6: **for** each $\omega \in \Omega$ **do**
7: Compute $f_v(\omega)$, the frequency of label ω in $VU(w)$ for each $w \in D_v$
8: Let $F_v = \max_{\omega \in \Omega} f_v(\omega)$ ▷ *Maximum frequency of any label*
9: $VU(v) \leftarrow \{\omega \in \Omega \mid f_v(\omega) = F_v\}$ ▷ *Most frequent labels*
10: $VL(v) \leftarrow \{\omega \in \Omega \mid f_v(\omega) = F_v - 1\}$ ▷ *Second most frequent labels*
▷ **Output:** Sets $VU(v)$ and $VL(v)$ for all internal nodes $v \in V^I$

The algorithm presented for finding all minimum mutation fits (Algorithm 4.2) proceeds in a systematic and recursive manner, ensuring that every valid label assignment in $V(v, \omega_u)$ from (4.6) for the internal nodes is explored. Its correctness is rooted in the following key aspects:

Initialization: Several data structures are introduced. `LA`, initially set to `NaN`, stores all labels in a valid label assignment $l \in \mathcal{L}_k(T)$; the `done` array is used to track whether all possible labels in $V(v, \omega_u)$ for each node have been considered; the `next` array tracks the index of the next label in $V(v, \omega_u)$ to be assigned to each node.

Traversal: A pre-order traversal of the internal nodes is performed, starting from the root. For each node, the set of possible label assignments $V(v, \omega_u)$ is defined based on the label of its parent node, ω_u, and the precomputed sets $VU(v)$ and $VL(v)$. The algorithm systematically assigns each label from $V(v, \omega_u)$ to the current node v, guaranteeing that all possible combinations of labels are explored.

Backtracking: Once all nodes are labeled, the algorithm saves the current label assignment, `LA`, and backtracks to the most recent unfinished node. Backtracking ensures that the algorithm continues to explore all remaining label combinations, resetting the state of nodes that need to be revisited along the way.

Correctness: The correctness of the algorithm follows directly from the completeness of the traversal and backtracking mechanisms. By iterating over all possible labels for each internal node and ensuring that no valid label assignment is skipped, the algorithm fully explores the solution space and finds all minimum mutation fits.

4.3.1.2 Modified label assignment method with slack

We modify the two-pass procedure from Section 4.3.1.1 such that all labels in $\mathcal{L}_k^{\text{slack}}(T)$ are found while adhering to constraint (4.5).

The first step is modifying the first-pass procedure in Algorithm 4.1. The change here is very simple; before, the lower label set $VL(v)$ was computed in line 13 as the set $\{\omega \in \Omega \mid f_v(\omega) = F_v - 1\}$; now it is computed as: $VL(v) = \{\omega \in \Omega \mid f_v(\omega) = F_v - S\}$, which ensures constraint (4.5). This is the only change to Algorithm 4.1.

The second step is modifying the second-pass procedure in Algorithm 4.2. Here, the required changes are more complicated than in the first-pass procedure. If Algorithm 4.2 is run using the new definition of $VL(v)$, there is no guarantee that all of the label assignments found correspond to a parsimony score of less than or equal to $\lceil z_k(T) \times (1 + slack) \rceil$ as required per definition (4.3). There are two apparent ways to go about this issue:

1. Before saving label assignment `LA` in line 28 of Algorithm 4.2, count the number of mutations in `LA` and only save the label assignment if the number of mutations does not exceed $\lceil z_k(T) \times (1 + slack) \rceil$.
   The advantage of this approach is that it is simple to understand and implement. However, the disadvantage is that for increasing values of slack, most of the computation time is spent on generating labels that will eventually be discarded.
2. Dynamically keep track of the parsimony score of label assignment `LA` while it is being constructed. As soon as the score of `LA` exceeds $\lceil z_k(T) \times (1 + slack) \rceil$, abort the current node search and backtrack to the last unfinished

Algorithm 4.2 second-pass: Finding All Minimum Mutation Fits, $\texttt{LA} \in \mathcal{L}_k(T)$

```
▷ Initialization: ◁
LA[v ∈ V^I] = NaN                          ▷ Label assignment for internal nodes
done[v ∈ V^I] = false                      ▷ True if all labels in V(v, ω_u) have been considered
next[v ∈ V^I] = 0                          ▷ Index of next label in V(v, ω_u) to consider
walkOrder ← preOrder(V^I)                  ▷ Pre-order traversal of V^I
pos ← 0                                    ▷ Current position in walkOrder
v ← walkOrder[pos]                         ▷ Start with root node

for ω ∈ VU(v) do
    LA[v] ← ω                              ▷ Assign label to root
    pos ← pos + 1
    v ← walkOrder[pos]
    while v ≠ root do
        u ← parent(v)                      ▷ Parent of node v
        ω_u ← LA[u]                        ▷ Label of parent node u
        if ω_u ∈ VU(v) then                ▷ Define the set V(v, ω_u)
            V = {ω_u}
        else if ω_u ∈ VL(v) then
            V = {ω_u} ∪ VU(v)
        else
            V = VU(v)
        LA[v] ← V[next[v]]                 ▷ Assign label from V(v, ω_u)
        if next[v] + 1 = |V| then
            done[v] ← true                 ▷ Mark node v as done
        else
            next[v] ← next[v] + 1          ▷ Point to next label in V(v, ω_u)
        if v = walkOrder[end] then         ▷ All nodes labeled
            save(LA)                       ▷ Save label assignment
            while done[v] is true do       ▷ Backtrack to last unfinished node
                done[v] ← false            ▷ Reset
                next[v] ← 0                ▷ Reset next label index
                pos ← pos − 1              ▷ Move back
                v ← walkOrder[pos]
        else
            pos ← pos + 1
            v ← walkOrder[pos]
```

node in the same manner as in lines 29-34 in Algorithm 4.2.
The advantage of this approach is that it avoids unnecessary explorations and is computationally efficient, but the disadvantage is that it is more complicated to implement as the parsimony score of `LA` changes both when adding or changing labels in `LA` as well as when regretting labels in `LA` during backtracking.

We choose to go with option 2. in the above due to the speed-up advantage. The modified version of Algorithm 4.2 is not included as pseudo-code because of its size and complexity.

4.4 Preliminary results

To assess the proposed methodology, a preliminary study is conducted using the networks and simulated modem data from Paper I. The network generation method and the modem simulation approach are both explained in Section 2.5 of Paper I; key network and data characteristics are also described. In this paper, we only consider the subset of instances from Paper I with up to and including 30% random background noise. This is because Paper I already concluded that background noise above 30% leads to poor solutions.

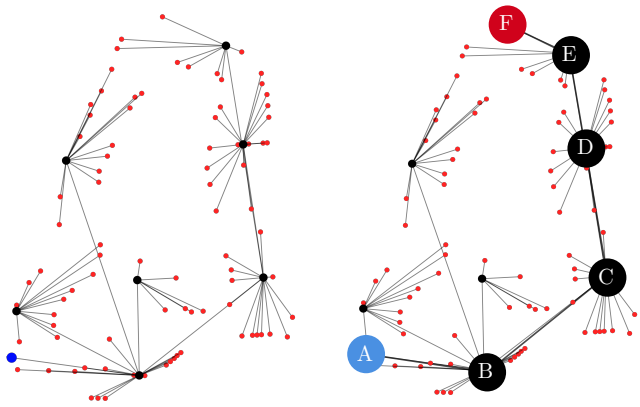
As discussed in Section 4.1, the proposed framework could be applied for both *fault resolution* tasks and *predictive maintenance* tasks. We report results focused on the *fault resolution* aspect only, as this is also the topic or Paper IV. Specifically, we consider the setting where a customer reports an issue and a technician must localize the underlying fault. When a customer reports an issue, the nodes on the path between the customer leaf node and the root node make up the set of potential fault locations as pictured in Figure 4.4.1. Below, we use the proposed framework to estimate a *fault likelihood* for each of these locations, with the goal of obtaining a high likelihood for the true fault location.

4.4.1 Fault simulation

Recall from Paper I that modem data is recorded every 15 minutes. Moreover, from a topology reconstruction standpoint, it is crucial to have *events* (data mutations) occur on every edge of the network. Therefore, for each data point in the modem data from Paper I, a random subset of network edges was chosen to 'behave abnormally', and the (binary) data point of each descendant customer was affected by flipping from a zero to a one bit.

On top of this rather noisy data, we simulate *persisting* faults that eventually result in a customer complaint that a technician will have to handle. The fault simulation process is described in the following:

We go through each of the 3200 data points in the modem data one by one and



(a) (b)

Figure 4.4.1: (a) Illustrates part of a real-life *last-mile* network. The blue node is the root node that connects the network to the back-bone grid. The red nodes are customers. The black nodes illustrate cable splitter, and the lines are cable connections. Parts of the network connected to the root node have been left out for simplicity.
(b) Illustrates the set of potential fault locations in the case that customer F reports an issue. The set of potential fault locations includes all components (nodes) on the path from the customer to the root node. The proposed framework sets out to estimate the likelihood of each node on the path being faulty in order to aid the process of localizing the fault affecting node F.

simulate the occurrence of a new fault with a 1 in 400 (0.25%) chance. Each fault persists until a customer reports an issue, at which point the fault is assumed to be resolved instantaneously. The duration between the first occurrence of a fault and up until a customer complains is sampled from a normal distribution with mean and standard deviation $(\mu, \sigma) = (3, 1.18)$ days and then rounded up to the nearest full day. From the first occurrence of a fault, at least two days have to pass before a new fault can occur. Due to the aforementioned fault duration, this means that faults can overlap in time. Figure 4.4.2 shows the output of the fault simulation process for one of the instances considered.

For every fault, three additional characteristics are decided: The fault type, the data mutation type, and the data interference level.

Fault type: We distinguish between *customer faults* and *network faults*, where the former fault type occurs at a customer's premise and only affects a single customer and the latter occurs out in the network and affects multiple customers.

Data mutation type: Decides *how* the modem data is affected in the set of customers below a failure point. For each fault, one of three cases is chosen at random:

(a) (b)

Figure 4.4.2: Output of the fault simulation process for instance F0 (see Table 2.1) with 225 customers. (a) Shows the seven faults that were generated as well as their duration and position within the 3200 data-points long modem data. (b) Shows the (normal) distribution of the fault duration of the seven faults.

- Case 1: if a data point k is affected, it is changed to a 1, regardless of what it was before.
- Case 2: if a data point k is affected, it is changed to a 0, regardless of what it was before.
- Case 3: for each affected data point, k, a coin-flip decides if the data point should be changed to a 0 or a 1—in all modems affected—regardless of what data was before.

Interference level: Specifies the likelihood that each data point k is affected during a fault occurrence. The interference level is selected from the set $[0.2, 0.4, 0.6, 0.8, 1.0]$, where, for example, a level of 0.4 means that each data point k has a 40% chance of being affected while the fault is active. The interference level is used to emulate sporadic fault behavior.

4.4.2 Framework setup

Ideally, the framework variables should be tuned and tailored to each general type of dataset on which it is applied. However, for this preliminary study, we have simply chosen the framework characteristics based on trial and error. The allowed *slack* on the parsimony score of a label (4.3) is set to 10%. Moreover, we allow node labels to have a frequency up to two less than the most frequent label, i.e., $S = 2$ in equation (4.5).

When a customer reports an issue, in real life, it is not known when exactly the issue started. Therefore, we have arbitrarily chosen that the framework

Inst.	# customers	Background noise level			
		0%	10%	20%	30%
A	100	135	162	383	1,597
B	125	122	132	361	1,366
C	150	120	147	495	3,267
D	175	119	148	316	1,531
E	200	114	183	828	18,688
F	225	118	143	473	4,599
G	250	121	171	1,217	47,959
H	275	114	175	769	11,578
J	300	115	191	1,306	57,003
K	325	119	129	381	7,109
L	350	106	286	2,672	198,074
M	375	113	206	646	11,048
N	400	110	381	6,001	714,272
P	450	115	194	2,407	762,123
Q	500	110	213	3,072	759,967

Table 4.4.1: The average number of label assignments, $|\mathcal{L}^{\text{slack}}(T = \text{Inst.})|$, found per fault for each instance across all 96 data points considered. The instances listed were first introduced in Section 2.5 of Paper I. For clarity, the number of customers in each instance is also given.

always considers the past 24 hours of data prior to a customer complaint. In the accumulated score function (4.2), this means that k_2 is set to the time of the complaint and k_1 is set to 24 hours earlier. Given that data is collected every fifteen minutes, this corresponds to 96 data points, meaning the proposed framework from Section 4.3.1.2 is run 96 times per fault.

4.4.3 Results

Table 4.4.1 lists the average number of label assignments, $|\mathcal{L}^{\text{slack}}(T = \text{Inst.})|$, found per fault for each instance, when considering the entire time frame of the 96 data points prior to a customer complaint. It is clear that the variations in background noise in the original modem data paired with the chosen slack of 10% and $S = 2$ yield very different results. For a background noise level of 0%, the chosen slack variables appear too conservative as only one label assignment is found for the majority of the 96 time points. Oppositely, for instances with 30% background noise, an excessive amount of label assignments are found, and tighter slack parameters might have been better.

Table 4.4.2 shows the performance of the proposed framework and scoring function (4.2). The performance is measured by comparing the score, $\delta_{vk_1k_2}$, for each of the components $v \in V$ that are on the path from the customer node to

	v_{fault} score rank:	Best	Second best	Other
# faults	0%	45 (65%)	22 (32%)	2 (3%)
	10%	34 (56%)	26 (43%)	1 (2%)
	20%	32 (51%)	24 (38%)	7 (11%)
	30%	31 (47%)	19 (29%)	16 (24%)
	All:	**142 (55%)**	**91 (35%)**	**26 (10%)**

Table 4.4.2: The performance of the fault location, v_{fault}, when its score, $\delta_{vk_1k_2}$, is compared to the score of all nodes in the set of *potential fault locations*. The table lists the number of faults where the score of the fault was either the highest, second highest, or something else. The first four rows show the performance across instances with 0%, 10%, 20%, and 30% background noise, respectively. The last row shows the performance when considering all instances.

	v_{fault} score rank:	Best	Second best	Other
# faults	Case 1: $\rightarrow 1$	68 (87%)	7 (9%)	3 (4%)
	Case 2: $\rightarrow 0$	29 (32%)	49 (53%)	14 (15%)
	Case 3: $\rightarrow 0$ or 1	45 (51%)	35 (39%)	9 (10%)

Table 4.4.3: The performance of the fault location, v_{fault}, when its score, $\delta_{vk_1k_2}$, is compared to the score of all nodes in the set of *potential fault locations*. The table lists the number of faults where the score of the fault was either highest, second highest, or something else. The table shows the performance across the three different cases of data mutation types, i.e. how a fault affects the modem data.

the root node, i.e. the scores of the set of *potential fault locations* (see example in Figure 4.4.1b). This set always contains at least two locations: the customer's own premise and the parent splitter node, as no customers are connected to the root node directly. Given the goal is that the fault location, v_{fault}, scores highest, Table 4.4.2 lists the number of times the score of v_{fault} was: best, second best, or something else. When considering all faults across all scenarios, the score of v_{fault} is either the highest or second highest in 90% of the cases.

The table also shows how the score of v_{fault} performs across different levels of background noise. The best performance is reached for the instances without background noise, and for increasing background noise, the number of times the fault location, v_{fault}, scores highest decreases while the number of times the score of v_{fault} ranks outside the top 2, increases.

Next, we consider the impact of each of the three data mutation types in Table 4.4.3. Given that the modem data, on average, contains a 25%/75%-split between 1-bit and 0-bit data points, respectively, before the faults were added, it is not surprising that case 1, where a fault causes the data to mutate to 1-bits, performs best. Case 2, where a fault causes the data to mutate to 0 bits,

v_{fault} score rank:		Best	Second best	Other
# faults	0.2	13 (25%)	28 (55%)	10 (20%)
	0.4	21 (50%)	15 (36%)	6 (14%)
	0.6	32 (52%)	23 (38%)	6 (10%)
	0.8	31 (67%)	14 (30%)	1 (2%)
	1.0	45 (76%)	11 (19%)	3 (5%)

Table 4.4.4: The performance of the fault location, v_{fault}, when its score, $\delta_{vk_1k_2}$, is compared to the score of all nodes in the set of *potential fault locations*. The table lists the number of faults where the score of the fault was either the highest, second highest, or something else. The table shows the performance across the five different data interference levels considered, i.e. the likelihood of each data point being affected during an active fault.

performs worst. Intuitively, it makes sense that it is harder to localize a fault that causes the data to mutate to what the majority of the modem data already consists of, making it difficult to distinguish between normal behavior and the effects of the fault.

Table 4.4.4 shows the impact on the performance based on the fault interference level. Unsurprisingly, the more data points a fault affects while the fault is active, the better the framework works.

Lastly, Table 4.4.5 shows how multi-faults impact the performance, i.e., when two faults overlap. The table suggests that there is seemingly no difference in performance. Due to the way multi-faults were simulated, there can be multiple reasons for that:

- Two simultaneously occurring faults might affect two separate subsets of modems in the network.
- The overlap period can be small or might not even be considered since the framework only looks at the last 96 data points of a fault duration.

In conclusion, the current fault simulation setup does not allow for a fair evaluation of the impact of multi-faults.

The above results analysis of the proposed framework shows that, under decent data conditions, the proposed scoring function (4.2) is a useful tool for localizing faults in telecommunication networks. However, it is not apparent how to translate the score of a network component into a *likelihood* for the component being the fault location. Figure 4.4.3 and 4.4.4 show two examples of scores across the set of resolution locations that vary both in magnitude and in their relative proximity to each other.

v_{fault} score rank:		Best	Second best	Other
# faults	With:	64 (56%)	41 (36%)	10 (9%)
	Without:	78 (54%)	50 (35%)	16 (11%)

Table 4.4.5: Multi faults The performance of the fault location, v_{fault}, when its score, $\delta_{vk_1k_2}$, is compared to the score of all nodes in the set of *potential fault locations*. The table lists the number of faults where the score of the fault was either highest, second highest, or something else. The table shows the performance between with and without multi-faults, i.e. two faults occurring simultaneously.

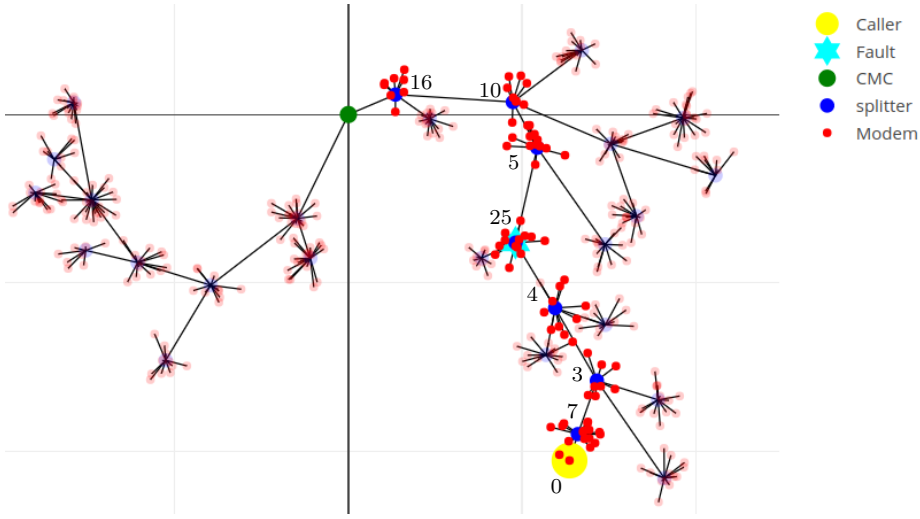


Figure 4.4.3: The score, $\delta_{vk_1k_2}$, for each of the nodes in the set of potential fault locations, i.e. the calling customer's premise and the set of splitter nodes connecting the customer with the root (CMC) node. The score is listed next to each node. Apart from the path between the calling customer and the root node, the rest of the network has been toned down for clarity. The above network is instance M0 with 375 customers, 0% background noise, case 3 data mutation type, and an interference level of 0.4.

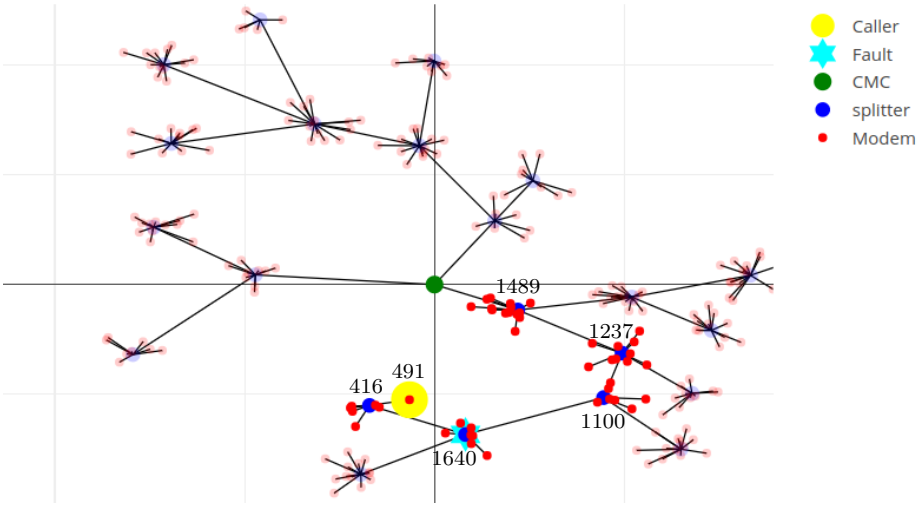


Figure 4.4.4: The score, $\delta_{vk_1k_2}$, for each of the nodes in the set of potential fault locations, i.e. the calling customer's premise and the set of splitter nodes connecting the customer with the root (CMC) node. The score is listed next to each node. Apart from the path between the calling customer and the root node, the rest of the network has been toned down for clarity. The above network is instance F3 with 255 customers, 30% background noise, case 3 data mutation type, and an interference level of 0.8.

4.5 Conclusion and next steps

Based on the preliminary study above, we can conclude that our hypothesis holds true and that the maximum parsimony principle can, in fact, be used to localize or detect faults in last-mile telecommunication networks.

We proposed a labeling framework based on the maximum parsimony principle, in which the most *likely* state of the network is reconstructed at each of the internal nodes. Based on these data state predictions, we accumulate data mutations at each component in the network and assume that where the most data mutations occur is the most likely fault location.

The problem of detecting faults in HFC networks is not a new one, and multiple studies of the problem in various forms exist [177, 60, 56]. State-of-the-art methods often employ a machine learning or unsupervised learning approach, while our framework offers a new and promising perspective on the problem. Additionally, our approach provides further insights, as reconstructing the data states in the internal nodes offers a more nuanced picture of the state of a network as a whole at each point in time.

Our preliminary study considered various aspects of the fault detection and

localization problem, modeling the impact of faults in different ways across multiple networks and modem datasets. We showed that the two slack parameters of our proposed framework should be tuned to the specific data, but even with loosely chosen parameters, good results can be obtained. Overall, we demonstrated that in 90% of our test cases, the true fault location is either the first or second location suggested by the framework. This offers great potential for telecommunication infrastructure owners, as being able to visit at most two locations in 90% of fault resolution tasks would save both time and reduce emissions associated with driving.

Future research related to our findings could include a more in-depth study using real modem data. Investigating multi-fault scenarios and their impact is also relevant, especially for practitioners. Lastly, an open question remains how to convert the *fault-scores* assigned by our framework into a set of probabilities. This question is particularly interesting as it would allow for an intriguing crossover to the technician scheduling and routing problem, where uncertain fault locations and their respective fault probabilities could be actively incorporated into route-planning methodologies.

Chapter 5

Paper IV

Technician Routing and Scheduling for Tasks with Stochastic Resolution Locations

Status: Paper is submitted for publication in *Transportation Science*

Authors: Siv Sørensen[a], David Pisinger[a]
[a] DTU Management, Akademivej, Building 358, DK 2800 Kgs. Lyngby

Keywords: Stochastic Routing, Technician Routing and Scheduling, Markov Decision Process

Abstract

We introduce a novel stochastic extension of the Technician Routing and Scheduling Problem (TRSP), where tasks are associated with multiple potential resolution locations, each having a distinct probability of success. Our goal is to efficiently resolve all tasks while minimizing resource consumption, such as time or driving distance. All resolution locations are located in a sequence, and when inspecting a location the technician can either resolve the problem, or measure whether the fault is to the left or right in the sequence.

We propose a recursive algorithm based on a Markovian model formulation of the customer subproblem and show that the optimal policy can be found in $O(N^3)$ time, where N is the number of resolution locations for a task. This allows us to precompute expected travel times and solve the overall TRSP as a deterministic problem. We also introduce and evaluate several greedy strategies and compare their performance to the optimal solution.

Our computational experiments demonstrate significant time savings of 70-80% using the optimal policies compared to current industry approaches. The algorithm is shown to be effective even in uninformed settings, where no prior information about resolution probabilities is available. Moreover, we highlight the broad applicability of our methodology across different domains, such as pipeline failures and inventory management, where binary sequential search processes are similarly required for fault localization.

The findings present practical strategies for improving fault localization in telecommunication networks and beyond, offering real-world applicability and potential for substantial operational improvements.

5.1 Introduction

The global telecommunications sector is expected to experience significant growth in the coming years, driven by increasing data demands from digitalization in both commercial and governmental sectors, expanding global internet connectivity, and emerging markets in regions like Asia, Africa, and Latin America. Despite the focus on wireless technologies such as 5G, the wired telecommunication market is still projected to nearly double in value between 2022 and 2031 [12].

Wired infrastructure comprises copper wires, fiber-optic cables, and coaxial cables. While copper and coaxial cables remain integral to existing networks and specific applications, fiber-optic technology is expected to experience the most significant growth due to its superior speed and capacity.

Common to all wired networks is the challenge of maintaining extensive and

typically widespread physical infrastructure. In Denmark alone, the biggest infrastructure owner, TDC-NET, employs over a thousand technicians who service their wired network infrastructure by carrying out installation, maintenance, and fault order tasks.

While many tasks are associated with a specific location and an accurate estimate of the task duration, especially fault tasks can be more tricky. Fault tasks are typically created when a commercial or private customer reports an issue with their connection. However, the fault is not necessarily located at the customer premise, but could be anywhere along the wired connection between the customer's home and the backbone grid.

To better understand fault localization, a more thorough introduction to *last-mile* networks is needed. The *last-mile* network is the part of a telecommunications network that connects the service provider's central network (the backbone grid) to the end users, such as homes or businesses. These networks are arranged in a tree-like structure, with the root node connecting to the backbone grid, leaf nodes representing customer locations, and intermediate nodes acting as amplification points or cable splitters. Figure 5.1.1a illustrates part of a real *last-mile* network in Denmark. *Last-mile* networks can, despite the name, serve anywhere from a handful to hundreds of customers and can span areas from less than a kilometer to over 20 kilometers, depending on the wired technology.

When addressing a fault order, any cable splitting or amplification points between a customer's leaf node and the root node can be potential sources of error. This is because the signal traveling downstream from the backbone grid to the customer is influenced by each element it passes through. As a result, any fault in the network propagates downward, affecting all components and connections located beyond the failure point. Common faults include corrosion or installation errors associated with the signal amplifiers or cable-splitters, or issues related to the customer's equipment. This means that for each fault task, there is a predefined set of locations in the *last-mile* network serves as potential error (and resolution) points. Since these points are connected in sequence by a wired connection, in the search of a fault, technicians can exploit the fact that only the signal quality in the points below the failure point will be impacted by the fault whereas the signal quality in the points above the failure point will not.

The challenge of fault localization is not unique to telecommunications; it also occurs in various supply systems, such as pipelines and electrical circuits. For instance, in a leaking pipeline, pressure tests can be conducted at different pipe segments to identify where a pressure drop occurs, indicating a leak either at that segment or upstream. Similarly, in electrical cables, signal integrity tests can determine if a fault lies before or after a particular point.

A similar challenge also occurs in environmental monitoring, for example, iden-

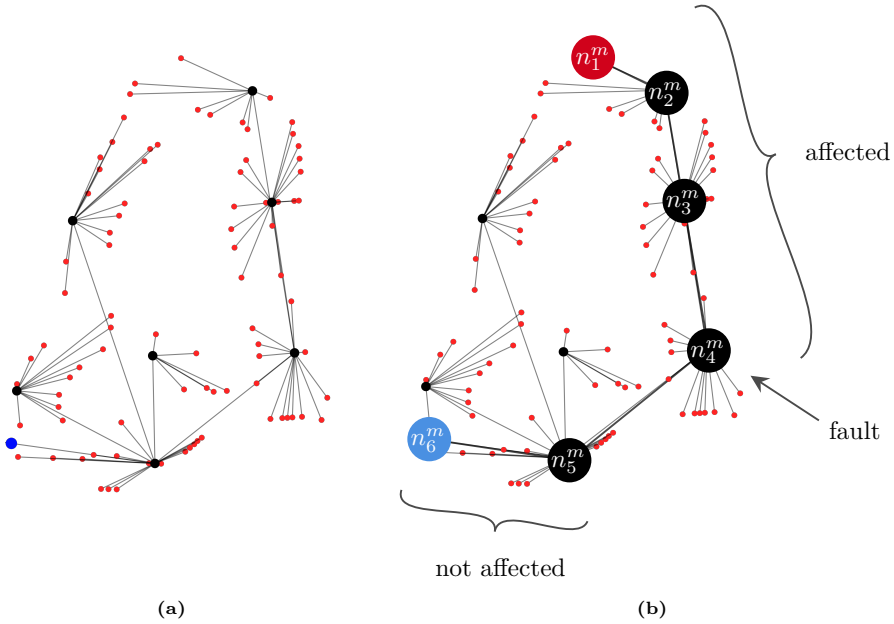


Figure 5.1.1: (a) Illustrates part of a real-life *last-mile* network. The blue node is the root node that connects the network to the back-bone grid. The red nodes are customers. The black nodes illustrate cable splitter and the lines are cable connections. Parts of the network connected to the root node have been left out for simplicity.
(b) Illustrates the sequential dependency among the options associated with a task (here pictured with six task options) as given by the option ordering list L^m. Here, the sequence is illustrated for the task m that is created if the red customer n_1^m experiences a fault. Assuming task option n_4^m is the location of the fault, then options $[n_1^m, ..., n_4^m]$ are affected by the fault and options $[n_5^m, n_6^m]$ are not.

tifying pollution sources in a river. Depending on the size of the river and the ease of accessibility, testing the water quality at various points along the river can be both time-consuming and costly and a structured approach can be of significant value.

Lastly, inventory management in large warehouses or distribution centers presents similar challenges, where discrepancies between recorded and actual stock levels often arise. Identifying the source of these discrepancies is crucial for maintaining accurate records and smooth operations, but can require lengthy investigations of various operational procedures. These procedures are typically sequential when considering individual products, involving multiple handling steps from product arrival to departure.

In many fault localization problems, the search is not done entirely blind. Typically, various data, either current or historical, serve as input to making informed decisions that guide the search. In telecommunication networks, we have previously shown how the most recent modem data from all the customers in the last-mile network of a fault can be used to assign failure probabilities to each potential point of failure [151]. In pipelines, maintenance records, the age of various components, and the external environment (mineral levels, soil moisture, etc) are all factors that can be used to determine the likelihood of different sections of the pipes being faulty.

The above demonstrates that the problem is widespread and arises in various forms across different sectors. The importance of localizing and resolving faults is two-fold: First, while a fault persists, it can be costly to a business (e.g., losing product through a leak), negatively impact customers (e.g., poor signal quality), or cause long-term damage to a system (e.g., an ecosystem surrounding a polluted river). Second, fault localization can be time-consuming and difficult to allocate resources to, as it is challenging to estimate how many locations must be investigated before the search is successful.

To the best of our knowledge, this particular problem in its entirety has not been studies in the literature, but several studies share similarities. For instance, multiple papers explore the challenge of parking in urban areas, where each parking location is associated with a probability of availability. Factors such as the distance to parking spots and parking fees are also considered when determining the order in which to visit the parking locations until successful [162, 169].

Based on the identified research gap and the broad applicability and impact of the problem, we propose several general search strategies as well as an optimal search procedure, and compare the effectiveness of each approach in a detailed computational study centered around telecommunications.

We initially decompose the problem into two levels: the outer and inner problem. The outer problem is a *Technicians Routing and Scheduling Problem* (TRSP), which determines the tasks each technician should handle and in what order. The inner problem focuses on determining the optimal sequence in which the technician should visit the set of possible resolution locations for each task.

For the inner problem—in addition to the greedy strategy currently employed by our partner telecommunications company, TDC-NET—we propose two additional greedy search strategies. We then argue that the inner problem is Markovian by formulating each decision that a technician makes as an independent subproblem. We model the subproblem using a new unified framework proposed by Powell [123, 121] for making, in particular sequential, decisions under uncertainty. Powell's versatile framework, applicable to all the major communities within stochastic optimization, addresses the fragmentation in the

field by unifying various notational styles used by different research groups, who often tackle similar problems with overlapping methods.

Based on this model of the subproblem, we demonstrate that any valid search policy for a technician, considering all possible realizations of uncertainty, can be represented as a Binary Search Tree (BST). Moreover, we show that finding the BST policy that minimizes the expected resolution time can be recognized as a generalization of the Optimal Binary Search Tree (OBST) problem, which can be solved optimally in polynomial time.

Finally, we propose a new bi-recursive algorithm for the generalization of the OBST, whose asymptotic complexity matches the efficiency of the well-known dynamic programming algorithm, proposed by Knuth [74] and others.

We present a comprehensive set of computational experiments to showcase the difference between the greedy strategies and the optimal approach. The experiments are based on a synthetic dataset of 640 instances with varying spatial configurations of tasks, different numbers of tasks, and different numbers of possible resolution locations per task.

The results are presented in two parts. First, assuming that the probability of finding the fault in each location is known (i.e., perfect foresight), and second, where probabilities are assumed unknown but the resultant optimal strategy is evaluated using the true underlying probabilities.

In conclusion, our study provides practical and generalizable strategies for improving fault localization in telecommunication networks. By evaluating different search approaches, we offer insights that enhance operational efficiency and have broad applicability beyond telecommunications.

The main **contributions** of this paper are:

- We present a new stochastic formulation of the TRSP, where each task has an associated set of resolution options. Exactly one of the options solves the task, and each option has a known probability of being the resolution point.
- We show that the problem can be broken up into a set of structurally identical customer subproblems that, when solved, enable us to solve a deterministic TRSP model that returns an expected time solution to the global problem.
- We propose several greedy search strategies for finding solutions to the set of subproblems.
- We model the subproblem as a sequential decision problem using a recently proposed unified modeling framework proposed by Powell [123].
- We present a recursive algorithm that yields optimal policies for the customer subproblem, with running time $O(N^3)$ where N is the number of

resolution locations for a given task.
- We present a new set of simulated instances constituting four different spatial configurations between tasks and task options. The instances vary in size with respect to the number of tasks and task options.
- We present **optimal expected value solutions** and compare these results to the proposed set of greedy methods.

The paper is **organized** as follows: First, in Section 5.2 we present an overview of relevant literature. Next, we formally describe the problem and introduce the relevant notation in Section 5.3. Section 5.4 details the proposed solution methodologies and provides both a formal TRSP model as well as a sequential decision model for the sub-problem. In Section 5.5 we describe the synthetic instances used in the computational study that is presented in the subsequent Section 5.6. Section 5.7 provides reflections and addresses the results in a broader context, while Section 5.8 concludes the paper.

5.2 Literature

The *Technician Routing and Scheduling Problem* (TRSP) was formally introduced by Tsang and Voudouris [152] in response to the real-life scheduling challenges faced by British Telecom with respect to their engineering workforce. Since then, numerous variants of the original problem formulation have been studied, but the core problem remains: scheduling a number of technicians to a set of tasks, minimizing total costs or time. Each technician and each task are typically characterized by an n-tuple of attributes. Essential characteristics of a technician include their start and end locations, while tasks are characterized by their specific locations and durations. Other common attributes include skills, time-windows, experience levels, spare part requirements, work-hours, and overtime availability.

Significant attention has been given to stochastic variants of the problem. Pillac et al. [115] were the first to introduce the dynamic TRSP, where tasks can arrive dynamically throughout the scheduling period. In an extensive review of all dynamic routing problems by some of the same authors, Pillac et al. [114] notably point out that there are two categories of dynamic routing problems: one where the underlying probability distributions of task arrivals are known and those where task arrivals are unpredictable. In their work on the TRSP, Pillac et al. [115] explored the latter category and proposed two approaches: a reoptimization approach using parallel *Adaptive Large Neighborhood Search* (ALNS) and a continuous reoptimization algorithm based on a multiple plan approach.

Multiple papers also address stochastic service times [25, 148, 174], Chen et al. [25] considered experience-based service times, where the duration required

to complete a task decreases as a technician becomes more familiar with the required skills. They use a multi-period timeline to showcase the effects of technicians gaining experience and model their problem as a *Markov Decision Process* (MDP). In each time period, a set of tasks is revealed, and a Mixed Integer Programming (MIP) model assigns tasks to technicians while minimizing both the current period's service time and an approximation of future service times—based on the current period decisions. The future value of today's workforce assignments is approximated with a dynamic programming-based solution approach, which, combined with an enumeration trick, transforms the problem into a linear mixed integer program. Their approach is thus similar to ours, as we also present an MDP embedded in a TRSP problem, that we then show can be transformed into a deterministic expected value TRSP using (exact) dynamic programming.

Pham et al. [113] considered an infinite planning horizon TRSP with dynamic tasks, multi-day service windows, and uncertain spare part demand. Their objective is to minimize daily costs while deciding each day which tasks to service and which spare parts to bring. Throughout the day, a technician cannot restock spare parts, meaning that unsuccessful first-attempt repairs necessitate second visits on a subsequent day. They model their problem as a sequential decision problem—also using the unified framework proposed by Powell [123]—where all decisions about next day's route and spare part inventory are made at the same time. They learn policies for the resultant high-dimensional model formulation by approximating the state value function using a graph neural network and subsequently use a genetic search to iteratively find and evaluate candidate decisions until a stopping criterion is met.

In general, there are two primary modeling approaches for multiperiod dynamic and stochastic logistical problems with combinatorial decision space: two-stage stochastic programming with recourse and sequential decision problems [113]. While two-stage stochastic programming with recourse is ideal for problems where uncertainties are revealed at distinct stages and allow for flexible recourse actions, it is less suited for real-time or ongoing decisions such as route decisions in a TRSP. On the other hand, the complexity of sequential decision models, both in computation and state representation, can make them impractical for problems with large decision spaces unless simplified or approximated. Still, sequential decision modeling approaches (such as MDP models) seem to be more common in recent research [154, 153, 104, 25, 113] with popular solution methodologies such as *Approximate Dynamic Programming* (ADP) and *Value Function Approximation* (VFA) relying on an underlying sequential decision model to be effective.

Modeling a sequential decision problem, or any stochastic problem for that matter, is often as difficult as designing the solution methodology. Moreover, where the community for deterministic decision problems has largely agreed

on a standard mathematical framework constituting an objective, constraints, and decision variables, the communities within sequential decision problems use several different notational styles [122]. This has resulted in a fragmented field of research, making it difficult to compare studies and methods. Additionally, scientific models that effectively capture the stochastic nature of the problem are often lacking.

In a review of stochastic dynamic vehicle routing problems, Ulmer et al. [153] found that 44 of the 68 papers considered lacked a proper model. Powell attempted to bridge this gap by proposing a unified framework for all communities within stochastic optimization [123, 121]. Moreover, he also argued that four *universal* policy classes exist that encompass all policy-based solution methodologies for sequential decision problems. For this reason, we have chosen to model our problem using this framework as opposed to modeling it as an MDP.

The most comparable work we could find to ours is the work by Wang et al. [164] on fault localization for wireless sensor networks. They consider the problem of finding an optimal testing sequence of components in a faulty network based on end-to-end data in sensor networks to minimize expected testing costs. Their problem differs from ours in that they assume multiple faults can occur simultaneously in the same network. Through an iterative approach, they identify a set of faulty components that can explain all end-to-end faulty behaviors; they then repair the components and observe through the end-to-end data if some connections are still faulty, and if so, the procedure is repeated. Although each component in their network is also associated with a testing cost and a fault probability, much like in ours, in our problem, the testing cost is dynamic because it includes the time it takes a technician to drive to a new location, and that time depends on which location the technician is driving from. Wang et al. prove that their problem is NP-hard and propose an optimal top-down recursive approach that builds a decision tree (testing policy) for the problem. Their approach takes polynomial time for line topologies and exponential time for tree topologies.

5.3 Problem

Given a set of tasks, M, each associated with a set of possible resolution options, N^m (Figure 5.1.1b show an example of a task with six resolution options), the problem at hand, is for a set of technicians, T, to resolve all tasks using the least amount of expected time. Here, both service and driving time are considered. Service time is a fixed amount of time, c^s, associated with visiting a task option, and driving time is obtained by converting the distances driven to time, assuming that every unit of distance driven takes c^d time.

For every task, exactly one option solves the task, and every option is associated

with a resolution probability π_n^m, where $\sum_{n \in N^m} \pi_n^m = 1$ for every task $m \in M$. Moreover, every task option and every technician is associated with a unique location (x, y). It is assumed that once a technician begins a task, the task must be resolved before moving on to the next task. This is to limit customer inconvenience as the customer's home must be accessible during the fault resolution process.

For network faults in telecommunication networks, the possible resolution options associated with a fault task are linked by a connection (such as a cable) that transmits data to and from a customer. Since a fault on this connection will be detectable in the downstream data *below* the failure point, the technician can determine whether the search should continue *up* or *down* the connection after inspecting a possible resolution point. This sequential dependency among task options is added to the problem by a predefined option ordering: $L^m = [n_1^m, n_2^m, ..., n_N^m]$ for all tasks $m \in M$. Here, if option n_i^m denotes the point of failure, then options $[n_1^m, ..., n_i^m]$ are affected by the fault and options $[n_{i+1}^m, ..., n_N^m]$ are not. An example of an option ordering L^m for a task with six options is shown in Figure 5.1.1b.

5.4 Methodology

The telecommunication company, TDC-NET, from which this problem originates, handles tasks with stochastic resolution locations by splitting the problem into two and then finds a solution to each problem independently of the other:

Outer problem TDC-NET generates a schedule for each of their technicians by solving a classic Technician Routing and Scheduling Problem. Tasks, $m \in M$, with stochastic resolution locations are estimated to take a fixed amount of time to resolve based on historical data. Moreover, the start and end location of such tasks are both estimated to be the location of the customer node equivalent to the first node n_1^m in the options ordering L^m. This means that the schedule for each technician is a simplification of the actual driving and resolution time needed to complete their assigned tasks.

Inner problem When a technician begins a task with stochastic resolution locations, they will first drive to the customer location, n_1^m. If the customer-premise is not the source of the fault, the technician will visit the remaining possible fault options in the order by which they are connected by cabling, i.e. n_2^m, n_3^m, ..., n_N^m, until the fault is found. The technician will then go directly to the next task in their schedule.

Our initial solution approach is based on a similar idea. We propose a set of greedy strategies, each defining a policy for how a technician should search

among the possible resolution options for a task. We then show how each strategy can be used to compute the *expected* distance—or, in our case, the expected time—between any two pairs of tasks, or between a technician's home and a task. This allows us to solve a regular deterministic TRSP model where the objective is to minimize the expected time. This approach factors in both the true start and end location of a technician with respect to each task. Additionally, the estimate of the total task resolution time is more precise as it considers service and individual driving time instead of historical averages.

To simplify the interpretation of the results of our computational study, we study a base version of the TRSP, where we ignore common constraints like *skills*, *working hours*, *skill levels*, and *breaks*. All these constraints can, without loss of generality, be added to the problem, as will be discussed at the end of this paper.

5.4.1 Service time

As explained in Section 5.3 our objective in the outer problem is to minimize the total expected *time*. To simplify this task, we define a new distance measure between any two task options or a technician and a task option. This new distance measure simply converts distances to time.

When a technician goes from a tasks option i (or from their home) to a new option j they spend the time equivalence of the distance traveled, $c^d \cdot dist(i,j)$, plus the service time, c^s, at option j. Thus, we define the new time-based distance measure as:

$$\begin{aligned} d(i,j) &= c^d \cdot dist(i,j) + c^s && \forall i \in \{N^m \cup T\},\ j \in N^m,\ i \neq j,\ m \in M \\ d(i,t) &= c^d \cdot dist(i,t) && \forall i \in N^m,\ t \in T,\ m \in M \\ d(i,i) &= 0 && \forall i \in N^m \end{aligned} \tag{5.1}$$

Here, recall from Section 5.3 that every unit of distance driven takes c^d time and c^s is the fixed service time associated with visiting a task option. The above *'distance'* measure is in minutes and is used for the remainder of this work.

5.4.2 The Outer Problem

We begin by introducing the base TRSP model that solves the outer problem and is used to obtain the task assignment and schedules for each technician and the total expected solution time. Most variants of the TRSP assume that the entire fleet starts in a centralized depot, but in our problem, each technician must start and end at their respective homes.

Let x_{ijt} define the binary decision for whether or not technician t goes from task i to task j, x'_{ti} the binary decision for whether or not technician t goes straight

from home to task i, and x''_{it} the binary decision for whether or not technician t goes from task i to their home. Lastly, let q_{tm} represent the position of task m in the sequence of tasks assigned to technician t, where q_{tm} is used to ensure sub-tour elimination. Then, the model can be formulated as follows:

$$\min \sum_{t \in T,\, i \in M} \left(c^1_{ti} x'_{ti} + c^2_{it} x''_{it} + \sum_{j \in M} c^3_{ij} x_{ijt} \right) \tag{5.2}$$

$$\text{s.t.} \sum_{t \in T} \left(x'_{tj} + \sum_{i \in M} x_{ijt} \right) \geq 1 \qquad j \in M \tag{5.3}$$

$$x'_{ti} + \sum_{i \in M} x_{imt} = x''_{mt} + \sum_{i \in T} x_{mit} \qquad m \in M,\, t \in T \tag{5.4}$$

$$\sum_{m \in M} x'_{tm} \leq 1 \qquad t \in T \tag{5.5}$$

$$q_{ti} - q_{tm} \geq 1 - |M|\,(1 - x_{mit}) \qquad m,\, i \in M,\, t \in T \tag{5.6}$$

$$x_{mit},\, x'_{tm},\, x''_{mt} \in \{0, 1\} \qquad m,\, i \in M, t \in T \tag{5.7}$$

$$1 \leq q_{tm} \leq |M| \qquad t \in T,\, m \in M \tag{5.8}$$

The objective (5.2) minimizes the *expected* time needed to solve all tasks, including the expected time between all pairwise tasks, c^3_{ij}, as well as the expected time from a technician to their first task, c^1_{ti}, and the expected time from their last task to home, c^2_{it}. Note that these time matrices include both the travel time, c^d, and the service time, c^s. The reason why we cannot use the $d(i, j)$ distance measure introduced in (5.1) directly instead of c^1_{ti}, c^2_{it}, and c^3_{ij}, is because the actual set of task options that will be visited by a technician is not known beforehand, as it depends on which task option turns out to be the fault location. Due to this uncertainty, we must compute the expected distances between all tasks, based on the $d(i, j)$ distance measure, by factoring in all possible outcomes of uncertainty.

Constraint (5.3) ensures that every task $m \in M$ is visited exactly once. Constraint (5.4) ensures flow conservation for every task $m \in M$ in every technician's route $t \in T$. Constraint (5.5) ensures that every technician $t \in T$ leaves their house at most once. Constraint (5.6) define an MTZ sub-tour elimination constraint [92] for every technician's route and constraints (5.7)-(5.8) define the domain of the variables. Although the MTZ sub-tour elimination constraints are known to be quite weak, they work fine for the considered problem, since a technician cannot service more than 20 tasks per day.

5.4.3 The Inner Problem

We tackle the inner problem in two different ways. First, in Section 5.4.3.1 we introduce three greedy approaches inspired by current industry practices. Second, in Section 5.4.3.2 we show that the inner problem can be modeled as a sequential decision problem. Based on on this model we present an optimal recursive approach that find the optimal policy for the inner problem.

The greedy approaches mainly serve as benchmarks for the optimal approach in the results section, but they are presented first to give a better intuition of the inner problem, before diving into the model details.

5.4.3.1 Greedy policy search

To obtain c^1_{ti}, c^2_{it}, and c^3_{ij}, the expected time matrices needed to solve model (5.2)-(5.8), we need a formulation for the expected time between any two *completed* tasks m_1 and m_2 using strategy ρ. I.e., once the fault in task m_1 has been found and resolved, what is the expected time until the fault in the subsequent task, m_2, has also been found and resolved. Let $G = (V, E)$ denote the fully connected graph that represents the network of possible trips, where V is the set of nodes (all task options and technician homes), and E is the set of edges corresponding to driving from one location to another. Then, the expected distance between two completed tasks can be expressed as:

$$\mathbb{E}(D(m_1, m_2, \rho)) = \sum_{i=1}^{|N^{m_1}|} \left(\pi_i^{m_1} \sum_{j=1}^{|N^{m_2}|} \pi_j^{m_2} \hat{d}(n_i^{m^1}, n_j^{m^2}, \rho) \right) \tag{5.9}$$

Here, $\hat{d}(n_i^{m^1}, n_j^{m^2}, \rho)$ is the length of the path from option i in task m_1 to option j in task m_2 using strategy ρ:

$$\hat{d}(n_1, n_2, \rho) = \sum_{(i,j) \in E(P_\rho(n_1, n_2))} d(i, j) \tag{5.10}$$

where $P_\rho(n_1, n_2)$ denotes the path from task options n_1 to n_2 following policy ρ, $E(P_\rho(n_1, n_2))$ the set of edges $(i, j) \in E$ in the path $P_\rho(n_1, n_2)$, and $d(i, j)$ the distance between any two nodes in G as given by equation (5.1).

The outer sum in equation (5.9) accounts for the possibility that a technician can resolve and conclude their search at any option in task m_1 before proceeding to task m_2. Given a specific endpoint i in task m_1, the inner sum in (5.9) incorporates the possibility that task m_2, likewise, can be resolved at any of its resolution options, by multiplying the fault probability $\pi_j^{m_2}$ by the path length from i to j using strategy ρ for all possible $j \in N^{m_2}$.

Equation 5.9 also works when finding the expected distance between a technician's home and a task or vice versa. If $m_\bullet$ corresponds to a technician home,

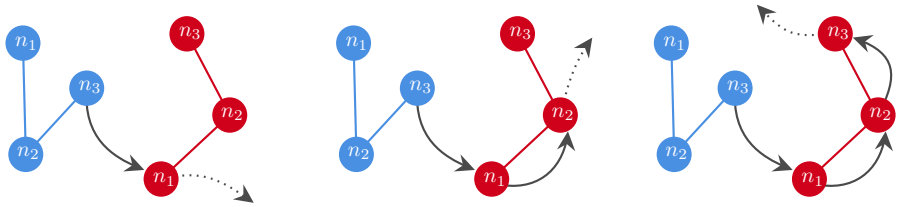


Figure 5.4.1: An example of the linear search (LS) strategy between two tasks (red and blue), each with three resolution options. The sequential dependency among the options is illustrated with lines matching the color of the tasks. Here, it is assumed that a technician concludes the search for the blue task at option n_3. Subfigures (A), (B), and (C) illustrate with arrows how the technician would travel according to the strategy if the red task is resolved at options n_1, n_2, and n_3, respectively.

then $|N^{m\bullet}|$ is simply one (as a technician is associated with a single home option), and the associated probability $\pi^{m\bullet}$ is also one.

Linear search (LS)

This strategy is equivalent to the approach currently employed by TDC-NET. A technician will start the search of a new task m at the customer node n_1^m and sequentially visit each option in L^m until the fault that resolves the task is found. We have included this strategy to serve as a baseline comparison for how much time can be saved by employing other approaches. Figure 5.4.1 illustrates the search strategy.

Using Subfigure (B) as an example, the edges in the path $P_\rho(n_3^{blue}, n_2^{red})$ are (n_3^{blue}, n_1^{red}) and (n_1^{red}, n_2^{red}) meaning (5.10) can readily be computed. Similarly, the paths between all other combinations of endpoints in the two tasks pictured can be used to obtain the expected distance between the two tasks using (5.9).

Binary search by number of options (BSNO)

This strategy is a classic binary search in which the center node in the sequence of options considered is visited. After each node visit, the sequence of options is therefore halved. In the case that the length of the sequence is even, the closest of the two most central nodes is visited with respect to the current location. Figure 5.4.2 illustrates the search strategy.

Binary search by cumulated probability (BSCP)

This strategy is inspired by binary search, but instead of halving the number of nodes, the cumulative probability is halved with each new node visit. For a

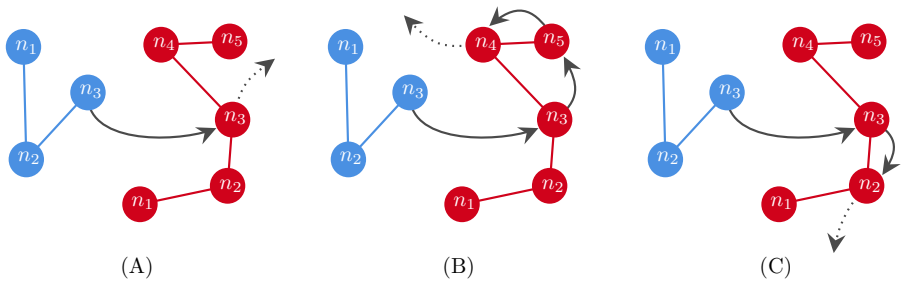


Figure 5.4.2: An example of the binary search strategy by number of options (BSNO) between two tasks (red and blue), with five and three resolution options, respectively. The sequential dependency among the options is illustrated with lines matching the color of the tasks. Here, it is assumed that a technician concludes the search for the blue task at option n_3. Subfigures (A), (B), and (C) illustrate with arrows how the technician would travel according to the strategy if the red task is resolved at options n_3, n_4, and n_2, respectively. In (B), the technician travels from n_3^{red} to n_5^{red} because, after visiting n_3^{red}, the technician knows the fault is to the right of n_3^{red} in sequence L^{red}. This leaves the subsequence of possible options: $[n_4^{\text{red}}, n_5^{\text{red}}]$. Of the two most central options in the sequence, n_5^{red} is the closest to n_3^{red} and is therefore visited first.

sequence of possible options $[n_a^m, \ldots, n_b^m] \in L^m$, the optimal split with respect to halving the cumulative probability is first found:

$$\underset{i \in N^m}{\operatorname{argmin}} \left| \sum_{j=a}^{i} \left(\pi_j^m\right) - \sum_{k=i+1}^{b} \left(\pi_k^m\right) \right| \tag{5.11}$$

The resultant sequence split is: $[n_a^m, ..., n_i^m]$ and $[n_{i+1}^m, ..., n_b^m]$. The decision of which option to visit next then becomes:

$$\text{next} = \begin{cases} n_i^m & \text{if } \sum_{j=a}^{i} \pi_j^m \geq \sum_{k=i+1}^{b} \pi_k^m \\ n_{i+1}^m & \text{else} \end{cases} \tag{5.12}$$

Figure 5.4.3 illustrates the search strategy.

5.4.3.2 Optimal policy search (OPS)

Although a greedy strategy might work well under specific conditions, it is seldom robust for a wide range of differing instances. Ideally, an optimal strategy factors in all available information, including all distances and all possible realizations of the success probability. Therefore, we take a step back and attempt to generalize the problem.

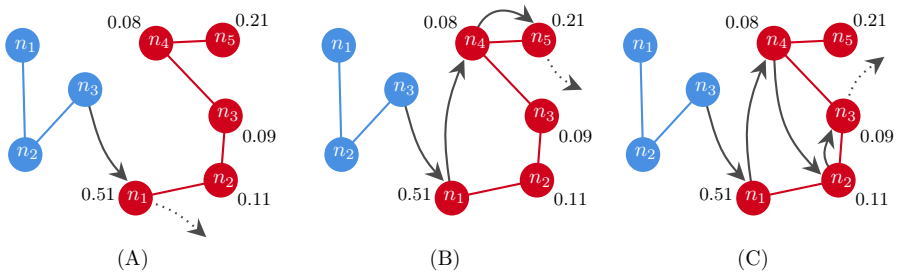


Figure 5.4.3: An example of the binary search by cumulated probability (BSCP) strategy between two tasks (red and blue), with five and three resolution options, respectively. The sequential dependency among the options is illustrated with lines matching the color of the tasks, and success probabilities are listed next to the red options. Here, it is assumed that a technician concludes the search for the blue task at option n_3. Subfigures (A), (B), and (C) illustrate with arrows how the technician would travel according to the strategy if the red task is resolved at options n_1, n_5, and n_3, respectively.

Sequential decision problem

Sequential decision problems occur as a subfield of problems within stochastic optimization. All sequential decision problems adhere to the same basic structure:

decision, new information, decision, new information, ...

The TRSP with stochastic resolution locations fits within this category. Each technician will repeatedly perform the following two steps until all tasks are resolved: a) decide on a task option to visit, and b) observe whether the visited option solves the task. If the task is not solved, the technician will also determine if the resolution point is located to the left or right of the visited option in the option ordering L^m.

In a sequential decision problem, rather than trying to find an optimal set of decisions, the goal is to find an optimal *policy*, denoted by $X^\rho(S_\nu)$, with respect to an objective that considers all realizations of uncertainty. A policy is a function that maps the current state of a problem, S_ν, at time ν, to an action, x_ν. Here, ρ, denotes the type of function $f \in \mathcal{F}$ and any tuneable parameters $\theta \in \Theta^f$ that make up a policy. Although a policy can be an analytical function such as a linear decision rule, a policy is *any* function that returns a feasible action to the current state of the system. Therefore, a policy can also be a separate optimization problem that must be solved in order to provide the action x_ν to state S_ν at time ν.

A sequential decision problem can be modeled using five elements: state variables, decision variables, exogenous information, the transition function, and the objective function [121]. We will define each of these elements in the context of

our problem inspired by Powell's modeling choices for both the static and the dynamic version of the Stochastic Shortest Path Problem [122].

Initially, the scope of our problem must be limited. The state set will explode in size if we attempt to search for a policy that takes into account multiple technicians, tasks, and task resolution options all at once. Adhering to the same approach as earlier, we define our subproblem as:

$$\begin{aligned} &\textit{A technician resolving a task } m_2 \textit{ given they start in option} \\ &n_\bullet^{m_1} \textit{ of task } m_1 \textit{ using the least amount of expected time.} \end{aligned} \tag{P1}$$

If we solve this problem for all possible 'ending points' $n_\bullet^{m_1} \in N^{m_1}$, then we can compute the expected distance, $\mathbb{E}(D(m_1, m_2))$, between any two tasks m_1 and m_2, exactly as in (5.9). This would enable us to solve the same deterministic model (5.2)-(5.8) (the same outer problem) as before.

State variables S_ν - Let ν represent the number of visits made by a technician, where a visit is defined as going from one node (task option) to another. Then, the state variable S_ν must capture all relevant current and past information for the decision maker at 'time' ν. Typically, $S_\nu = (N_\nu, I_\nu)$, where N_ν is the physical state of the system and I_ν is additional information.
Let $N_\nu = i \in \{N^{m^2} \cup n_\bullet^{m_1}\}$ be the node where the technician is located after ν visits. Moreover, let $I_\nu = (a_\nu, b_\nu)$ where $[n_{a_\nu}^{m_2}, \ldots, n_{b_\nu}^{m_2}]$ is the subinterval of options in L^{m_2} where the technician can go next after ν visits.
In our problem definition, we have defined a fixed starting point, $n_\bullet^{m_1}$. This means we can formulate the initial state as: $S_0 = (N_0, (a_0, b_0)) = (n_\bullet^{m_1}, (1, |N^{m_2}|))$.

Decision variables x_ν - Only one decision variable is needed, namely $x_{\nu ij} \in \{0, 1\}$, which is 1 if a trip from node $i \in \{N^{m^2} \cup n_\bullet^{m_1}\}$ to node $j \in N^{m^2}$ is made at time ν and otherwise 0.
It is always assumed that the decision returned by a policy, $x_{\nu ij} = X_\nu^\rho(S_\nu)$, is feasible at time ν, meaning $x_{\nu ij} = X_\nu^\rho(S_\nu) \in \mathcal{X}_\nu(S_\nu)$, where $\mathcal{X}_\nu(S_\nu)$ denotes the feasible region at time ν given state S_ν.
Without defining a feasible solution space, the zero solution where all $x_{\nu ij} = 0$ would be optimal. To avoid this, we define a valid decision after ν visits as a technician going from their current location N_ν to a node in the subinterval of valid options $[n_{a_\nu}^{m_2}, \ldots, n_{b_\nu}^{m_2}]$:

$$\mathcal{X}_\nu(S_\nu) \begin{cases} \displaystyle\sum_{j \in L^{m_2}[a_\nu, b_\nu]} x_{\nu, N_\nu, j} = 1 \end{cases} \tag{5.13}$$

Exogenous information $W_{\nu+1}$ - This is the new information that arrives after a decision is made. In this problem, after making decision $x_{\nu ij}$ and

arriving at node j, the decision maker is presented with two types of information: a) $\hat{s}_j \in \{0, 1\}$, which is 1 if node j solves task m_2, otherwise 0; and b) $\hat{h}_j \in \{left, right\}$, which is $left$ if the option that solves task m_2 is to the left of j in $L^{m_2}[a_\nu, b_\nu]$, and $right$ otherwise.

The transition function $S^M(S_\nu, x_\nu, W_{\nu+1})$ **-** Also known as the state (or system) transition model (hence the M in the superscript), is a set of equations that describe how all elements of the state variable $S_\nu = (N_\nu, I_\nu)$ evolve based on the decisions and the exogenous information, i.e. $S_{\nu+1} = S^M(S_\nu, x_\nu, W_{\nu+1})$. The physical state, N_ν is updated based on the decision variables alone:

$$N_{\nu+1} = \{j | x_{\nu,N_\nu,j} = 1\} \tag{5.14}$$

The additional information I_ν is updated based on the exogenous information as illustrated in Figure 5.4.4 as follows, given that $x_{\nu,N_\nu,j} = 1$:

$$a_{\nu+1} = \begin{cases} j & \text{if } \hat{s}_j = 1 \\ a_\nu & \text{if } \hat{s}_j = 0 \text{ and } \hat{h}_j = left \\ j+1 & \text{if } \hat{s}_j = 0 \text{ and } \hat{h}_j = right \end{cases} \tag{5.15}$$

$$b_{\nu+1} = \begin{cases} j & \text{if } \hat{s}_j = 1 \\ j-1 & \text{if } \hat{s}_j = 0 \text{ and } \hat{h}_j = left \\ b_\nu & \text{if } \hat{s}_j = 0 \text{ and } \hat{h}_j = right \end{cases} \tag{5.16}$$

In the above equations, once the resolution node j (where $\hat{s}_j = 1$) is found, the valid subinterval of nodes where the technician can go next includes only the resolution node j. Adhering to the feasible space of actions, $\mathcal{X}_\nu$, given in (5.13), this allows the technician to travel from node j to j using no additional time for the remainder of the time horizon considered. If node j is not the resolution location, then the technician will be forced to travel to a new node in the next time period $\nu + 1$, as the new feasible subinterval of nodes defined by $a_{\nu+1}$ and $b_{\nu+1}$ never includes a previously visited node.

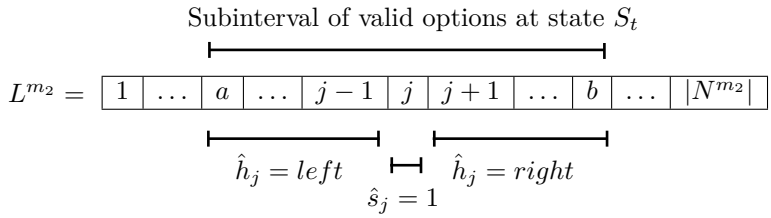


Figure 5.4.4: An illustration of how $I_\nu = (a_\nu, b_\nu)$ is updated at time $\nu + 1$ using (5.15) and (5.16) from the transition function $S^M(S_\nu, x_\nu, W_{\nu+1})$. The interval above L^{m_2} illustrates the subinterval of valid options in state S_ν, and the three intervals below L^{m_2} illustrate the three possible valid subintervals in state $S_{\nu+1}$ based on the realization of the exogenous information $W_{\nu+1}$.

The objective function - The objective function in a *general* sequential decision problem (assuming minimization) can be stated as [122]:

$$\min_{\rho} \mathbb{E}_{S_0} \mathbb{E}_{W_1,\ldots,W_\Gamma|S_0} \left\{ \sum_{\nu=0}^{\Gamma} C_\nu(S_\nu, X_\nu^\rho(S_\nu), W_{\nu+1}) | S_0 \right\} \tag{5.17}$$

The policy ρ, which returns the minimal value for the above expression is referred to as the *optimal policy*. Optimal policies are hard to come by, so typically the goal is to find a *good* policy. In the above, $C_\nu(S_\nu, X_\nu^\rho(S_\nu), W_{\nu+1})$ is the cost (or reward) of the decision $x_\nu = X_\nu^\rho(S_\nu)$ made at time ν based on state S_ν using policy X^ρ and the exogenous information, $W_{\nu+1}$, revealed after the decision is made. Uncertainty can arise from the new information $W_1, \ldots, W_\Gamma$ and from the initial state of the problem S_0, hence the two expectation operators. Γ defines the timeline of the problem.

In *our* problem, Γ is easy to define as a technician at most visits all resolution options $n \in N^m$ in a task before the task is resolved. Moreover, in our problem, the exogenous information that is revealed after action x_ν does not affect the cost (time spent), as the travel time from node i to j as well as the service time are both static. Lastly, going back to our subproblem definition P1, we wish to find a policy for solving task m_2, given we start the search in a specific option, $n_\bullet^{m_1}$, of another task m_1. Therefore, there is no uncertainty associated with the initial state S_0 and the general expression in (5.17), can in our problem be simplified to:

$$\begin{aligned} &\min_{\rho} \mathbb{E}_{W_1,\ldots,W_\Gamma|S_0} \left\{ \sum_{\nu=0}^{|N^{m_2}|} C_\nu(S_\nu, X_\nu^\rho(S_\nu)) \mid S_0 \right\} \\ &\text{where} \quad C_\nu(S_\nu, X_\nu^\rho(S_\nu)) = \sum_{j \in N^{m^2}} x_{\nu,N_\nu,j} \cdot d(N_\nu, j) \end{aligned} \tag{5.18}$$

This concludes the *base model* for our problem subproblem (P1).

The next step is to design an effective policy for the problem, typically through two core strategies: *Policy Search* and *Lookahead Approximations*. Policy Search involves exploring a set of decision-making functions within the same family, such as various linear decision rules differentiated by their parameters. Lookahead Approximations estimate the immediate cost of a decision, x_ν, and approximate future costs contingent on that decision. The optimal decision, x_ν, minimizes the sum of current and estimated future costs [123].

The three greedy strategies we previously introduced represent a basic form of Policy Search. Each strategy implicitly defines a decision rule that decides where a technician should go next based on the index of a node in the valid subinterval $L^{m_2}[a_\nu, b_\nu]$ or additionally also a cumulated probability sum. The

greedy strategies therefore belong to the same family of functions $f \in \mathcal{F}$ where the tunable parameters $\theta \in \Theta^f$ include the importance, or weight, assigned to the index of a node as well as its cumulated probability with respect to a two-way split of the valid subinterval of nodes.

Before we move on, we make an important observation about all classes of policy functions which are feasible for our subproblem, namely that:

Theorem 5.1 *Any policy that is feasible for subproblem* (P1) *makes up a decision tree that is a Binary Search Tree (BST) over all options $n \in N^{m_2}$ when considering all realizations of the uncertainty.*

PROOF. The exogenous information revealed after each trip ν depends only on the current location of the technician and the true resolution node in a given scenario. Since the task, m_2, that the technician is trying to resolve has $|N^{m_2}|$ possible resolution options, a feasible policy will always return a path from $n_\bullet^{m_1}$ to some $n_j^{m_2} \in N^{m_2}$ where $\hat{s}_j = 1$. This results in $|N^{m_2}|$ distinct paths when considering all realizations of uncertainty.

At each time step ν, assuming m_2 is still unresolved, the only information a technician receive about the fault location is whether it is located to the right or left of the current physical location N_ν. Since a policy $X^\rho(S_\nu)$ returns an action based solely on the current state of the system, there are always at most two distinct decisions available in any physical location N_ν. These (at most) two decisions can be depicted as children of a node in a binary tree.

In a BST, for every node, the indices in the left subtree are less than the node index, and the indices in the right subtree are greater than the node index. This structure ensures that only the nodes in the valid subinterval $L^{m_2}[a_{\nu+1}, b_{\nu+1}]$ are available after any ν (unsuccessful) edge traversals in the BST. Thus, each decision in a valid policy corresponds to a binary choice that adheres to the BST structure by ensuring that the left and right subtrees maintain the index ordering property.

This concludes the proof that any policy feasible for subproblem (P1) inherently satisfies the properties of a BST when considering all realizations of the uncertainty.

Figure 5.4.5 illustrates an example of a valid policy and an invalid policy.

Since we now have a definition that covers all valid policies, we can perform a policy search over all possible BSTs and return the best one. By removing the outer summation over i in (5.9) and setting $i = n_\bullet^{m_1}$, the expected cost of a policy ρ from an option $n_\bullet^{m_1}$ in task m_1 to task m_2 becomes the sum of the path length, $\hat{d}$, from $n_\bullet^{m_1}$ to each of the options in m_2, weighted by their respective

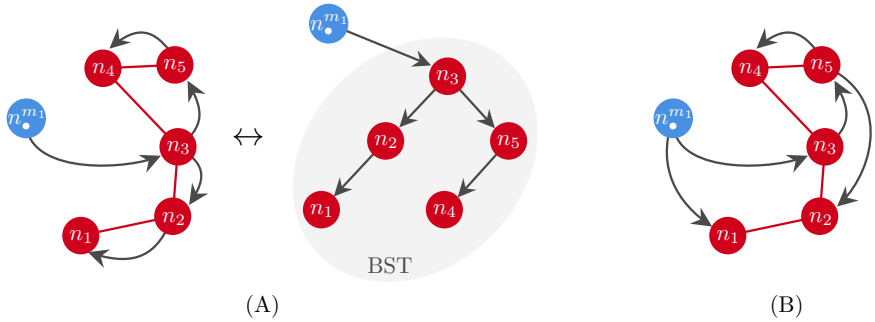


Figure 5.4.5: (A) illustrates a policy that is valid under all realizations of uncertainty. That is, the policy provides a path to all nodes in the red task. Regardless of which option $n \in N^{m_2}$ (red) turns out to be the resolution point, the policy depicted provides a set of valid actions to a technician with respect to the exogenous information and the current state. (A) also illustrates how this policy corresponds to a BST. (B) depicts an *invalid* policy with two design flaws: (i) in the initial state S_0, where $N_0 = n_\bullet^{m1}$ and no exogenous information has been revealed yet, the policy provides two actions to the same state; and (ii) assuming the resolution point is node n_2, then at $N_{\nu=1} = n_3$, the exogenous information revealed will tell the technician to go left in sequence L^{red}, but instead, the policy directs the technician to option n_5, which is located to the right of the current state in L^{red}.

resolution probability π:

$$\mathbb{E}(D(n_\bullet^{m_1}, m_2, \rho)) = \sum_{j=1}^{|N^{m_2}|} \pi_j^{m_2} \hat{d}(n_\bullet^{m^1}, n_j^{m^2}, \rho) = \sum_{(i,j)\in E(\rho)} \left(d(i,j) \sum_{k \in D_j^\rho} \pi_k^{m_2} \right) \tag{5.19}$$

In the above equation, an alternative way of calculating the expected cost is also presented, which will be useful later. Here, the length of each edge (i, j) in the policy tree $E(\rho)$ is multiplied by the sum of probabilities in the descendant nodes D_j^ρ of node j (including j), and then summed. This corresponds to adding together the contribution from each node that uses edge $(i, j) \in E(\rho)$ in their path from $n_i^{m_2}$, with respect to policy ρ.

To guarantee the optimal policy is found, a policy search must compute and compare the value of (5.19) for all possible BSTs. The number of unique BSTs that can be formed with n distinct nodes, such that the inorder traversal returns the nodes in sorted order, is given by the n-th Catalan number, which can be computed as:

$$\mathcal{C}_n = \frac{1}{n+1}\binom{2n}{n} \tag{5.20}$$

The growth rate of the Catalan numbers is superpolynomial, meaning it grows faster than any polynomial but not as fast as exponential functions. As an example, the 10th Catalan number is 16796. This means that if a task has ten potential resolution options, solving our subproblem (P1) would require generating 16796 BSTs and comparing their expected cost. Doing so would only yields an optimal policy for resolving task m_2 assuming the search starts in a specific option $n_\bullet$ in task m_1. Assuming task m_1 also has ten potential resolution options, we would need to do this ten times, for each of the options $n_i \in N^{m_1}$, to find the expected cost between the two tasks $\mathbb{E}(D(m_1, m_2, \rho))$. Moreover, doing this for all pairwise combinations of tasks, assuming, as an example, that there are also ten tasks, would result in having to consider $(10 \times 10) \times 10 \times 16796$ BSTs in total to generate the expected cost matrix needed to solve the full TRSP problem modeled by (5.2)–(5.8).

In short, although a policy search over all feasible policies might be doable for small problem instances, it does not scale well for larger instances.

Exploiting the Markovian model structure

The alternative to a *policy search* that aims to find the policy ρ which solves (5.18) is a *lookahead approach*. Lookahead solution approaches involve evaluating the impact of current decisions on the future and are based on Bellman's Principle of Optimality which states that: *"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [8]."*

The perhaps most famous example of a lookahead approach is dynamic programming where an optimal solution to a problem can be found by breaking it into sub-problems and then recursively finding the optimal solutions to the sub-problems.

A lookahead approach corresponds to solving:

$$X^*_\nu(S_\nu) = \arg\min_{x_\nu} \left(C(S_\nu, x_\nu) + \mathbb{E}\left\{ \min_\rho \mathbb{E}\left\{ \sum_{\nu'=\nu+1}^{\Gamma} C(S_{\nu'}, X^\rho_{\nu'}(S_{\nu'})) \middle| S_{\nu+1} \right\} \middle| S_{\nu+1}, x_\nu \right\} \right) \tag{5.21}$$

where $X^*_\nu(S_\nu)$ returns the optimal decision x_ν to being in state S_ν [123].

The above equation is easier to comprehend in the context of a decision tree $\mathcal{T}$ with a discrete set of actions and associated random outcomes, as is the case in our problem. Let the states S_ν correspond to the nodes in $\mathcal{T}$, S_ν corresponds to the root node, and x_ν—the initial set of possible decisions—corresponds to the set of edges connected to the root node in $\mathcal{T}$. Then, $C(S_\nu, x_\nu)$ is the cost of the initial decision made at time ν, and the remainder of the equation equates to

the expected cost of all future decisions, given that the initial decision x_ν leads to state $S_{\nu+1}$.

The first expectation is over the first set of random outcomes $W_{\nu+1}$, resulting from the initial decision x_ν.

Next, the policy ρ represents the action x_ν at every subsequent node S_ν for all realizations of the future $\nu' > t$ within the time horizon Γ. Thus, solving the minimization over all policies ρ corresponds to considering a set of recursive calls $X^\rho_\nu(S_\nu)$, which return the action at time $\nu' \in [\nu+1, \dots, \Gamma]$ using policy ρ. These actions are used to compute and sum the costs of the remaining time frame. Here, the second expectation covers all the random outcomes W_ν, $\nu' \in [\nu+2, \dots, \Gamma]$ revealed after every future action.

The optimal solution to any Markovian stochastic process can theoretically be computed using (5.21), but in practice, this is often computationally intractable. However, it can still be beneficial to model a stochastic problem as Markovian—which can always be done if enough information is included in the state variable (even if this is clearly computationally intractable)—as it makes all lookahead approaches applicable including the vast class of approximation strategies often exclusively used for problems modeled specifically as Markov Decision Processes.

We will, however, show that in our specific problem (P1), approximation strategies are not needed, as the problem can be solved to optimality recursively by exploiting the Markovian structure of our base model.

Recursion for finding the cheapest strategy

Finding the optimal policy for (P1) can be recognized as a generalization of the *optimal binary search tree* (OBST) problem [74]. Where in our case the edges can have arbitrary distances, the OBST has unit cost on edges. The dynamic programming recursion for the extended problem is described in the following.

Recall, π_i is the probability of solving a task at node i and $d(i,j)$ is the distance between node i and j in time units computed using (5.1). Furthermore, for efficiency reasons, we pre-calculate: $\Pi_{a,b} = \sum_{i=a}^{b} \pi_i$.

We now introduce two recursions: A left recursion λ and a right recursion ϕ, depending on whether we continue the search to the left or right of the current position in the considered interval of nodes. Let $\phi_{a,b}$ be the cheapest cost of a policy when starting at node $a-1$ and searching among nodes $a, \dots, b$. Let $\lambda_{a,b}$ be the cheapest cost of a policy when starting at node $b+1$ and searching among nodes $a, \dots, b$. Then we can use the bi-recursion:

$$\lambda_{a,b} = \begin{cases} 0 & \text{if } a > b \\ \min\limits_{i=a,\dots,b} \left(\Pi_{a,b} \cdot d(b+1,i) + \lambda_{a,i-1} + \phi_{i+1,b}\right) & \text{if } a \leq b \end{cases}$$

$$\phi_{a,b} = \begin{cases} 0 & \text{if } a > b \\ \min\limits_{i=a,\dots,b} \left(\Pi_{a,b} \cdot d(a-1,i) + \lambda_{a,i-1} + \phi_{i+1,b}\right) & \text{if } a \leq b \end{cases}$$

If we assume that node 0 is the start-node, $n_\bullet^m$, then the optimal solution value is found as:

$$z_{opt} = \phi_{1,|N|}$$

In each recursive call we find the optimal node to go to next, i, in the considered interval $a, \dots, b$. The cost of going to a node i is the distance from the previous node to i; $d(b+1,i)$ or $d(a-1,i)$, depending on which recursion is used. This edge will be used by a technician if the task m_2 can be resolved in any of the nodes in the interval $a, \dots, b$, hence why the cost of the edge must be multiplied with $\Pi_{a,b}$. This is exactly the same argument we used in (5.19) when we showed the expected distance between an option $n_\bullet^m$ and a task m_2 using policy ρ can be calculated as:

$$\mathbb{E}(D(n_\bullet^m, m_2, \rho)) = \sum_{(i,j)\in E(\rho)} \left(d(i,j) \sum_{k \in D} \pi_k^m \right)$$

While the first term in each recursion accounts for the cost of going to node $i \in \{a, \dots, b\}$, the two last terms accounts for the cost of having an onward path to the remainder of the nodes in $\{a, \dots, b\}$. Here, $\lambda_{a,i-1}$ is the cost of the nodes to the left of i and $\phi_{i+1,b}$ is the cost of the nodes to the right of i. This is exactly equivalent to dynamics of our transition functions introduced in the base model and pictured in Figure 5.4.4. Algorithm 5.1 presents the pseudocode for the recursion where the initial call RECURSION$(n_\bullet^m, n_1^m, n_N^m)$ returns the value of $\mathbb{E}(D(n_\bullet^m, m_2, \rho^*))$, where ρ^* is optimal. Figure 5.4.6 illustrates how each call of the bi-recursion works.

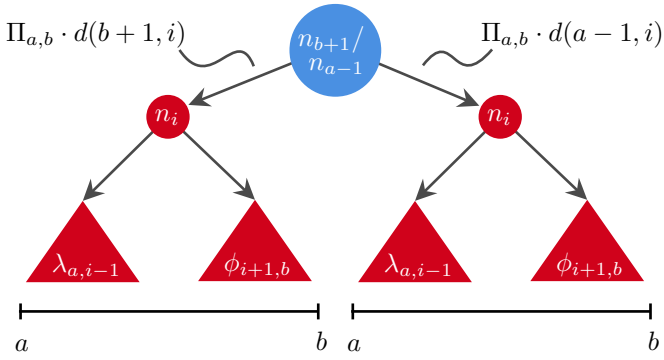


Figure 5.4.6: An illustrative example of the bi-recursion given by Algorithm 5.1.

To improve running time, we pre-calculate the values of $\lambda_{a,b}$ and $\phi_{a,b}$ for all intervals $[a, b]$ of size 1, then for all intervals of size 2, etc. This ensures that all sub-values in the recursion are well-defined before being called. Since we have $O(N^2)$ intervals, and the inner loop of the recursion takes $O(N)$ time, this results in an overall running time of $O(N^3)$.

Knuth showed that using the Monge Property [94], the running time of constructing an OBST can be reduced to $O(N^2)$ [74]. However, the Monge property in OBST is proven under the assumption that the edge costs associated with choices are uniform, which they are not in our extension of the problem.

The space consumption is clearly $O(N^2)$, since we save one value of $\lambda_{a,b}$ and $\phi_{a,b}$ for each pair of a, b.

The input size is $|I| = N^2$ since we have $d(i, j)$ given between all pairs of nodes. This means that the running time is $O(I\sqrt{I})$ and space consumption is $O(I)$.

The recursion only finds the solution value of the optimal strategy. If we want to find the corresponding optimal strategy, we need to save the optimal index i in the inner loop of the recursion.

Based on the above we arrive at our final theorem.

Theorem 5.2 *The optimal policy for* (P1) *can be computed in polynomial time* $O(N^3)$*, where* N *is the number of nodes.*

Proof. First, we established the Markovian structure of (P1) in our base model and proved that an optimal policy under all realizations of uncertainty is a BST, as stated in Theorem 5.1. We then demonstrated that the set of edges constituting the optimal policy could be generated recursively. Lastly, we showed that the recursive approach operates in $O(N^3)$ time complexity. Therefore, by combining these results, we conclude that the optimal policy for (P1) can indeed be computed in polynomial time $O(N^3)$, where N is the number of nodes, as stated in the theorem.

Expected distance matrix

To generate the entire expected distance matrix between all M tasks such that the deterministic model (5.2)-(5.8) can be solved, the recursion (Algorithm 5.1) must be run M^2N times giving a total running time of $O(M^2N^4)$.

The expected distance matrix between all technicians' homes and the M tasks must also be computed, but given there is no uncertainty associated with the technicians' home locations, this does not change the total running time of $O(M^2N^4)$.

Algorithm 5.1 Find an optimal policy

Input: A distance matrix of size $N \times N$ and probability π_j for each node

procedure RECURSION(k, a, b) ▷ *Arguments: parent node k and current interval $[a, b]$*
  if $a > b$ **then return** 0
  $min\_cost \leftarrow \infty$
  for $i \in [a, b]$ **do** ▷ *Loop through each node in interval $[a, b]$*
    $cost_{parent} \leftarrow \sum_{j=a}^{b} \pi_j \cdot d(k, i)$
    $cost_{left} \leftarrow$ RECURSION$(i, a, i-1)$
    $cost_{rigth} \leftarrow$ RECURSION$(i, i+1, b)$
    $cost_i \leftarrow cost_{parent} + cost_{left} + cost_{right}$
    if $cost_i < min\_cost$ **then**
      $cost\_min \leftarrow cost_i$
      save best i, and corresponding left and right child node
  return min_cost

5.5 Instances

A set of synthetic simulated instances is used to evaluate the expected performance of the different greedy strategies as well as the optimal policy. In all instances, we only consider a single technician. This is because the base TRSP model we are solving in (5.2)–(5.8) in practice often will find it optimal to deploy only a single technician and let the remaining technicians stay at home. The time contribution from technicians driving to and from their homes is reduced significantly if some technicians never leave their homes. This could of course be avoided in a number of ways such as: forcing all technicians to leave their homes, introducing work-load balance, or by introducing an upper time limit to a technician's work day. However, given the scope of this work is to compare different strategies, we believe solving the proposed base TRSP model using one technician yields the most fair and generalizable results.

We consider four different spatial instance configurations as pictured in Figure 5.5.1. Each instance fits inside a one-by-one square to ease the conversion from distance to time units (5.1).

(A) Tree-like Tasks are arranged in a circular manner around the center. Within each task, the sequence L^m denoting the ordering of the task options is decided based on the distance from the center to each option. This spatial configuration closely resembles real telecommunication networks.

(B) Circular Tasks are arranged in a circular manner around the center. The option sequences are decided randomly.

(C) Column-wise Tasks are arranged in a column-wise manner. The option sequences are decided randomly.

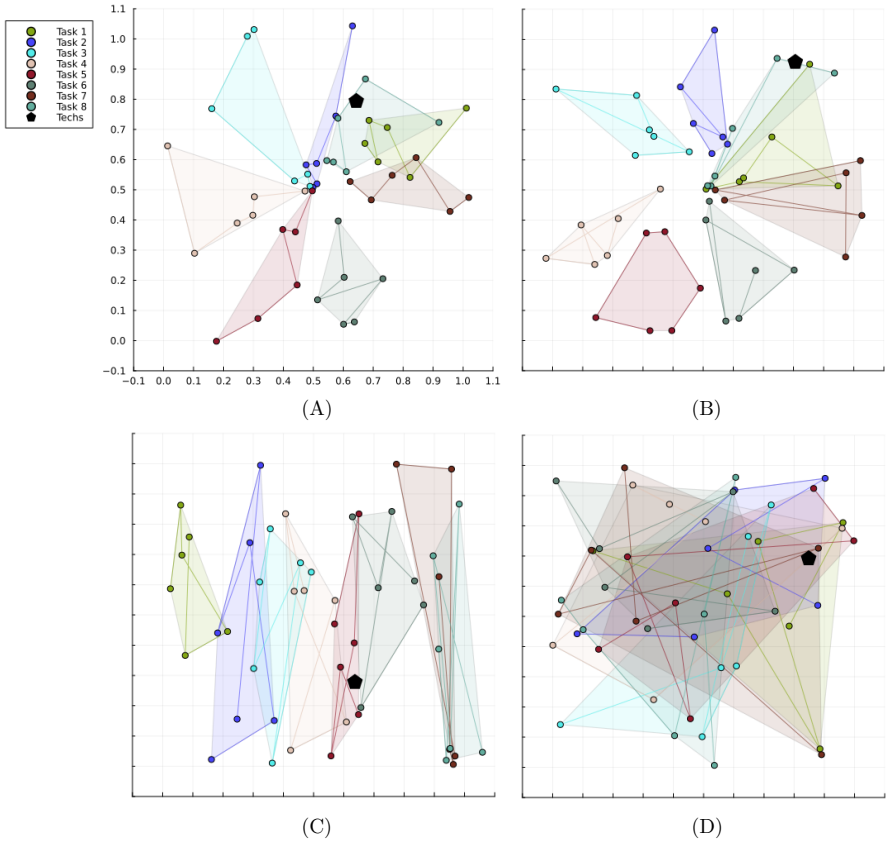


Figure 5.5.1: Illustration of each of the four different spacial configurations for an instance; (A) tree-like, (B) circular, (C) column-wise, (D) random. Here pictured with one technician and eight tasks and six possible resolution options per task. Every instance spans approximately one unit distance across its widest point.

(D) Random Tasks and task options are placed randomly. The option sequences are also decided randomly.

For each spatial configuration, we consider the number of tasks to range from three to ten. The same range applies to the number of task options. This results in a total of 64 combinations of tasks and task options. For each combination, we compute ten different instances, resulting in 640 instances per spatial configuration and 2560 instances in total.

The resolution probabilities π_i^m for all options $i \in N^m$ are generated using a Dirichlet distribution with a concentration parameter of 1.6. The Dirichlet distribution is commonly used to model probability distributions over a discrete

set of outcomes because it generates a vector of probabilities that always sum to one.

A concentration parameter of 1.6 indicates that the generated probabilities will not be too skewed, meaning each possible resolution option has a relatively similar likelihood of being the correct one. This is particularly useful in scenarios where there is limited prior knowledge about which option is most likely to resolve the fault, as it prevents the probabilities from being overly biased. The choice of the Dirichlet distribution also aligns with real-life distributions where some inherent uncertainty exists, making it a natural fit for modeling probabilities in a fault diagnosis setting.

5.6 Computational results

Results are reported in three parts. First, we compare all the sub-optimal greedy methods. Second, we include the results obtained from using the optimal policy search (OPS) approach. Third, we compute an *uniformed* OPS solution and compare it to the true OPS solution. All results are based on the total expected time it takes one technician to solve all tasks including driving from and to home, and are obtained by solving the TRSP model (5.2)-(5.8).

In each part, we present a table with four sections corresponding to the four spatial instance configurations. This is done to evaluate how much the layout of the tasks and task options affect the results. Additionally, we investigate the relationship between the service time, c^s, and the driving time per unit distance, c^d. For all analyses, the driving time per unit distance is fixed at 30 minutes, where all instances approximately span one unit distance in both height and width. Contrastively, the service time is varied within the interval of $[0, 5, 10, 15, 20, 50, 100]$ minutes.

The running time of each instance varies but for one of the biggest instances, using a service time of zero minutes, the expected distance matrices used in OPS can be computed in approximately 0.05 seconds and the resultant deterministic TRSP solved to optimality in 3.5 seconds. All results are computed using Gurobi 11.0.3 on a 2.6 GHz Intel Core i5 PC with 16 GB of RAM.

5.6.1 Greedy methods

Table 5.6.1 reports the results for the three greedy methods: linear search (LS), binary search by number of options (BSNO), and binary search by cumulated probability (BSCP). Across all instances and for all combinations of service and driving time, BSCP consistently finds the best solutions when averaged over all 640 instances. BSNO always returns solutions that are 4%-8% worse, while LS performs significantly worse, with results that are 60%-75% worse on average.

Table 5.6.1: Averaged results for all 640 instances for each of the four spatial configurations considered using the three greedy methods. Here, LS is the linear search method, BSNO is the binary search by number of options method, and BSCP is the binary search by cumulated probability method. The last column reports the best average of the total expected time found by any of the three methods. The right side of the table shows how much worse each method performed compared to the best greedy method, and the left side reports the number of instances out of all 640, where the given method found the best solution. Note that it is possible for multiple methods to find the same solution to the same instance.

c^s [min]	c^d [min]	# instances where best			% worse than best avg.			**Best** avg. total expected time [min]
		LS	BSNO	BSCP	LS	BSNO	BSCP	
Tree-like instances								
0	30	0	226	422	61.5%	3.4%	0%	111.75
5	30	0	111	537	65.2%	5.0%	0%	183.00
10	30	0	61	587	66.9%	5.7%	0%	254.25
15	30	0	38	610	67.8%	6.0%	0%	325.50
20	30	0	32	616	68.4%	6.2%	0%	396.74
50	30	0	17	631	69.8%	6.5%	0%	824.23
100	30	0	14	634	70.4%	6.5%	0%	1536.72
Circular instances								
0	30	4	205	436	59.2%	5.8%	0%	126.85
5	30	2	131	513	63.6%	6.3%	0%	197.98
10	30	0	79	567	65.7%	6.6%	0%	269.11
15	30	0	53	593	66.9%	6.7%	0%	340.24
20	30	0	35	611	67.7%	6.8%	0%	411.36
50	30	0	14	632	69.6%	6.8%	0%	838.13
100	30	0	13	633	70.5%	6.7%	0%	1549.41
Column-wise instances								
0	30	1	207	438	65.6%	5.2%	0%	171.49
5	30	1	149	496	66.6%	5.9%	0%	242.61
10	30	0	106	540	67.7%	6.2%	0%	313.74
15	30	0	88	558	68.4%	6.4%	0%	384.87
20	30	0	62	584	68.9%	6.5%	0%	456.00
50	30	0	24	622	70.2%	6.7%	0%	882.76
100	30	0	15	631	70.8%	6.7%	0%	1594.04
Random instances								
0	30	1	165	481	74.8%	7.7%	0%	212.12
5	30	1	122	523	74.0%	7.6%	0%	283.25
10	30	0	96	550	73.5%	7.5%	0%	354.37
15	30	0	75	571	73.1%	7.4%	0%	425.50
20	30	0	57	589	72.9%	7.4%	0%	496.63
50	30	0	25	621	72.3%	7.1%	0%	923.40
100	30	0	16	630	71.9%	6.9%	0%	1634.67

In practical terms, this means that even if a real-life company lacks data on the resolution probability of each location, implementing a simple binary search yields significantly better results than the linear search method currently used by our collaborating telecommunication company.

When comparing the four different spatial configurations, the overall trend remains consistent, with only slight variations in the percentage by which LS and BSNO underperform compared to BSCP.

For instances where the service time is zero minutes—meaning that only the distances between tasks influence the decision—BSNO finds the best solution in approximately one-third of the instances, while BSCP achieves the best results in the remaining two-thirds. However, as service time increases, the effectiveness of BSNO diminishes, with BSCP finding the best solutions more and more frequently.

5.6.2 Greedy methods compared to optimal policy search

Table 5.6.2 compares the results for the three greedy methods: LS, BSNO, and BSCP with the results of using the optimal policy search (OPS). The greatest performance improvement of OPS compared to the three greedy strategies occurs when the service time is zero minutes. For the circular, column-wise, and random instances, the difference in average total expected time is approximately 123%, 42%, and 34%, respectively, for the LS, BSNO, and BSCP strategies.

In the tree-like instances, the best performance improvement is also found for a service time of zero minutes, but here, the difference in average expected time between OPS and the three greedy strategies is only 81%, 16%, and 12%, respectively.

As the service time increases, the BSCP strategy is equal to the OPS in more and more instances. As a result, for a service time of 100 minutes, the BSCP strategy is only 2% worse on average compared to the OPS approach.

These findings are intuitive because OPS takes into account driving time, service time, and resolution probabilities, whereas none of the greedy strategies consider driving time in their decision-making process. As a result, the greedy strategies perform poorly when only the driving time matters. However, when driving time becomes negligible due to long service times, minimizing the number of visited options becomes crucial, which binary-based search strategies excel at.

5.6.3 Uninformed optimal policy search

Lastly, Table 5.6.3 presents the outcome of running Algorithm 5.1 with all resolution probabilities, π_i^m, set to be equal. This approach simulates the case of lacking insights into the underlying probabilities, effectively establishing a set

Table 5.6.2: Averaged results for all 640 instances for each of the four spatial configurations considered using the three greedy methods *and* the optimal method. Here, LS is the linear search method, BSNO is the binary search by number of options method, BSCP is the binary search by cumulated probability method, and OPS is the optimal policy search approach. The last column reports the average of the total expected time found using optimal policies. The right side of the table shows how much worse each greedy method performed compared to the optimal approach, and the left side reports the number of instances out of all 640, where the given method found the best solution. Note that it is possible for multiple methods to find the same solution to the same instance.

c^s [min]	c^d [min]	# instances where best				% worse than best avg.			**Optimal** avg. total expected time [min]
		LS	BSNO	BSCP	OPS	LS	BSNO	BSCP	
Tree-like instances									
0	30	0	0	0	640	80.5%	15.6%	11.8%	92.52
5	30	0	2	2	640	78.3%	13.4%	7.9%	169.57
10	30	0	4	5	640	75.0%	10.9%	4.9%	242.39
15	30	0	4	13	640	74.0%	9.9%	3.7%	313.93
20	30	0	4	19	640	73.6%	9.5%	3.1%	384.86
50	30	0	4	52	640	73.6%	8.9%	2.2%	806.18
100	30	0	5	72	640	74.0%	8.7%	2.1%	1504.60
Circular instances									
0	30	0	1	1	640	109.7%	38.9%	31.7%	96.32
5	30	0	3	4	640	86.3%	21.0%	13.9%	173.89
10	30	0	4	7	640	80.1%	15.8%	8.7%	247.60
15	30	0	6	11	640	77.6%	13.5%	6.4%	319.82
20	30	0	6	13	640	76.3%	12.3%	5.1%	391.29
50	30	0	5	40	640	74.7%	10.0%	3.0%	814.00
100	30	0	4	60	640	74.7%	9.4%	2.4%	1512.46
Column-wise instances									
0	30	0	2	0	640	123.2%	42.6%	35.6%	126.45
5	30	0	2	1	640	97.4%	25.4%	18.5%	204.78
10	30	0	3	3	640	88.3%	19.2%	12.3%	279.47
15	30	0	3	7	640	83.8%	16.1%	9.2%	352.59
20	30	0	2	8	640	81.3%	14.3%	7.3%	424.85
50	30	0	5	18	640	76.6%	10.7%	3.8%	850.79
100	30	0	5	43	640	75.4%	9.6%	2.7%	1551.60
Random instances									
0	30	0	0	0	640	137.7%	46.4%	36.0%	155.96
5	30	0	0	0	640	110.4%	30.1%	21.0%	234.14
10	30	0	0	1	640	98.6%	23.1%	14.5%	309.46
15	30	0	0	2	640	92.2%	19.3%	11.0%	383.28
20	30	0	0	3	640	88.2%	16.9%	8.9%	456.17
50	30	0	2	15	640	79.8%	11.8%	4.4%	884.72
100	30	0	3	39	640	77.0%	10.1%	2.3%	1587.46

of *uninformed optimal policies*. Consequently, the cost matrices derived from this approach, reflecting the expected distances between tasks, are inherently inaccurate, yet they represent the best one can do without information about the resolution probabilities.

Solving the TRSP model (5.2)-(5.8) using the inaccurate cost matrices still yields a route that details in which order a technician should solve the set of tasks considered. The true expected cost of this route can then be computed using the true expected distances between tasks, which can readily be obtained by running Algorithm 5.1 using the true resolution probabilities, i.e. a regular informed OPS.

In Table 5.6.3 we see that the solutions found using an uninformed OPS are only slightly worse than than the solutions obtained using the informed OPS. The uninformed OPS performs best on the tree-like instances and worst on the random instances. These results are of course strongly tied to the underlying probability distribution, which, as explained in Section 5.5 were generated to be slightly balanced. More unbalanced resolution probabilities would likely be a disadvantage to the uninformed OPS approach.

We explore this hypothesis by conducting the same analysis using a different, unbalanced approach to simulating the probabilities. In this approach, the fault probability is highest for the option corresponding to the customer node, n_1^m, and halves for each subsequent option when moving away from the customer, connection by connection. Formally, the new probabilities for the options in the ordering sequence L^m, for any task $m \in M$, are given by: $[\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}]/(1-\frac{1}{2})$.

Table 5.6.4 shows the result of this new version of the analysis. As hypothesized, the uninformed OPS results are now worse than before when compared to the informed OPS—in some cases, the percentage difference has quadrupled. However, the uninformed OPS still gives great results. For the tree-like instances—which are the most similar to real telecommunication networks—the uninformed OPS is on average at most 2% worse than the informed OPS when considering all service times. Only in the random instances is there a significant gain in knowing the underlying probabilities, with an up to a 9% difference in average expected time between the uninformed and informed OPS.

5.7 Discussion

Although we presented the model and computational results for a base TRSP, our framework, which precomputes expected travel times between each pair of tasks, can be generalized to any TRSP variant without loss of generality by simply adjusting the expected travel times accordingly.

Some TRSP variants impose limited working hours for each technician. Given

Table 5.6.3: Averaged results for all 640 instances for each of the four spatial configurations considered using uninformed optimal policies, OPS $\setminus\pi$, and the informed optimal policy search method, OPS. The last column reports the average of the total expected time found using an informed OPS. The right side of the table shows how much worse the uninformed approach performed compared to informed approach, and the left side reports the number of instances out of all 640, where the given method found the best solution. Note that it is possible for both methods to find the same solution to the same instance.

c^s [min]	c^d [min]	# instances where best		% worse than best avg.	**Optimal** avg. total expected time [min]
		OPS $\setminus\pi$	OPS	OPS $\setminus\pi$	
Tree-like instances					
0	30	610	640	0.38%	92.52
5	30	393	640	0.44%	169.57
10	30	345	640	0.45%	242.39
15	30	321	640	0.46%	313.93
20	30	303	640	0.45%	384.86
50	30	253	640	0.32%	806.18
100	30	239	640	0.21%	1504.60
Circular instances					
0	30	430	640	0.42%	96.32
5	30	343	640	0.55%	173.89
10	30	287	640	0.57%	247.60
15	30	260	640	0.58%	319.82
20	30	241	640	0.57%	391.29
50	30	214	640	0.43%	814.00
100	30	196	640	0.27%	1512.46
Column-wise instances					
0	30	246	640	1.26%	126.45
5	30	205	640	1.17%	204.78
10	30	181	640	1.11%	279.47
15	30	157	640	1.01%	352.59
20	30	151	640	0.94%	424.85
50	30	132	640	0.67%	850.79
100	30	111	640	0.44%	1551.60
Random instances					
0	30	140	640	2.20%	155.96
5	30	124	640	2.00%	234.14
10	30	117	640	1.73%	309.46
15	30	108	640	1.62%	383.28
20	30	102	640	1.54%	456.17
50	30	76	640	1.09%	884.72
100	30	67	640	0.74%	1579.60

Table 5.6.4: Averaged results for all 640 instances for each of the four spatial configurations considered using uninformed optimal policies, OPS $\setminus\pi$, and the informed optimal policy search method, OPS. The results are based on unbalanced, halving probabilities as explained in Section 5.6.3. The last column reports the average of the total expected time found using an informed OPS. The right side of the table shows how much worse the uninformed approach performed compared to informed approach, and the left side reports the number of instances out of all 640, where the given method found the best solution. Note that it is possible for both methods to find the same solution to the same instance.

		# instances where best		% worse than best avg.	**Optimal** avg. total expected time [min]
c^s [min]	c^d [min]	OPS $\setminus\pi$	OPS	OPS $\setminus\pi$	
Tree-like instances					
0	30	610	640	2.07%	101.25
5	30	198	640	1.81%	165.47
10	30	185	640	1.46%	227.36
15	30	175	640	1.26%	288.83
20	30	159	640	1.11%	350.11
50	30	142	640	0.64%	716.46
100	30	130	640	0.38%	1325.21
Circular instances					
0	30	286	640	1.73%	90.39
5	30	209	640	1.57%	154.76
10	30	176	640	1.35%	216.83
15	30	161	640	1.21%	278.41
20	30	146	640	1.07%	339.76
50	30	119	640	0.62%	706.18
100	30	103	640	0.37%	1314.98
Column-wise instances					
0	30	121	640	4.37%	115.70
5	30	88	640	3.69%	180.62
10	30	70	640	3.06%	243.14
15	30	65	640	2.64%	305.06
20	30	68	640	2.30%	366.67
50	30	48	640	1.34%	733.98
100	30	39	640	0.81%	1343.67
Random instances					
0	30	69	640	8.91%	136.71
5	30	58	640	7.30%	201.64
10	30	44	640	6.08%	264.54
15	30	43	640	5.21%	326.74
20	30	37	640	4.64%	388.60
50	30	26	640	2.67%	756.61
100	30	26	640	1.61%	1366.82

the stochastic nature of the problem, various policies can be implemented to manage situations where realized working time exceeds the technician's limit.

In practice, it is common to re-optimize the TRSP after each task completion, updating the plan with the most recent information about progress. This would include the actual task completion time as well as the actual completion location, allowing for recalculation of the expected travel times from the completed task and onward.

Additionally, bounding the resolution time of each task can be achieved relatively easily by assuming that all task options must be visited. A conservative policy would prevent a technician from starting a task if the worst-case resolution time exceeds their available working hours.

In reality, the resolution probabilities associated with each task option are not the only stochastic elements of the problem. Here, we assumed a fixed service time for each option, while in practice, service time may vary across locations and technicians. Furthermore, the driving time between any two options is assumed to be known exactly, though in practice, factors such as traffic and parking conditions introduce variability into real driving times. Thus, an interesting avenue for future research would be to incorporate more stochastic elements into the problem formulation.

5.8 Conclusion

In this paper, we have introduced and solved a novel variant of the *Technician Routing and Scheduling Problem* (TRSP) for tasks with stochastic resolution locations. By modeling the customer subproblem as a Markovian process, we have shown that a recursive algorithm can find the optimal policy for resolving a task in $O(N^3)$ time, where N is the number of resolution options for a given task. This approach allows us to efficiently compute the expected time between any pair of tasks, which can be utilized to solve an ordinary deterministic TRSP using the resulting distance matrix.

Our computational experiments demonstrate significant time savings of approximately 70-80% when using optimal policies to resolve the set of tasks as opposed to the linear approach currently employed by our collaborating telecommunication company. We also demonstrated that our algorithm is effective even in an uninformed setting—where no information about the resolution probabilities is know—and still beats all the greedy strategies we proposed, some of which leverages the true resolution probabilities. These results highlight the potential for substantial operational improvements saving both time and resources.

Given the speed at which an optimal policy can be computed, our proposed algorithm can easily be implemented in an app that guides technicians in real

time. Moreover, the routing solution can be reoptimized on the go as new information comes in during a work day. This makes our results very applicable in practice. The algorithm can also readily be implemented to co-exist with existing routing software, as the set of optimal policies can be precomputed prior to any routing.

One of the key strengths of our approach lies in its general applicability. While the problem was formulated within the context of fault localization in cabled networks, the methodology is broadly applicable to other domains with similar binary search characteristics. Potential application areas include pipeline failures, pollution detection, inventory management, and other infrastructure-related problems where fault detection follows a sequential, dependent process.

In conclusion, this work fills an important gap in the study of stochastic network maintenance, contributing both theoretically and practically. By demonstrating its adaptability across different sectors, we open doors for further research and real-world integration, with the potential for long-term cost reductions and operational improvements in industries reliant on sequential infrastructure.

Acknowledgments

This project was supported by Innovation Fund Denmark, project number 0224-00055B, GREENFORCE.

Chapter 6

Paper V

A Consensus Fixing Heuristic to Workforce Investments in Field Service

Status: Paper is submitted for publication in *Computers and Operations Research*

Authors: Siv Sørensen[a], Clara Chini Nielsen[a], David Pisinger[a] and Joaquín Ignacio Fürstenheim[a]
[a] DTU Management, Akademivej, Building 358, DK 2800 Kgs. Lyngby

Keywords: Strategic workforce planning, scenario analysis, ALNS, out-of-sample testing, consensus fixing

Abstract

Workforce planning is a critical challenge across various sectors, encompassing strategic, tactical, and operational decision-making to optimise goals such as profit maximisation, greenhouse emission reduction, and customer satisfaction. This paper addresses the strategic workforce planning problem as investment planning in the context of field service in telecommunications, focusing on the *Technician Routing and Scheduling Problem* (TRSP).

To tackle these complexities, we propose a novel strategic framework inspired by a new methodology called *consensus fixing*. Our approach evaluates a set of scenarios (here, operational days) independently using a complete TRSP model, computing the value of each potential investment. A consensus scheme then fixes a single investment decision across all scenarios iteratively, one by one, until a stopping criterion is met. This methodology allows for parallelisation and maintains low computational complexity, integrating detailed operational considerations into strategic decision-making.

We apply our framework to real-life data from the Danish telecommunication company TDC-NET, utilising an *Adaptive Large Neighbourhood Search* (ANLS) to solve each scenario. Different consensus schemes are investigated, and the performance of the hired workforce is evaluated through out-of-sample testing and comparison to a novel *scenario-tailored* solution.

Due to its fast run-time, the methodology can equip decision-makers with a wide range of detailed investment solutions, including actual routes, that transparently demonstrate the trade-offs between a robust workforce and a cost-efficient one. The scenario-tailored solutions enhance this transparency by providing a benchmark for the best possible driving, hiring, and unserved task costs, helping to assess how effective a proposed workforce is. This provides decision-makers with a solid foundation for selecting a set of investments that balances customer satisfaction with operational costs.

Based on the results, we also highlight several other valuable managerial insights, such as demonstrating that a more detailed set of possible investments may not always lead to better results, and showing how the robustness of the investments chosen by our methodology can be verified through a post-investment sensitivity analysis.

6.1 Introduction

Workforce planning problems occur in almost all sectors and generally describe the problem of utilising a workforce optimally with respect to the goals of a company. Typical goals include maximising profits, reducing greenhouse emissions,

and maximising customer satisfaction. Workforce planning problems can be divided into three main categories related to the time horizon: strategic, tactical, and operational. The strategic deals with long-term decisions such as facility investments, the tactical deals with mid-term decisions such as area partitioning while the operational deals with short-term decisions such as day-to-day work schedules [9].

Strategic decisions often involve the highest level of uncertainty due to the sometimes infinite number of future scenarios a company could face. Strategic decisions are also typically both expensive and complex, and once a decision is made, it can be difficult to change. A vast part of the research literature considers simplified models of the future that focus on computational feasibility and rarely on the real-life implications of the various simplifications [10]. Despite the extensive study of operational-level workforce planning problems, problems focusing on longer time frames have received little attention. This gap highlights the need for comprehensive strategic workforce planning models that can address real-life complexities and uncertainties.

Field service is a dynamic business, as demands may fluctuate significantly from day to day, or even from hour to hour. Field service applications include maintenance operations and staff routing in sectors such as telecommunications, public utilities, and health care. In field service, strategic decisions include hiring and firing staff, upgrading and maintaining skills and equipment (such as vehicles), and opening, closing, or moving centralised hubs and depots. In this paper, we consider the strategic workforce planning problem of hiring a field service workforce to support and maintain telecommunication infrastructure.

The operational variant of the problem, the *Technician Routing and Scheduling Problem* (TRSP), first introduced by Tsang and Voudouris in relation to British Telecom [152], has been studied extensively. The TRSP is a known NP-hard problem where a set of technicians must solve a set of tasks. Each technician possesses a set of skills, and each task requires one or more of these skills. Tasks are located in different geographical locations, and technicians have to be assigned a set of tasks as well as a route respecting possible time windows. The dynamic variant of the problem, in which a set of dynamic tasks, initially unknown, are revealed throughout the workday, has also frequently been studied [115, 117]. These operational factors make strategic planning even more complex, since it is not only the number of technician hires that must be decided, but also the skills they should possess and the geographical areas they should be based in.

To address these complexities, making strategic decisions that are robust towards operational fluctuations, we will adopt a framework inspired by the recently proposed strategic approach, *consensus fixing*, that can be seen as a kind of scenario analysis algorithm. Consensus fixing was originally introduced [118] in a simpler two-stage prize-collecting *Travelling Salesman Problem* (TSP) set-

ting with binary first-stage decisions. The framework lets us consider a set of scenarios, here a set of operational days, which are evaluated separately using a full-scale TRSP model. For each scenario, we compute the value of every investment from a set of available investments. Afterwards, we use a consensus scheme to fix a single investment decision across all scenarios. This procedure is repeated until a given stopping criterion is reached. This methodology differs from traditional approaches by integrating detailed operational considerations into strategic decision-making, thereby ensuring that the selected strategic plans are robust against daily operational fluctuations.

We consider real-life data from a Danish telecommunication company, TDC-NET. Each scenario is solved heuristically using Adaptive Large Neighbourhood Search (ANLS) where we can hire from a pool of technicians with different skills and home locations. We investigate several consensus schemes and evaluate the performance of the workforce hired by the proposed framework by doing out-of-sample testing as well as a comparison to *scenario-tailored* solutions. This comprehensive approach aims to bridge the gap between strategic planning and operational execution, providing a more realistic and practical solution to workforce planning in complex environments.

The **main contributions** of this paper are:

- We demonstrate that a consensus-fixing-inspired framework can be applied to large-scale, complex strategic investment problems in technician routing and scheduling.
- The proposed framework's parameters and decision criteria are flexible and can be adapted to various investment problems involving a discrete set of possible investments.
- The proposed framework utilises the output from a separate routing and scheduling procedure as a black box, allowing for easy implementation in other use cases with different constraints or pre-existing solution methodologies.
- We demonstrate that the framework is parallelisable with respect to both the input scenarios and the set of investments, allowing for an efficient evaluation of long-term investment effects over many scenarios while considering a potentially large set of investments.
- For each scenario, we estimate the best possible total cost by identifying a *scenario-tailored* workforce and let this cost estimation serve as a benchmark for the consensus-fixing framework.
- We present promising results using real-life data from the telecommunications industry.

This paper is **organised** as follows: Section 6.2 presents previous work on investment planning for TRSP that forms the basis for our model as well as other relevant literature. Section 6.3 introduces the investment problem and

its objective, and Section 6.4 presents a Mixed Integer Programming (MIP) formulation of the TRSP. The ALNS algorithm used to solve the TRSP in our experiments is briefly outlined in Section 6.5.

The consensus fixing framework and the parameters used are presented in Section 6.6. Section 6.7 describes the real-life scenarios used in the experiments. In Section 6.8 we present and discuss computational results and validate the quality of the solutions found. The paper is concluded in Section 6.9 with a summary of the obtained results and ideas for future work.

6.2 Literature

The first formulation for the Technician Routing and Scheduling Problem (TRSP) was introduced by Tsang and Voudouris [152] in 1997. The problem saw a surge of interest in 2007, when it was proposed as part of the French Operations Research Society (ROADEF) challenge [30]. The challenge included a new set of benchmark instances that since has been featured in numerous papers [31, 46, 42]. Since its introduction, the TRSP has gained significant attention due to its wide applicability. Most research efforts have concentrated on employing metaheuristic approaches to find near-optimal solutions [90, 116]. Despite these advances, achieving optimal solutions remains challenging, with only a few studies managing to do so for smaller problem instances involving up to 45 customers [87, 89]. In contrast, heuristic methods have been applied to larger instances around 500 tasks, sometimes addressing multi-day periods. For comprehensive overviews of the literature, readers are referred to surveys by Chen et al. [24] and Mathlouthi et al. [90], which outline the progress in this domain up until 2017 and 2021, respectively.

The majority of TRSP research has been driven by practical applications, often incorporating additional constraints to reflect real-world scenarios. For example, Pillac et al. [115] explored the dynamic TRSP, where new tasks can arise during the service period. They proposed an ALNS algorithm capable of solving instances with up to 200 customers and 25 technicians.

Mathlouthi et al. [90] addressed a variant of the problem involving spare parts, where technicians may need to visit a depot to retrieve necessary parts before proceeding to the task location. They used a Tabu Search method integrated with adaptive memory, which allows for the storage and reuse of elite solutions during new searches. This approach has also been applied successfully to the vehicle routing problem [55, 163].

Another notable variant of the TRSP considers the variability in service times based on the technicians' experience levels, recognising that more experienced technicians complete tasks quicker. Chen et al. [24] tackled this challenge using a record-to-record travel heuristic approach with a rolling horizon, showing that

explicitly modelling workforce heterogeneity leads to better solutions compared to assuming homogeneous, static productivity.

Despite these advancements, the strategic aspects of investment planning in TRSP, such as determining the optimal number and location of technicians or the necessary skill sets, remain under-explored. One significant work on the strategic planning of TRSP is by Gamst and Pisinger [54]. Their study focuses on strategic decisions like the number of technicians to employ and which qualifications to upgrade. Their proposed solution method employs a matheuristic based on column generation. However, this model assumes a complete set of technicians is already available and addresses only skill upgrades and task digitisation. Moreover, their investment decisions are based on a highly simplified model of the TRSP. For simpler vehicle routing problems that do not involve technician skills or depot locations, Cortés et al. [34] propose a method to optimise fleet sizes by developing policies for a one-year planning horizon. Their approach involves conducting a detailed demand analysis over the year and using a branch-and-price method to simulate various scenarios, with the objective of minimising operational costs while maintaining high service quality.

In recent years, the scope of the TRSP and related problems, such as home care scheduling, has expanded to include broader workforce scheduling issues. These problems are increasingly examined in an integrated manner, facilitating cross-learning between problem types and highlighting similarities in solution methods, e.g. similarities of MIP constraints or heuristic neighbourhoods, and problem characteristics such as the need for specific tools or machines, customer preferences and teams working together to tackle tasks. Noteworthy surveys on this broader perspective include works by Bruecker and Pereira [10, 112].

A number of recent papers have studied *tactical planning* for the dynamic TRSP. By scheduling some tasks over longer time horizons, significant savings in driving distance can be achieved by clustering nearby tasks on specific weekdays. Nielsen and Pisinger [101] investigated this problem in a dynamic TRSP context, while a subsequent paper by the same authors extended the tactical planning approach to consider skills and multiple home depots for technicians [102].

Song et al. [147] explored a two-stage grocery delivery and unpacking problem. Their work considered the requirement that each subscription customer must be served by the same driver, in contrast to on-demand customers. In the first stage, they assign subscription customers to drivers, and in the second stage, they solve a team-orienteering problem to maximise profits by serving additional on-demand customers. This second stage problem is solved via an extension of a standard labeling algorithm for solving the Vehicle Routing Problem with Time Windows (VRPTW).

The strategic planning problem for the TRSP is related to the *Capacity Expansion Problem* (CEP) in energy planning. The CEP seeks to determine in-

vestments in energy-producing units that meet demand while minimising expected investment and operational costs [86]. This is based on the knowledge of forecasted demand patterns, assuming that the costs of expansions are known. Likewise, the strategic planning problem for the TRSP uses historical data to approximate demand and makes decisions on investments based on the costs of each investment.

6.3 The investment problem

Given a set of scenarios S that each represents a possible realisation of tasks that need to be handled during a day, we define our problem as hiring the set of technicians $\mathcal{T}$ that minimises the expected cost of future operations.

Let each scenario $s \in S$ be the set of tasks T_s that must be served in a single day. Each task has a duration, a location, a time window and required skills. Moreover, let the set of scenarios be split into two groups; a set of historic scenarios S^H (training data) and a set of future scenarios S^F (test data). The optimal workforce must be chosen based on the set of historic scenarios but will be evaluated on the set of future scenarios.

Technicians can be hired from a set of predefined technicians, denoted by the investment set $\mathcal{T}^I$. We define two main types of technicians who work on different types of telecommunication infrastructures (COAX and fiber). Two variations of skill levels exist for each technician type: one experienced technician who is equipped with all skills and one entry-level technician who possesses a predefined set of base skills. Lastly, every technician $t \in \mathcal{T}$ is associated with a home location δ_t that serves as their starting and ending point in all scenarios. We consider three possible home locations, which give a total of $|\mathcal{T}^I| = 12$ unique types of technician hires. Each of the 12 technician types can be hired as many times as needed when constructing a workforce.

6.3.1 Objective

The total cost of operations includes both fixed investment costs and variable operational costs and consists of three contributions: staffing costs, routing costs, and the cost of unserved tasks. Future costs of operations are obtained by computing these costs in the set of unseen future scenarios.

A technician is associated with a monthly cost of c_t, and assuming an average month consists of d work days, the cost of a technician in each scenario is c_t/d.

Routing costs are calculated by solving a TRSP model using the hired set of technicians. The travel time of the resultant routes is converted into costs by assuming a fixed monetary rate per travel time, c^r, that covers both vehicle and fuel costs.

When working with real-life data, it cannot be assumed that it is always feasible to solve all tasks in a scenario. Moreover, without adding a penalty for leaving a task unserved, the solution where no tasks are handled would be optimal. We therefore assume a fixed monetary penalty, c^p, that, when multiplied by the duration of a task, equates to the cost of leaving the task unserved.

6.4 The technician routing and scheduling problem

In the following, we formally describe the daily technician routing and scheduling problem (TRSP), and in the next section, we describe the heuristic solution method. The formulation of the TRSP is a simplified version of [103] where we have left out work balance constraints and changed the objective to accommodate the investment and operational costs. For simplicity, we have left out the scenario index s in the description below, but for the entirety of this paper, the TRSP model is always solved for a single scenario at a time.

Let $\mathcal{T}$ be the set of technicians. Each technician $t \in \mathcal{T}$ has a set of skills $K_t \subseteq K$, work hours $[a'_t, b'_t]$, and a home location δ_t. A route for a technician t should start and end in the home location δ_t. Moreover, let T be the set of tasks that need to be served. Each task $i \in T$ has a duration f_i, a time window $[a_i, b_i]$ within which it must be started, and a skill requirement. The binary parameter σ_i^t denotes whether or not technician t has the skill required to solve task i. The driving distance in minutes from tasks i to j is denoted d_{ij}.

Let the binary variable x_{ij}^t indicate whether technician t drives directly from task i to j. Binary variable y_i indicates whether task i is unserved, while the continuous variable z_i^t denotes the arrival time of technician t at task i.

This leads to the following MIP formulation of the TRSP:

$$\min \quad \sum_{t\in\mathcal{T}} \frac{c_t}{d} + \sum_{t\in\mathcal{T}}\sum_{i\in T}\sum_{j\in T} c^r d_{ij} x_{ij}^t + \sum_{i\in T} c^p f_i y_i \tag{6.1}$$

$$\text{s.t.} \quad \sum_{t\in\mathcal{T}}\sum_{j\in T} x_{ij}^t \, \sigma_i^t + y_i \geq 1 \qquad i \in T \tag{6.2}$$

$$\sum_{i\in T} x_{\delta i}^t \leq 1 \qquad t \in \mathcal{T} \tag{6.3}$$

$$\sum_{j\in T} x_{ij}^t - \sum_{j\in T} x_{ji}^t = 0 \qquad t \in \mathcal{T}, i \in T \tag{6.4}$$

$$\sum_{i\in T} x_{i\delta}^t \leq 1 \qquad t \in \mathcal{T} \tag{6.5}$$

$$x_{ij}^t = 1 \implies z_i^t + f_i + c_{ij}^t \leq z_j^t \qquad t \in \mathcal{T}, i, j \in T \qquad (6.6)$$

$$a_i \leq z_i^t \leq b_i \qquad t \in \mathcal{T}, i \in T \qquad (6.7)$$

$$a'_t \leq z_i^t \leq b'_t - f_i - c_{i\delta}^t \qquad t \in \mathcal{T}, i \in T \qquad (6.8)$$

$$x_{ij}^t \in \{0, 1\} \qquad t \in \mathcal{T}, i, j \in T \qquad (6.9)$$

$$y_i \in \{0, 1\} \qquad i \in T \qquad (6.10)$$

$$z_i^t \geq 0 \qquad t \in \mathcal{T}, i \in T \qquad (6.11)$$

The objective function (6.1) corresponds to the description in Section 6.3.1 and minimises all fixed and variable costs, i.e., staffing costs, routing costs, and the cost of unserved tasks. The first set of constraints (6.2) ensures that a task $i \in T$ is either served by a technician with the proper skills or marked as omitted. The following three constraint sets (6.3)–(6.5) state that a technician $t \in \mathcal{T}$ can leave the home depot δ_t at most once, must respect flow conservation, and return to the depot δ_t at most once. Constraint set (6.6) keeps track of start times z_i^t at each visited task i for each technician t. Constraints (6.7)–(6.8) ensure that task start times z_i^t are within the task's and the technician's time windows. Finally, Constraints (6.9)–(6.11) define the domain of the variables.

6.5 ALNS algorithm for solving TRSP

In order to solve the TRSP problem defined in the previous section, we use the *Adaptive Large Neighbourhood Search* (ALNS) algorithm from Nielsen and Pisinger [103]. The ALNS algorithm [133] makes use of a set of efficient destroy Ω^- and repair Ω^+ operators. In each iteration, a destroy method $\mathbf{d} \in \Omega^-$ and a repair method $\mathbf{r} \in \Omega^+$ is selected using a roulette wheel selection with probabilities ρ^- and ρ^+. After the destroy and repair operation, the resulting solution is evaluated, and if specific requirements are met, it becomes the new current solution. Furthermore, the ALNS framework monitors how well the destroy and repair operators perform and assign them performance scores ρ^- and ρ^+ based on their success. In this way, the algorithm learns which operators perform best and can choose the well-performing operators more frequently. The ALNS algorithm is outlined in Algorithm 6.1.

The acceptance criteria used in our implementation is a simple *Simulated Annealing* (SA) acceptance, in which improved solutions are always accepted, while worsened solutions are accepted with a probability that depends on the change in the objective function.

We use the same neighbourhoods as in Nielsen and Pisinger [103], meaning that we have a total of six destroy neighbourhoods and three repair neighbourhoods. The destroy neighbourhoods are: *Random destroy*, *Route destroy*, *Critical destroy*, *Shaw destroy*, *Type destroy* and *Skill destroy*. The repair neighbourhoods

Algorithm 6.1 Adaptive Large Neighborhood Search (ALNS)

Input: x a feasible solution
Output: x_b the best found solution
$x_b \leftarrow x$; $\rho^- \leftarrow (1, \ldots, 1)$; $\rho^+ \leftarrow (1, \ldots, 1)$
while *stop criterion not met* **do**
    select destroy and repair methods $\mathbf{d} \in \Omega^-$ and $\mathbf{r} \in \Omega^+$ using ρ^- and ρ^+
    $x_t \leftarrow \mathbf{r}(\mathbf{d}(x))$
    if $z(x_t) > z(x_b)$ **then**
        $x_b \leftarrow x_t$
    if accept(x_t, x) **then**
        $x \leftarrow x_t$
    update ρ^- and ρ^+
return x_b

are: *Greedy repair*, *Regret repair*, and *Random repair*. We refer the reader to [103] and [54] for a detailed description of the neighbourhoods.

6.6 Investment methodology

In this section we first describe the consensus fixing framework and adapt it to our setting. Next, we present the detailed investment decision parameters for the studied real-life problem. Finally, we compare the investment model with previous work.

6.6.1 The consensus framework

Consensus fixing was introduced by Pisinger [118] as a general heuristic for a two-stage stochastic optimisation problem. The framework relies on a set of scenarios to describe the outcome of the stochastic distribution for the second-stage problem.

The main idea of consensus fixing is to solve the optimisation problem corresponding to the second stage independently for each scenario, using an exact or heuristic method. Then, the scenarios try to reach a consensus about fixing a single first-stage decision using some greedy or heuristic consensus strategy. The process of alternating between solving the second-stage problem and variable fixing is repeated until all first-stage variables have been fixed or some maximum number of investments is reached. Since all second-stage problems are independent, the framework lends itself well to a parallel implementation.

Consensus fixing can be characterised as a scenario analysis algorithm (SAA) [141], but as opposed to previous SAA algorithms that are based on an a-

posteriori analysis, consensus fixing uses SAA as an integrated part of the solution process.

Algorithm 6.2 Pseudo-code for consensus-fixing heuristic. Notice that line 3 through 4 can be executed in parallel for each combination of scenario $s \in S$ and investment $t \in \mathcal{T}^I$.

1: Let the set of fixed first-stage investments be $\mathcal{T} := \emptyset$
2: **while** stop criterion not met **do**
3: **for** all scenarios $s \in S$ **do**
4: For all possible investments $t \in \mathcal{T}^I$, solve TRSP model (6.1)–(6.11) with workforce $\mathcal{T} \cup \{t\}$
5: Calculate Π_t according to (6.13) for all investments $t \in \mathcal{T}^I$
6: Select the investment t^* with the largest polarity Π_t
7: Set $\mathcal{T} := \mathcal{T} \cup \{t^*\}$

Assume that we heuristically (or if possible, optimally) have solved the TRSP for all possible i^{th} investments $t \in \mathcal{T}^I$ for all scenarios $s \in S$, meaning that we have the objective values z^i_{st}.

For each scenario $s \in S$ and each possible investment $t \in \mathcal{T}^I$ we now calculate the *scores* δ_{st} as the change in objective value when adding technician $t \in \mathcal{T}^I$ to the workforce $\mathcal{T}$ in iteration i:

$$\delta_{st} = z^{i-1}_{st} - z^i_{st} \tag{6.12}$$

Furthermore, for each scenario $s \in S$ we have a *scaling factor*, f_s, that express how much weight we should give to each scenario, where we can use various strategies for weighting the scenarios.

In each iteration, we calculate the *polarity* of investing in each possible technician $t \in \mathcal{T}^I$ as:

$$\Pi_t = \sum_{s \in S} f_s \delta_{st} \tag{6.13}$$

The technician hire with the largest value of Π_t, corresponding to the largest (scaled) reduction in cost, is then added to the set of fixed first-stage investments constituting the current workforce $\mathcal{T}$.

Algorithm 6.2 outlines the proposed consensus-fixing-inspired framework. A parallelised implementation runs in ($I \cdot$ `TRSP_time_limit`) when the stopping criterion in line 2 is chosen as I iterations which correspond to making I investments.

The main difference between our framework and the framework proposed by Pisinger [118] is that Pisinger in line 4 of Algorithm 6.2 only solves the TSP

model once for each scenario $s \in S$, and then *approximates* the *scores*, δ_{st}, of all possible investments based on only those $|S|$ solutions.

Pisinger [118] also proposed a number of different scaling functions f_s. We have chosen to consider a total of five of these, where three are included in the paper, and the remaining two are given in Section S2. First, for each investment $t \in \mathcal{T}^I$, we rank the scenarios $s \in S$ according to their scores δ_{st}. Hence, the first scenario has the best change in objective value, and the last scenario has the worst change in objective value when hiring technician t. We then propose the following scaling factors (hereinafter referred to as consensus-fixing strategies):

Uniform scaling (`uniform`)
: In this case, all scenarios are weighted equally. Hence, we have $f_t^u = (1, \ldots, 1)$.

Best performing scenario (`elitarian`)
: This is the elitarian approach, where we select the single investment that performs the best across all scenarios. Hence, we have $f_t^u = (1, 0, \ldots, 0)$.

Worst performing scenario (`proletarian`)
: This is a robust approach, where we select the best-performing technician in the single worst-performing scenario. Hence, we have $f_t^u = (0, \ldots, 0, 1)$.

Apart from the consensus-fixing strategies described here, two more strategies have been studied, namely *Elitarian linear scaling* and *Robust linear scaling*. Both strategies are described, and their results are given in Section S2.

6.6.2 Investment decision parameters

The strategic investment framework utilises several parameters. The values of these parameters have been carefully chosen based on industry data and a review of relevant literature.

Hiring Costs The monthly cost, including overheads, of a technician, denoted by c_t, is set at €5000 for a base-level technician, while an experienced technician incurs an additional 10% cost, amounting to €5500 per month. Assuming a month consists of $d = 20$ workdays, the per-day cost of a technician for any given scenario can be calculated as c_t/d, which equals €250 for a base-level technician and €275 for an experienced technician.

Driving costs The cost, c^r, is set at €0.5 per minute of driving. This cost reflects fuel prices, vehicle maintenance, and depreciation costs. This cost is chosen based on an extensive 2024 analysis of car ownership costs in Denmark [43] as well as the transport allowance rates [144] issued by the Danish government.

Unserved tasks penalty As tasks vary in duration, the penalty for leaving a task unresolved is a rate, c^p, that, when multiplied by the task duration f_i, gives the penalty for leaving task $i \in T$ unserved. We set c^p at €3 per minute.

While TDC-NET aims to serve all tasks within normal working hours, in practice, overtime is sometimes needed to do so. The penalty, c^p, is chosen to be high such that even for shorter tasks with a duration of, e.g., 15 minutes, up to 90 minutes of additional driving is still cheaper than leaving the task unresolved.

Driving time The driving time between any two locations in the scenario is estimated based on the as-the-crow-flies distance between the points. This distance is scaled by a factor of 1.2 to approximate the actual road distance. Furthermore, an average driving speed of 60 km/h is assumed, allowing the conversion of distances into corresponding travel times. The resulting travel times are considered fixed and do not account for variations due to traffic conditions at different times of the day.

6.6.3 Comparison with previous studies

Not much research has been applied to the strategic aspect of TRSP, and to date, the only significant works on the strategic planning of TRSP is by Gamst and Pisinger [54] and Cortés et al. [34].

The investment model developed by Gamst and Pisinger aims to minimise the combined operational (OPEX) and capital expenditure (CAPEX) costs for the TRSP. Their model simplifies the TRSP by using a facility location problem to assign tasks to technicians instead of generating actual routes. Investment decisions are made through a scenario-building approach, which strategically determines the optimal technician fleet, their skills, working hours, and task digitisation. This approach allows for detailed what-if analyses, evaluating various combinations of investment decisions, such as overtime, task digitisation, and skill upgrades, which are integrated into the objective function and converted into time equivalents for comparability.

Column generation is used to solve the assignment problem, and the model's performance is validated on both literature-based and real-world data. To assess the accuracy of their approximations, Gamst and Pisinger compare the results from their approximation method with those obtained by heuristically solving the full problem using an ALNS, both before and after investment decisions. After comparing these outcomes, they evaluate whether the operational savings justify the investment costs.

Despite its strengths, the approach taken by Gamst and Pisinger has certain limitations. While their method provides a holistic tool for strategic decision-making in technician routing, it relies on route approximations that may overlook specific details. Moreover, their model is computationally demanding and does not lend itself naturally to a parallel implementation, unlike our approach.

Overall, the approach and the problem presented by Gamst and Pisinger differ significantly from ours, making direct comparisons challenging. While Gamst

and Pisinger upgrade an existing workforce, we start with an empty solution and generally focus on different types of investments. Additionally, we consider total costs to directly reflect the financial impact, whereas Gamst and Pisinger convert all costs into time equivalents, which emphasises time efficiency but might potentially obscure the financial perspective.

The simulation-based approach developed by Cortés et al. aims to optimise fleet design and technician dispatch for the Technician Dispatch Problem (TDP) under stochastic demand. Their method builds on the classic Vehicle Routing Problem with Soft Time Windows and accounts for uncertainty in service requests by simulating various fleet configurations and dispatch strategies to minimise operational costs while maintaining service quality.

The paper conducts a detailed analysis of the problem instances for the studied areas and simulates different days with varying fleet sizes, using a combination of constraint programming and a branch-and-price approach with an efficient branching strategy. The model is tested on a real-world case in the telecommunications industry, demonstrating its effectiveness in improving fleet management by addressing demand variability throughout the day, week, and year.

Despite offering valuable insights, the method has limitations. Its reliance on simulation for fleet design can overlook specific routing details, and its effectiveness is tied to the accuracy of demand forecasts. Furthermore, the model's computational requirements may limit its scalability to larger or more complex problems.

Cortés et al.'s approach differs from ours in several ways. They use exact routing with a single technician depot and no skill considerations, starting from an existing fleet size and reducing it step by step, whereas we begin with an empty fleet. Additionally, Cortés et al. examine the relationship between annual excess costs (due to fleet inefficiency) and service quality (measured by minutes of delay), while our approach focuses solely on minimising total costs without factoring in overtime, delays, or service quality.

6.7 Scenarios

The scenarios used for the computational experiments are anonymised real-life instances from the Danish telecommunication sector, specifically from the Danish Telecom Infrastructure Company TDC-NET obtained at the start of 2023. The training and test sets of scenarios, S^H and S^F, consist of 10 and 20 consecutive days of data, respectively. We chose to select the scenarios in this manner for simplicity and to capture the weekly pattern of fluctuations in task arrivals.

Selecting the most representative set of scenarios for a strategic long-term time

horizon study is a science in itself and is deemed outside the scope of this work. We refer the reader to [5] for a comprehensive review of the scenario planning literature and to [32] for a more recent *review of reviews* that bridges several research fields within scenario planning.

Each scenario represents a single workday and contains, on average, 85 tasks. Although all scenarios cover the same area, the number of tasks can vary between 55 and 114 tasks per day. This variation occurs because, in practice, the time window of some tasks (e.g., installation tasks) can span multiple days. As each scenario in this study represents a single workday, we have assumed that all tasks must be completed on the day their time window begins, without the option of postponement.

Task durations vary significantly: approximately 29% of tasks have a duration of 30 minutes or less, 44% range from 31 to 60 minutes, 18% range from 61 to 120 minutes, and 9% extend beyond 120 minutes. The total task duration for each scenario is detailed in Table 6.7.1, which also provides the exact distributions of tasks for each scenario.

Each technician in the workforce has a home location and a specialised skill set. All technicians work from 07:15 to 16:00, but they only perform tasks between 07:30 and 15:30, with the remaining time reserved for traveling to and from their first and last tasks, respectively. The distribution of skills is visualised in Figure 6.7.1. Approximately 90% of all tasks across all scenarios require a COAX technician, while the remaining 10% require a fiber technician. The set of base-level technicians can cover around 80% of all tasks and are assigned the skills most frequently requested by tasks.

Figure 6.7.2 depicts the ten scenarios used as test data, S^H, in the computational experiments. The locations of all tasks across all ten scenarios are plotted, along with the three locations from which a technician can be hired.

Scenarios	1	2	3	4	5	6	7	8	9	10
#Tasks	59	81	84	72	114	118	103	83	80	69
Avg. Duration	55.2	73.4	84.9	77.1	98.0	84.9	99.9	71.5	62.4	60.3

Scenarios	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
#Tasks	82	70	93	55	86	77	80	81	84	81	91	82	89	98	108	87	104	87	86	70
Avg. Duration	103.4	74.2	87.8	61.9	72.4	69.4	104.1	82.3	59.2	90.0	82.0	83.5	88.2	68.2	100.0	78.2	88.5	63.8	101.4	58.0

Table 6.7.1: Distribution of tasks as well as total task duration in hours for all scenarios. The first ten scenarios correspond to the historic test set, S^H, and the last 20 scenarios are the test set of future scenarios, S^F.

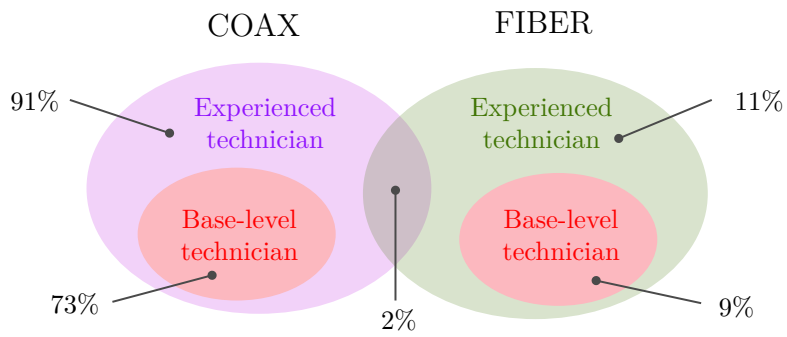


Figure 6.7.1: Skill distribution for the different types of technicians that can be hired. The values pictured refer to the percentage of all tasks across all scenarios S, that a technician has the skills to handle. The two base-level technician types can together handle roughly 80% of all tasks, whereas the two experienced technician types together can handle all tasks.

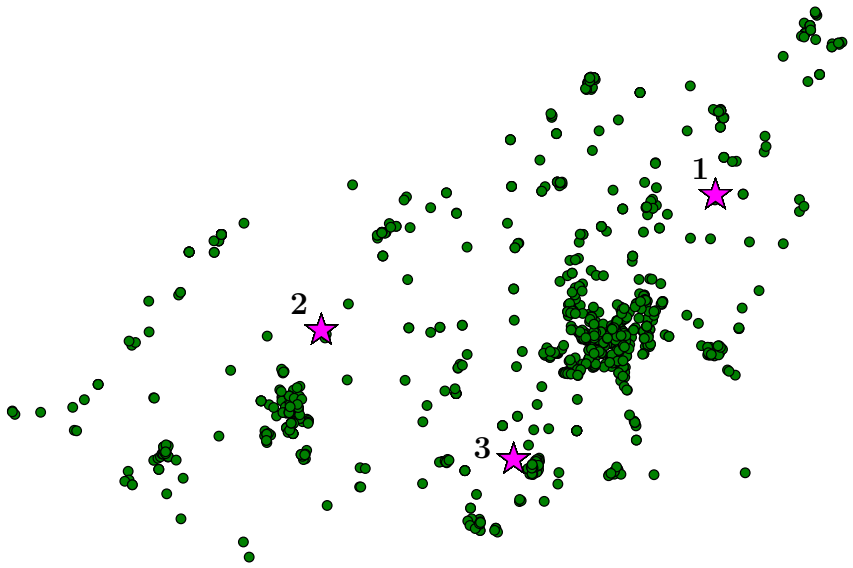


Figure 6.7.2: Visualisation of all tasks across the ten historic scenarios $s \in S^H$ used to compute the presented results. Green circles represent tasks, while magenta stars indicate the three possible technician start and end locations. The numbering of the set of possible technician locations corresponds to the results reported in Section 6.8.

6.8 Computational Experiments

All investment experiments are run on ten cores of an Intel Xeon Gold 6226R processor and are parallelised with respect to the ten historic scenarios S^H. Although the experiments could have been further parallelised with respect to the set of possible investments, this was not done due to limited access to hardware resources. We consider a stopping criterion of 30 technician investments in all analyses to ensure convergence, where the operational costs remain the same and only the investment costs increase. Using the aforementioned computational setup, 30 investment decisions based on ten scenarios take approximately four hours to run, and the ALNS runs with two restarts of 20 seconds per scenario.

The results of the computational experiments are split into two main parts:

1. The first part of the results (Section 6.8.1) is based on the ten historic scenarios S^H. Here, the three proposed consensus-fixing strategies are evaluated against each other. Based on this evaluation, the best strategy is used to hire the final workforce $\mathcal{T}^F$—including the decision of *how many* technicians should be part of this workforce.
2. The second part of the results (Section 6.8.2 and 6.8.3) is based on the 20 future out-of-sample scenarios S^F. Here, the performance of the final workforce $\mathcal{T}^F$ is compared to a *scenario-tailored* solution, which is introduced in more detail in Section 6.8.2. Moreover, a sensitivity study is carried out with respect to the technician hiring locations.

Additional results can be found in Appendix A and B.

6.8.1 Choosing the best consensus strategy

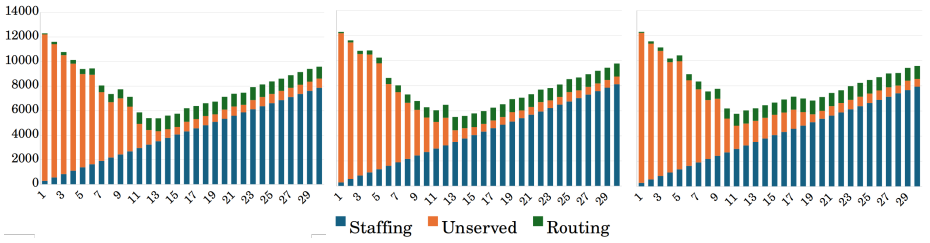
Figure 6.8.1 illustrates how the total cost, averaged over all ten historical scenarios S^H, evolves for each consensus strategy with each additional technician hire. Each bar in the plots represents the cost distribution among the three cost categories: staff costs, routing costs, and unserved task costs. As more technicians are hired, staffing costs (blue) steadily increase while the total cost of unserved tasks (orange) decreases. Eventually, each consensus approach reaches an optimal number of technician hires in terms of total cost. Beyond this optimal point, further hiring increases the total cost, as there is insufficient work to justify the costs of additional hires. Routing costs (green) continue to increase as more tasks are served, leading to the cost of unserved tasks decreasing since, generally speaking, serving more tasks requires more driving. Routing costs stabilise once enough technicians have been hired to cover nearly all tasks.

Since we are working with real data, there are certain tasks for which a combination of factors—such as the time window, the duration of the task, the location of

the task relative to the three hiring locations, or a combination thereof—makes it impossible to serve the task while respecting all constraints. As a result, the unserved task penalty never drops to zero. In real life, a company may resort to options like overtime or time window violations to ensure all tasks are served.

In the `uniform` and the `elitarian` consensus-fixing strategies, the lowest total cost is achieved with a workforce of 13 technicians, but the penalty for unserved tasks first reaches its minimal value after 15 technician hires and is therefore not considered converged until then. It is important to note that while routing and staffing costs are calculated using realistic parameters, the cost of unserved tasks has been artificially set to encourage the heuristic to prioritise serving all tasks.

The `proletarian` strategy follows a slightly different trend than the other strategies, where the convergence occurs in two different steps. Here, the lowest cost is achieved at a workforce of 11 technicians, and convergence seems to have been reached, but the cost of unserved tasks first reaches its minimal value after 19 technician hires, from where the real convergence starts. A reason for this behaviour is that the `proletarian` strategy only considers the scenario with the worst performance across all the different technicians. As the workforce gets closer to convergence, the worst-performing scenario is the one that converges the fastest and, thereby, cannot improve further by hiring new technicians. Therefore, a cheap technician will be hired instead of a technician who might be useful in the scenarios that are not yet converged, leading to only small improvements across the scenarios.



(a) `uniform` **(b)** `elitarian` **(c)** `proletarian`

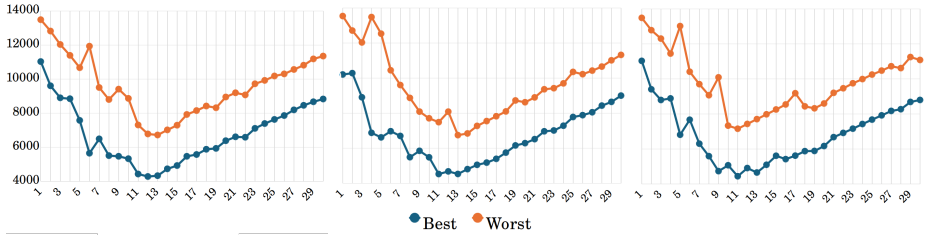
Figure 6.8.1: The total cost averaged over all scenarios $s \in S^H$ in each iteration and for each of the three proposed scaling strategies in the consensus-fixing framework. The y-axis describes the cost in €, and the x-axis lists the 30 iterations of the consensus-fixing framework corresponding to the 30 technician hires $t \in T^I$. Each bar shows the cost distribution among the three cost categories: staff costs, routing costs, and unserved task costs.

To further evaluate the effectiveness of each strategy, their robustness is studied. Figure 6.8.2 presents the range of objective values in each iteration of the framework, by showing the value of the worst and best objective (total cost)

across all scenarios S^H. Here, it is expected that the best objective value is the lowest for the `elitarian` strategy, as it only considers how an investment performs in the best scenario. Likewise, the `proletarian` strategy is expected to perform best in the worst case. However, not a big difference between the `elitarian` and the `proletarian` strategy is observed.

The two robustness curves in each of the three plots in Figure 6.8.2 mirror the trend in total cost averaged across all scenarios, as shown in Figure 6.8.1. Though they look very alike, the robustness trend for the `uniform` strategy most closely aligns with the average cost. Furthermore, the `uniform` robustness curves exhibit the smoothest behaviour, with minimal fluctuations in the worst and best objective values from one iteration to the next.

The scenarios differ in size, particularly in the number of tasks and total task duration. This variation explains the significant jumps in objective value—especially noticeable for the `proletarian` strategy—since, for example, the worst scenario in iteration i may be a large scenario, whereas in iteration $i+1$, it may be a small scenario since the consensus-fixing framework considers the difference in objective value across the investment iterations as given by equation (6.12).



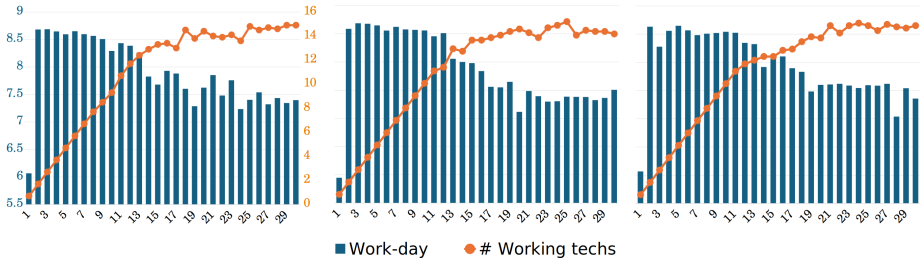
(a) `uniform` (b) `elitarian` (c) `proletarian`

Figure 6.8.2: The best and the worst total cost across all scenarios $s \in S^H$ in each iteration and for each of the three proposed scaling strategies in the consensus-fixing framework. The y-axis describes the total cost in € and the x-axis lists the 30 iterations of the consensus-fixing framework corresponding to the 30 technician hires $t \in T^I$. The total cost includes the three cost categories: staff costs, routing costs, and unserved task costs.

Lastly, we examine the workload of the hired workforce resulting from each consensus strategy. Figure 6.8.3 provides a twofold illustration: the orange line represents the average number of working technicians—those with at least one assigned task—in each iteration, while the blue bar chart in the background displays the average length of a workday for all active technicians. Once a sufficient number of technicians are hired to cover the workload, the `uniform` strategy assigns work to 14-15 technicians for the remaining iterations. The active workforce has an average workday of just under 7.5 hours. A similar

pattern is observed with the `elitarian` strategy, except that typically only 14 technicians are actively working. In the case of the `proletarian` strategy, we are back to 14-15 working technicians. However, the average workday is slightly longer at 7.7 hours. This indicates that the set of chosen technician investments serves the tasks in each scenario less efficiently compared to the other two strategies.

In practice, technicians also require lunch breaks, and planned schedules can be delayed due to uncertainties in task duration and driving times. Therefore, it is advantageous to schedule technicians for fewer hours than they are contracted for, as this gives flexibility to adapt to unforeseen changes.



(a) `uniform` (b) `elitarian` (c) `proletarian`

Figure 6.8.3: Each plot presents two key performance measures. The blue bar chart represents the average workday length in hours for an active technician, defined as a technician assigned to at least one task. The orange line indicates the total number of active technicians in each iteration of the framework. The three plots correspond to the average results for all historic scenarios $s \in S^H$ when using the three consensus-fixing strategies. The framework converges at approximately 14 active technicians, each with an average workday of around 7.5 hours.

Based on the above analysis of the results, we conclude that the `uniform` strategy performs best.

6.8.1.1 Selecting the final workforce $\mathcal{T}^F$

For the following experiments studying sensitivity and robustness, we choose to include the first 15 technician investments of the `uniform` strategy in the final workforce $\mathcal{T}^F$. We choose the first 15 technicians and not only the first 13 (corresponding to the minimal point in Figure 6.8.1a) for the following reasons:

- The penalty for unserved tasks decreases by an average of 25% when using 15 technicians as opposed to 13. This comes at the expense of a 7% increase in total cost, of which the majority is staffing costs.
- The average penalty for unserved tasks is minimal for a workforce of 15 technicians or more.

- Since serving as many tasks as possible is the foremost aim of the workforce, a larger workforce is likely to serve more tasks in future unseen scenarios. Here, 15 technicians appear to be the best trade-off between total costs and serving tasks.

The selected workforce consists of the set of technicians listed in Table 6.8.1. The workforce is predominantly composed of COAX technicians, which aligns with the distribution of skill requirements among the tasks, as depicted in Figure 6.7.1. Notably, the final workforce includes technicians exclusively from locations 1 and 2, with no hires from location 3, even though location 3 was similarly positioned relative to the task distribution, as shown in Figure 6.7.2.

This outcome suggests that the methodology employed is robust enough to produce reliable strategic insights without the need to consider a large number of hiring points. This simplicity is beneficial for long-term strategic analysis, as it reduces complexity without sacrificing the quality of the insights generated.

Lastly, the final workforce consists of mostly experienced technicians. An experienced technician costs €25 more per scenario than a base-level technician. This additional cost is equivalent to 50 minutes of driving time or the penalty incurred if a task with a duration of 8 minutes is not served. The preference for experienced technicians in the final solution reflects the strategic importance of minimising unserved tasks, where the broader skill set of experienced staff outweighs the cost increase.

Iteration #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Location (1:3)	2	1	1	2	1	2	2	1	2	2	1	2	1	1	2
Type (C/F)	C	C	C	C	C	C	C	C	C	C	F	C	F	C	F
Skill-level (B/E)	E	E	E	E	E	B	E	E	B	B	E	E	E	E	E

Table 6.8.1: The 15 technicians in the final workforce $\mathcal{T}^F$ and the order in which they were hired using the consensus-fixing strategy `uniform`. Home locations 1 through 3 are pictured in Figure 6.7.2, type C and F refer to COAX and fiber, respectively, and B refers to a base-level technician, while E refers to an experienced technician.

6.8.2 Solution validation

To validate the effectiveness of the final workforce $\mathcal{T}^F$, the TRSP model (6.1)–(6.11) is solved for 20 new out-of-sample future scenarios, S^F. The validation is done by estimating the best possible total cost of each scenario $s \in S^F$ and comparing this cost to the cost of $\mathcal{T}^F$. The cost estimation is computed by hiring a *scenario-tailored* (ST) workforce $\mathcal{T}_s^{ST}$ for each scenario and solving the TRSP model using this workforce.

The ST workforce for a scenario $s \in S^F$ is generated in a similar manner as

the consensus fixing framework, with one key difference: instead of requiring consensus across a set of scenarios for each hiring decision, only a single scenario determines the optimal technician hire in each iteration. In the consensus fixing heuristic (Algorithm 6.2), this corresponds to replacing line 3, "for all scenarios $s \in S$ do," with "for scenario s do," and then greedily choosing the best investment in each iteration.

Although this approach only approximates the best possible cost for each scenario $s \in S^F$—given that the scenarios are solved heuristically using ALNS and investments are made sequentially in a greedy manner—it should still yield a workforce that is better or as good as the final workforce for each scenario, as the final workforce is selected based on all the historical scenarios in S^H.

As the consensus-fixing analysis is carried out assuming technicians can be hired from three fixed locations, a sensitivity study is done to investigate the influence of the technician locations. This is done by randomly jittering each of the three technician locations inside a predefined area visualised in Figure 6.8.4. In reality, technicians start and end their daily routes from their own homes, so considering multiple start and end locations within some *hiring area* adds more transparency to the results and conclusions drawn.

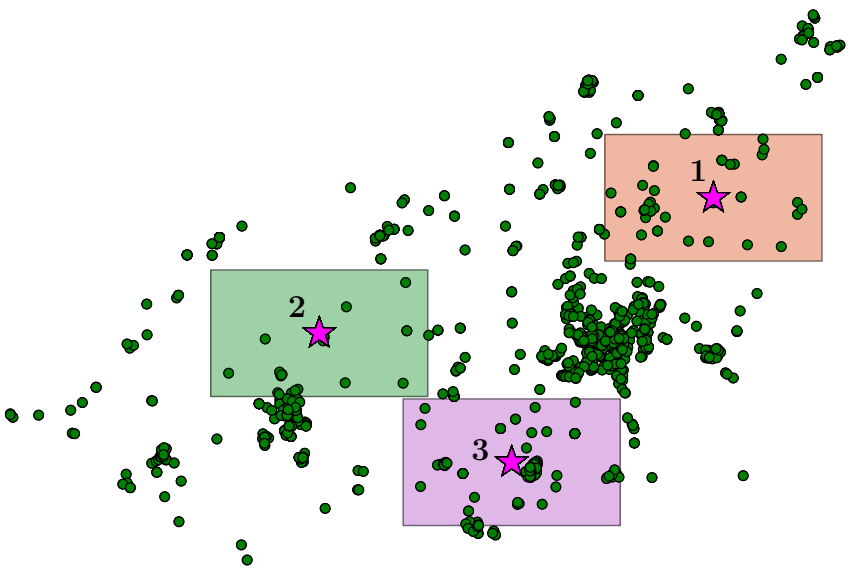


Figure 6.8.4: The above plot shows all tasks from all scenarios in the historic scenario set S^H. Green circles represent the tasks, while the magenta stars illustrate the three locations a technician can be hired from. The opaque square around each technician location illustrates the jitter regions used in the out-of-sample analysis to test how much the chosen hiring locations impact the results.

6.8.3 Out-of-sample performance of the final workforce

The performance of the selected workforce is evaluated in an out-of-sample test using the set of future scenarios S^F. We begin by obtaining a *scenario-tailored* (ST) solution for each scenario in the test set $s \in S^F$ using an ST workforce, as described in Section 6.8.2. The ST workforces vary in size, as the number of technician investments is selected to minimise the objective value. Table S1 in Appendix S1 lists the ST workforce identified for each scenario.

The objective values achieved when using the ST workforces, are presented in Table 6.8.2, alongside the objective values achieved by the final workforce $\mathcal{T}^F$. On average, the difference in objective value between the $\mathcal{T}^F$ solution and the ST solution is 30%. The ST workforces consist of 12.6 technicians on average, compared to the 15 technicians in the final workforce $\mathcal{T}^F$. While this might suggest that a smaller final workforce, such as 13 technicians, would have been more effective, choosing only the first 13 hires as the final workforce results in an average objective difference of 41% compared to the ST solutions. This is due to a significant increase in unserved task penalties, which far outweigh the savings in staffing costs. Moreover, the technicians in the ST workforces are tailored to their respective scenarios, while the technicians in the final workforce $\mathcal{T}^F$ are effective in some scenarios and less so in others.

Table 6.8.2 lists the out-of-sample scenarios in increasing order of total task duration. The size of the ST workforce is typically smaller than the final workforce in smaller scenarios. Since staffing costs constitute a significant portion of the total costs, part of the deviation in objective value between the ST solution and the $\mathcal{T}^F$ solution can be attributed to this. In the ten smallest scenarios with respect to the total task duration (except for scenario 15), the final workforce returns a solution that is at most 21% worse than the ST solution.

For larger scenarios, the final workforce $\mathcal{T}^F$ is, in some scenarios, too small to serve all tasks, resulting in large unserved task penalties. This is especially evident in scenarios 20, 11, and 17, where the average difference between the ST solution and the final workforce solution is 93%. In real life, if a company's strategy is to ensure that its workforce can fully serve even large scenarios, then a larger workforce must be hired, and the company must be willing to pay the additional staffing costs to guarantee the desired level of customer satisfaction.

Overall, the above results are promising, especially considering that the ST approach is conservative and is based on a workforce $\mathcal{T}_s^{ST}$ tailored to each individual scenario $s \in S^F$. The final workforce $\mathcal{T}^F$ was selected using a different set of scenarios, S^H, than those it was tested on and was moreover designed to meet the needs of multiple scenarios with varying demands. Despite this inherent disadvantage compared to the ST workforces, in 13 of the 20 out-of-sample scenarios, the difference in objective value is under 20%, with seven scenarios showing a difference of 10% or less.

Scenario #	30	19	14	28	24	16	15	12	26	18
ST # techs	9	10	10	12	13	13	13	12	12	14
ST obj. val. [€]	4944.0	4940.0	3637.5	4864.5	4885.5	4935.5	4442.0	4706.0	5107.0	5044.0
$\mathcal{T}^F$ obj. val. [€]	5807.5	5715.5	4396.5	5345.5	4942.5	4991.0	7975.0	5010.5	5551.0	5016.0
Difference [%]	17.5	15.7	20.9	9.9	1.2	11.4	79.6	6.5	8.7	-0.6

Scenario #	21	22	13	23	27	20	11	29	25	17
ST # techs	13	12	13	13	16	12	12	14	16	13
ST obj. val. [€]	5932.5	4063.0	5505.5	5826.5	5540.0	4713.0	6030.5	5873.5	7116.0	5554.5
$\mathcal{T}^F$ obj. val. [€]	6687.0	6618.0	6576.0	7553.5	5982.0	9053.5	11220.5	6693.0	7362.0	11115.0
Difference [%]	12.7	62.9	19.4	29.7	8.0	92.1	86.0	14.0	3.5	100.1

Table 6.8.2: Comparison of the objective value (total cost) obtained by the final workforce $\mathcal{T}^F$ with the ST solutions derived from the ST workforces listed in Table S1 in Appendix S1. The scenarios in the out-of-sample test set $s \in S^F$ are arranged in ascending order of total task duration, with scenario 30 representing the smallest and scenario 17 the largest. The first row indicates the number of technicians in each ST workforce $\mathcal{T}_s^{ST}$. The second and third rows present the objective values in euros for the ST and final workforces, respectively. The last row shows the percentage increase in the objective value of the final workforce compared to the ST solution.

6.8.3.1 The Influence of Hiring Locations

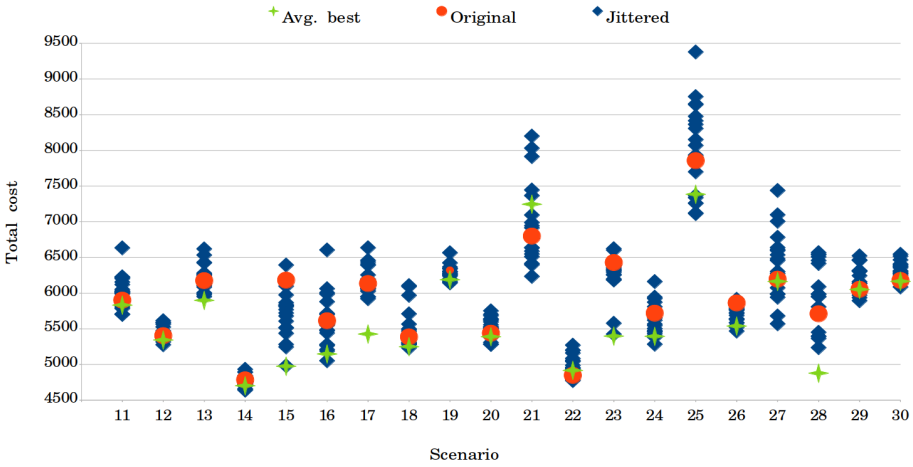


Figure 6.8.5: Scatter plot showing the impact on the objective values across the test set scenarios S^F, when jittering the technician locations at random 20 times within the regions pictured in Figure 6.8.4. Orange dots represent the original objective values without jitter, while blue diamonds indicate the objective values with jittering. The green stars highlight the set of jittered locations that resulted in the best objective value on average across all scenarios.

Figure 6.8.5 illustrates the impact of adding random jitter to the three potential hiring locations for technicians. In this analysis, all test scenarios $s \in S^F$ were re-solved 20 times using the same final workforce $\mathcal{T}^F$, but with the three technician locations randomly jittered in each re-run. The figure depicts these 20 solution values for each scenario as dots above each scenario. The original objective values, obtained without jittering, are shown as orange dots, while the jittered solution values are represented by blue diamonds. Additionally, the green star icons indicate the set of jittered locations that resulted in the best objective value on average across all 20 test scenarios.

As expected, jittering can lead to both improvements and deteriorations in the objective values, depending on the scenario and task locations. For most of the scenarios, the variation in objective value due to jittering is around €600, corresponding to roughly 10% of the objective value. The largest range of objective values is observed in larger scenarios, specifically scenarios 21, 25, and 27, where the objective value itself is generally higher due to the duration of unserved tasks. Given that the size of the final workforce $\mathcal{T}^F$ is insufficient for these scenarios with respect to serving all tasks, it is reasonable that the placement of technicians can have a considerable effect on the solution value.

Another source of uncertainty in Figure 6.8.5 is the optimality of the results. Each scenario is solved heuristically using an ALNS algorithm, which can lead to different technician routes even for the same scenario. This variation arises because the start and end locations of the technicians differ, causing the ALNS algorithm to explore different parts of the solution space.

Overall, the sensitivity analysis of the hiring locations depicted in Figure 6.8.5 demonstrates that the specific hiring locations considered in the consensus-fixing framework do not significantly influence the strategic insights as long as the potential investment decisions are approximately aligned with the task locations in the historic scenario set S^H. This is also a favourable outcome for real-life investments, where technicians are typically hired from a broader area rather than a specific location, especially when their home serves as the start and end point of their workday. In other field service applications where technicians operate out of one or more depots, our results generally suggest that decisions regarding the number and location of depots can be made using our framework without needing to consider an extensive number of investment options.

6.9 Conclusion

We have presented a novel framework for investment planning in a challenging transportation problem. Previous models, such as [54], relied on various approximations of technician routing to enhance tractability, whereas, in contrast, our framework bases the investments on actual technician routing solutions for a

set of scenarios. However, the investment choices still follow a greedy approach based on consensus fixing. Thus, the two approaches can be seen as complementary, each offering distinct benefits and drawbacks.

A significant advantage of the consensus-fixing method is its flexibility. Investments can be initiated from scratch or built upon an existing solution, such as a current workforce. The types of investments can be freely chosen, encompassing both fixed and variable-cost investments. Additionally, because the investment heuristic operates independently of the underlying solution methodology—in this case, an ALNS heuristic—investment planning can readily integrate with an existing routing and scheduling algorithm, thus accelerating strategic development.

While the framework is parallelisable with respect to both the number of scenarios and the number of investments, the combined growth of these parameters can be extensive in full-scale real-life investment problems, potentially requiring access to a substantial number of CPU cores to maintain manageable computation times. However, our results suggest that high levels of detail in the investment set are not always necessary to derive valuable strategic insights. Furthermore, we demonstrated how a simple post-analysis sensitivity study can evaluate the impact of choosing an investment set that only approximately represents the true set of investments.

Overall, the solutions generated by our framework are promising and sensible. The proposed investments that comprise the final workforce align well with task locations and skill requirements observed in the set of historic scenarios S^H. When comparing the performance of the final workforce against the conservative *scenario-specific* solutions, we observed an average cost difference of 30% in out-of-sample testing. We also noted that much of this cost difference arises from penalties associated with unserved tasks. While these penalties were roughly estimated and may not reflect real costs, they were necessary to ensure the methodology prioritised serving as many tasks as possible. The true managerial insights are derived not solely from the total cost but from a more holistic evaluation, including which tasks remain unserved in more challenging scenarios and whether additional technicians could reduce routing complexity and costs, aiding in optimal hiring decisions.

We have shown that our consensus-fixing heuristic can offer valuable strategic insights into large-scale investment problems. Promising directions for future research include investigating the number and selection of scenarios required to make reliable decisions. Given that the collaborating telecom company has a contract to serve a specified percentage of customers within a given time frame, a natural extension would be to study the chance-constrained version of the investment problem.

Additionally, the framework could be adapted to focus on upgrade investments

for a pre-existing workforce, such as transitioning parts of a fleet to electric vehicles or bikes, upgrading skills, or incorporating the possibility of overtime. Another avenue for future work involves exploring exact methods for smaller instances of the TRSP, to gain deeper insights into the true performance of the consensus-fixing methodology by moving away from heuristic-based scenario solutions.

6.10 Acknowledgements

Siv Sørensen, Clara Chini Nielsen and Joaquín Fürstenheim were supported by Innovation Fund Denmark, project number 0224-00055B. David Pisinger was funded by ERC-2022-ADG, project 101093188 DECIDE.

Appendices

S1 Scenario-tailored (ST) workforces

Table S1 lists the ST workforces $\mathcal{T}_s^{ST}$ that obtain the lowest total cost for each test scenario $s \in S^F$, when the consensus fixing framework is run considering a single scenario at a time. Although the ST workforces and their corresponding objective value serve as an estimate for the best possible solution, we cannot assess whether they actually are best. This is because a) the TRSP solutions that each technician hire is based on are not guaranteed to be optimal, and b) the heuristic consensus fixing framework hires technicians greedily one by one, which also does not guarantee optimality.

Scenario #	Iteration #:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
11	Location (1:3)	2	1	2	1	1	1	2	1	1	2	2	1				
	Type (C/F)	C	C	C	C	C	F	C	C	C	C	F	C				
	Skill-level (B/E)	B	B	E	E	E	E	E	E	E	B	E	E				
12	Location (1:3)	2	2	1	2	2	2	2	2	1	2	1	1				
	Type (C/F)	C	C	C	C	F	C	C	C	C	C	C	F				
	Skill-level (B/E)	E	E	B	E	E	E	E	E	E	E	B	E				
13	Location (1:3)	1	1	2	1	1	2	1	1	1	1	1	1	1			
	Type (C/F)	C	F	C	C	C	C	C	C	F	F	C	C	C			
	Skill-level (B/E)	B	E	E	E	E	E	B	B	E	E	E	B	E			
14	Location (1:3)	1	2	1	1	2	2	2	1	1	2						
	Type (C/F)	C	C	C	C	C	C	C	C	F	C						
	Skill-level (B/E)	B	E	E	E	E	E	E	B	E	E						
15	Location (1:3)	1	1	1	1	1	1	2	1	1	2	1	1	1			
	Type (C/F)	C	C	C	C	C	C	C	C	C	C	C	F	C			
	Skill-level (B/E)	E	E	E	E	E	E	B	B	B	B	E	E	E			
16	Location (1:3)	1	1	2	2	1	2	2	2	2	2	1	1	1			
	Type (C/F)	C	C	C	C	C	C	C	C	C	C	C	C	C			
	Skill-level (B/E)	B	B	E	E	E	E	E	E	B	B	B	B	B			
17	Location (1:3)	2	2	1	1	2	2	1	2	2	2	1	2	1			
	Type (C/F)	C	F	C	C	C	C	C	C	C	C	C	F	C			
	Skill-level (B/E)	E	E	B	B	E	B	E	B	B	B	B	E	E			
18	Location (1:3)	1	2	1	2	1	2	1	2	2	2	1	1	2	2		
	Type (C/F)	C	C	F	F	C	C	C	C	C	C	C	C	F	C		
	Skill-level (B/E)	B	E	E	B	B	B	B	B	B	E	E	E	B	B		
19	Location (1:3)	2	1	1	2	1	1	1	2	1	1						
	Type (C/F)	C	C	C	C	C	C	C	C	C	C						
	Skill-level (B/E)	B	E	E	E	E	E	B	B	B	E						
20	Location (1:3)	1	1	1	2	1	2	1	1	1	1	1	3				
	Type (C/F)	C	C	C	C	C	C	C	C	F	C	F	C				
	Skill-level (B/E)	B	E	E	E	E	E	E	E	E	E	E	B				
21	Location (1:3)	1	1	1	1	2	2	2	1	1	2	2	2	1			
	Type (C/F)	C	C	C	F	C	C	C	C	C	C	C	F	C			
	Skill-level (B/E)	E	B	E	E	B	E	B	B	E	E	E	E	B			
22	Location (1:3)	2	2	2	1	2	1	2	1	2	1	1	2				
	Type (C/F)	C	C	C	C	C	C	F	C	C	C	C	C				
	Skill-level (B/E)	B	E	B	B	E	E	E	B	E	B	E	B				
23	Location (1:3)	1	2	2	1	2	1	1	2	2	1	2	1	2			
	Type (C/F)	C	C	C	C	C	C	C	F	F	C	C	F	C			
	Skill-level (B/E)	E	E	E	E	E	E	E	E	E	E	E	E	E			
24	Location (1:3)	1	2	1	1	2	2	1	1	1	2	1	2	2			
	Type (C/F)	C	C	F	C	C	C	C	C	C	C	C	C	C			
	Skill-level (B/E)	B	B	E	B	B	E	E	B	E	B	E	E	E			
25	Location (1:3)	2	2	1	2	1	2	2	1	1	2	2	1	2	1	1	1
	Type (C/F)	C	C	F	C	C	C	F	F	C	C	C	C	C	C	C	C
	Skill-level (B/E)	B	E	E	E	E	E	E	E	E	B	E	B	B	E	E	E
26	Location (1:3)	2	2	1	2	1	1	2	1	1	2	1	2				
	Type (C/F)	C	C	C	C	C	C	C	C	C	C	C	C				
	Skill-level (B/E)	B	E	E	E	E	B	B	B	E	E	B	E				
27	Location (1:3)	2	2	1	1	1	2	1	2	2	2	1	2	2	2	1	2
	Type (C/F)	C	C	C	C	F	C	C	C	C	C	C	C	C	C	F	C
	Skill-level (B/E)	B	B	B	B	E	E	E	B	B	E	E	E	E	E	E	B
28	Location (1:3)	1	2	1	2	1	1	2	1	1	1	2	1				
	Type (C/F)	C	C	F	C	C	F	C	C	C	C	C	C				
	Skill-level (B/E)	B	E	E	B	B	E	E	B	B	B	B	B				
29	Location (1:3)	1	1	2	1	1	2	1	2	1	2	2	1	2	1		
	Type (C/F)	C	C	C	C	C	C	F	F	C	C	C	F	C	C		
	Skill-level (B/E)	B	E	E	E	E	E	E	E	E	E	E	E	E	E		
30	Location (1:3)	2	2	1	2	1	1	1	1	1							
	Type (C/F)	C	C	F	C	C	C	C	C	C							
	Skill-level (B/E)	B	E	E	B	E	E	B	B	B							

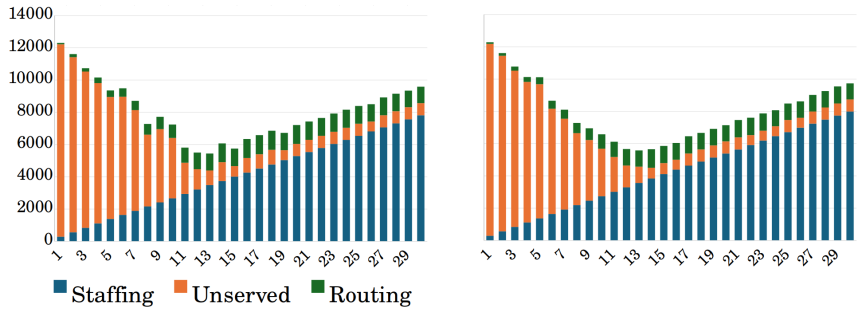
Table S1: The scenario-tailored set of technicians, $\mathcal{T}_s^{ST}$, for each scenario s in the test set S^F. Home locations 1 through 3 are pictured in Figure 6.7.2; type C and F refer to COAX and fiber, respectively, and B refers to a base-level technician, while E refers to an experienced technician.

S2 Linear scaling strategies

Let r_s be the rank of a scenario s with respect to its objective value, such that $r_s = 1$ for the best scenario, $r_s = 2$ for the second best scenario, etc.

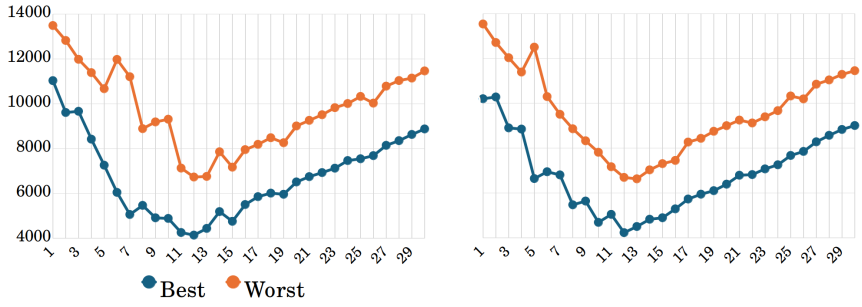
Elitarian linear scaling (`elinr`)
: This is the elitarian approach, where we select the best-performing technician with respect to assigning increasing weight to the best performing scenarios.. Hence, we have $f_t^u = (\frac{1}{r}, \ldots, \frac{1}{r})$.

Robust linear scaling (`rlinr`)
: This is a robust approach, where we select the best-performing technician with respect to assigning increasing weight to the worst-performing scenarios. Hence, we have $f_t^u = (\frac{1}{|S|-1+r}, \ldots, \frac{1}{r})$.



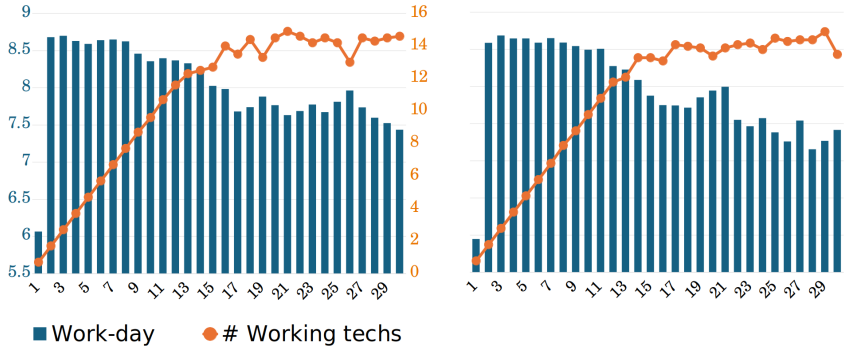
(a) `elinr` **(b)** `rlinr`

Figure S1: The total cost averaged over all scenarios $s \in S^H$ in each iteration and for each of the three proposed scaling strategies in the consensus-fixing framework. The y-axis describes the cost in €, and the x-axis lists the 30 iterations of the consensus-fixing framework corresponding to the 30 technician hires $t \in T^I$. Each bar shows the cost distribution among the three cost categories: staff costs, routing costs, and unserved task costs.



(a) `elinr` (b) `rlinr`

Figure S2: The best and the worst total cost across all scenarios $s \in S^H$ in each iteration and for each of the three proposed scaling strategies in the consensus-fixing framework. The y-axis describes the total cost in € and the x-axis lists the 30 iterations of the consensus-fixing framework corresponding to the 30 technician hires $t \in T^I$. The total cost includes the three cost categories: staff costs, routing costs, and unserved task costs.



(a) `elinr` (b) `rlinr`

Figure S3: Each plot presents two key performance measures. The blue bar chart represents the average workday length in hours for an active technician, defined as a technician assigned to at least one task. The orange line indicates the total number of active technicians in each iteration of the framework. The three plots correspond to the average results for all historic scenarios $s \in S^H$ when using the three consensus-fixing strategies.

Chapter 7

Conclusion

7.1 Summary

In this dissertation, various optimization problems related to the routing and scheduling of technicians have been studied. Specifically, the work has focused on the telecommunication industry, where technicians carry out a range of maintenance and installation tasks related to wired network infrastructure, including coax, fiber, and copper technology. Many of the challenges in this sector are driven by dynamic and highly stochastic processes, as well as a poor data foundation. Although these prevalent real-life difficulties lend themselves readily to interesting new avenues of research with high socio-economic impact, a recent review by Paraskevopoulos et al. [108] emphasized the limited body of literature addressing stochastic elements within the Technicians Routing and Scheduling (TRSP) domain. In light of this, the focus of this dissertation has been to specifically contribute to filling this research gap.

Most of the work in this dissertation has revolved around enhancing an often overlooked aspect of routing, which this dissertation refers to as *micro-routing*. Traditional approaches to routing problems typically aim to minimize the distance driven between a set of locations or tasks; however, in stochastic environments, this distance often serves as an unrealistic measure of the actual distance traveled. Micro-routing captures the additional routing required to complete a task due to uncertainties. For example, in a parcel delivery scenario within urban areas, micro-routing represents the extra driving needed to find parking and for the driver to walk the remaining distance to the delivery address. Not only is micro-routing crucial for accurately estimating the actual driving dis-

tance and time, but it can also influence the optimal sequence in which a set of locations should be visited.

In the first paper, a heuristic data-driven inference framework based on a Variable Neighborhood Search (VNS) was proposed to address the issue of missing or incomplete network topology information. Incomplete knowledge of the network topology complicates both installation and maintenance tasks, often requiring technicians to visit multiple locations within a network to resolve a single issue. This lack of accurate topology information can lead to decreased customer satisfaction, as technicians are more likely to perform incorrect installations or fail to identify the true root cause of a problem. The proposed framework borrowed the optimality criterion *maximum parsimony* from computational biology and relied solely on modem data and geographical location information of key network components. A simulated computational study demonstrated that large topologies with up to 500 customers could be reconstructed with a high degree of accuracy. Additionally, the framework proved robust to up to 30% background noise in the modem data.

The second paper focused on optimizing the processing of time-series modem data to enhance its effectiveness in topology inference. The data encoding scheme developed in this paper was designed as a preprocessing step to support the methodology outlined in both the first and third papers. Specifically, the second paper introduced an encoder trained to identify irregular patterns in modem time-series data and encode these patterns as binary sequences. The encoder employed unsupervised contrastive learning, where a Siamese neural network was trained using both the true network topology and a set of false topologies. The network's weights were iteratively refined using a modified version of the maximum parsimony criterion, enabling the encoder to learn which data would make the true topology the most likely topology under the maximum parsimony criterion. Preliminary computational studies on networks with up to 20 internal nodes showed that the learned encoder achieved an estimated topology reconstruction accuracy of 99.9% and a data event identification accuracy of 93.4%. Moreover, the paper highlighted the potential of integrating an optimization algorithm directly into a deep learning loss function.

The third paper aimed to provide technicians with enhanced information to make more informed decisions in the micro-routing process of network fault tasks. Such tasks are created when a customer reports an issue with their connection. Using the customer's location within the network and the network topology, a technician can deduce a set of components in the network that each potentially could be the location of the fault. Rather than allowing a technician to search these locations blindly, paper III introduced a methodology to *score* each location based on its likelihood of being the fault source, thereby enabling more informed micro-routing decisions. The paper introduced a novel framework based on the maximum parsimony criterion, where a set of likely

downstream data *states* were reconstructed at each interior node in a network based on modem data. These states were then utilized to generate a likelihood score for each potential fault location. Although the paper represents ongoing work, preliminary results demonstrated that the true fault location consistently obtained either the highest or second highest likelihood score in 90% of the analyzed cases, underscoring the methodology's potential.

The fourth paper was a continuation of the third paper and addressed the TRSP problem with micro-routing for network fault tasks. The paper proposed a set of greedy policies (search strategies) in line with current industry practices. The paper also demonstrated that the micro-routing aspect could be decoupled from the overarching TRSP problem and addressed separately, using the expected-time solution from the micro-routing process as an input to the TRSP problem. A key contribution was the development of a fast polynomial time recursive algorithm capable of determining the optimal search policy for the micro-routing problem. By subsequently solving a MIP formulation of the TRSP problem, optimal expected-time solutions to the TRSP problem with micro-routing could be computed. Results demonstrated potential time savings of up to 80% compared to industry industry standards.

The fifth and final paper considered a strategic problem related to the TRSP, namely how to hire an optimal workforce of technicians. the paper introduced a workforce investment framework inspired by *consensus fixing*, which greedily builds a workforce in a hire-by-hire manner. In each iteration of the framework, a set of TRSP scenarios had to reach *consensus* about which technician to hire from a set of possible technician hires. Various consensus strategies were explored with the goal of minimizing the combined costs of hiring, driving, and penalties for unserved tasks. The framework's effectiveness was evaluated through out-of-sample testing and comparisons to *scenario-specific* solutions. The framework proved to be both fast and flexible and easy to integrate with existing routing and scheduling models, offering valuable insights into balancing workforce size with costs. Moreover, it enabled decision-makers to clearly assess the trade-offs between minimizing expenses and ensuring comprehensive task coverage.

7.1.1 Directions for future research

Multiple papers in this dissertation established the importance and potential impact of factoring micro-routing into planning. Micro-routing also exists in other routing problems such as parcel delivery problems, as mentioned in the summary above. Micro-routing might also be an important aspect of Home Health Care problems or routing problems involving electric vehicles (EVs), where drivers may have to choose between making significant changes to their planned route to find an available charging spot or waiting in line at a closer, but occupied, station. Including the stochastic aspects of micro-routing into existing routing

problems will lead to more realistic solutions and provide decision-makers with clear actions to take when faced with uncertainty.

The TRSP micro-routing problem considered in the fourth paper of this dissertation was solved to optimality. However, future work could focus on incorporating even more stochastic elements. In the paper, fixed service times for each location visited were assumed. In reality, service times are mostly stochastic. For example, the service time at a customer node may be influenced by the 'hospitality level' of the customer, and in general, it tends to take longer than visiting a simpler location like a roadside cabinet.

A perhaps even more interesting extension to the TRSP problem with micro-routing is to relax the constraint that a technician must complete a task before moving on to the next. While this adds a whole new dimension of complexity to the problem, it almost certainly would result in improved routing solutions as the original solution space is expanded as a result of the relaxation. However, in a real-life setting, a customer might be dissatisfied if a technician visits their home in the morning but does not confirm that their issue has been resolved until much later in the day. This could be mitigated by enforcing that a technician must finalize a task immediately if a customer's home is visited as part of the fault localization process.

An open research question based on the third paper remains how to translate the proposed fault likelihood scores into more comparative measures of likelihood that can be used to decide whether to visit one location over another. Anomaly detection or root cause analysis in data are often tackled by the data science research community using machine learning (ML) or unsupervised learning approaches. However, where ML methods do not always offer transparency into *why* they arrived at a given result, the label assignment approach from the third paper provides full transparency for how it scores each potential fault location. It might be an interesting possibility to explore a combined approach that leverages both the research maturity of the ML community with the interpretability of the label assignment approach, similar to how the second paper integrated an optimization algorithm directly into a deep learning loss function.

Much of the work in this dissertation is based on the *maximum parsimony* optimality criterion from the field of phylogenetic inference. However, various types of evolutionary inference methods exist, such as distance, likelihood, and Bayesian approaches. Re-exploring some of the problems introduced in this dissertation using these alternative methods could potentially lead to improved results and also presents an interesting avenue for future research.

Lastly, one of the key limitations of the consensus-fixing framework is its greedy approach, which involves making sequential investments one-by-one. An interesting direction for future research could be to treat the current framework as a *construction heuristic*. Once the initial set of investments has been obtained,

the solution could be iteratively improved by removing one or more investments and then rerunning the consensus-fixing framework to reinvest. This process could continue until the set of investment decisions stabilizes, resembling a *local investment search*. Such an approach is feasible due to the fast run-time (by strategic investment standards) and the highly parallelizable nature of the consensus-fixing framework.

Bibliography

[1] Tdc net annual report 2023. Technical report, TDC NET, 2023. Accessed: 2024-09-16.

[2] R. Agarwala and D. Fernández-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM J. Comput.*, 23:1216–1224, 01 1994.

[3] P. Ajawatanawong. *Molecular Phylogenetics: Concepts for a Newcomer*, pages 185–196. Springer International Publishing, Cham, 2017.

[4] S. Alunni. Cable plant upgrades mean support from interactive, multimedia services. *Network World*, 11(26):35, Jun 1994.

[5] M. Amer, T. U. Daim, and A. Jetter. A review of scenario planning. *Futures*, 46:23–40, 2013.

[6] Y. Anoshkina and F. Meisel. Technician teaming and routing with service-, cost-and fairness-objectives. *Computers & Industrial Engineering*, 135:868–880, 2019.

[7] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271, 2018.

[8] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[9] B. S. Borba, M. Z. Fortes, L. A. Bitencourt, V. H. Ferreira, R. S. Maciel, M. A. Guimaraens, G. B. Lima, E. U. Barboza, H. O. Henriques, N. C. Bergiante, and B. S. Moreira. A review on optimization methods for work-

force planning in electrical distribution utilities. *Computers & Industrial Engineering*, 135:286–298, 2019.

[10] P. D. Bruecker, J. V. den Bergh, J. Beliën, and E. Demeulemeester. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243(1):1–16, 2015.

[11] P. Buneman. The recovery of trees from measures of dissimilarity. In F. Hodson, D. Kendall, and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.

[12] Business Research Insights. Wired telecommunication carriers market size, share, growth, and industry analysis by type, application, regional insights, and forecast from 2031. `https://www.businessresearchinsights.com/market-reports/wired-telecommunication-carriers-market-102892`, 2024. Accessed: 2024-09-02.

[13] BusinessResearch. Coaxial cables global market report 2022. Technical report, The Business Research Company, 2022.

[14] CableLabs. Cable broadband technology gigabit evolution. `https://www.cablelabs.com/insights/cable-broadband-technology-gigabit-evolution`, 2016. visited: 12/12/23.

[15] CableLabs. *PNM Best Practices: HFC Networks (DOCSIS 3.0)*. Cable Television Laboratories, 2016. `https://www.cablelabs.com/specifications/proactive-network-maintenance-using-pre-equalization`.

[16] N. Cabrera, J.-F. Cordeau, and J. E. Mendoza. The workforce scheduling and routing problem with park-and-loop. *Networks*, pages 1–23, 2024.

[17] J. H. Camin and R. R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, pages 311–326, 1965.

[18] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499 – 517, 2004.

[19] D. Catanzaro. The minimum evolution problem: Overview and classification. *Networks*, 53(2):112–125, 2009.

[20] A. Cayley. A theorem on trees. *Quart. J. Math.*, 23:376–378, 1889.

[21] C. Challu, P. Jiang, Y. N. Wu, and L. Callot. Deep generative model with hierarchical latent factors for time series anomaly detection. In *International Conference on Artificial Intelligence and Statistics*, 2022.

[22] W.-C. Chang, C.-L. Li, Y. Yang, and B. Póczos. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, 2019.

[23] W. Chen and K. Shi. Multi-scale attention convolutional neural network for time series classification. *Neural Networks*, 136:126–140, 2021.

[24] X. Chen, B. W. Thomas, and M. Hewitt. The technician routing problem with experience-based service times. *Omega*, 61:49–61, 2016.

[25] X. Chen, B. W. Thomas, and M. Hewitt. Multi-period technician scheduling with experience-based service times and stochastic customers. *Computers & Operations Research*, 82:1–14, 2017.

[26] D. Chicco. Siamese neural networks: An overview. In H. Cartwright, editor, *Artificial Neural Networks*, pages 73–94. Springer US, New York, NY, USA, 2021.

[27] F. Chung, M. Garrett, R. Graham, and D. Shallcross. Distance realization problems with applications to internet tomography. *Journal of Computer and System Sciences*, 63:432–448, 12 2000.

[28] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. *ACM SIGMETRICS Performance Evaluation Review*, 30(1):11–20, 2002.

[29] M. Coates, M. Rabbat, and R. Nowak. Merging logical topologies using end-to-end measurements. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 192–203, 2003.

[30] J. F. Cordeau, G. Laporte, F. Pasin, and S. Ropke. Roadef 2007 challenge. Technical report, 2007.

[31] J.-F. Cordeau, G. Laporte, F. Pasin, and S. Ropke. Scheduling technicans and tasks in a telecommunication company. *J. Scheduling*, 13:393–409, 08 2010.

[32] K. Cordova-Pozo and E. A. Rouwette. Types of scenario planning and their effectiveness: A review of reviews. *Futures*, 149:103153, 2023.

[33] C. E. Cortés, M. Gendreau, L. M. Rousseau, S. Souyris, and A. Weintraub. Branch-and-price and constraint programming for solving a real-life technician dispatching problem. *European Journal of Operational Research*, 238(1):300–312, 2014.

[34] C. E. Cortés, M. Gendreau, D. Leng, and A. Weintraub. A simulation-based approach for fleet design in a technician dispatch problem with stochastic demand. *Journal of the Operational Research Society*, 62(8):1510–1523, Aug. 2011.

[35] R. d. B. Damm and D. P. Ronconi. A multi-objective biased random-key genetic algorithm for service technician routing and scheduling problem. In *International Conference on Computational Logistics*, pages 471–486. Springer, 2021.

[36] T. H. Davenport and G. Westerman. Why so many high-profile digital transformations fail. *Harvard Business Review*, 2018. Accessed: 2024-09-23.

[37] W. Day. Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology*, 103:429–438, 1983.

[38] W. H. Day, D. S. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81(1):33–42, 1986.

[39] A. De Bruyn, D. P. Martin, and P. Lefeuvre. *Phylogenetic Reconstruction Methods: An Overview*, pages 257–277. Humana Press, Totowa, NJ, 2014.

[40] T. De Ryck, M. De Vos, and A. Bertrand. Change point detection in time series data using autoencoders with a time-invariant representation. *IEEE Transactions on Signal Processing*, 69:3513–3524, 2021.

[41] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359, 2021.

[42] B. Estellon, F. Gardi, and K. Nouioua. High-performance local search for task scheduling with human resource allocation. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics: Second International Workshop, SLS 2009, Brussels, Belgium, September 3-4, 2009. Proceedings 2*, pages 1–15. Springer, 2009.

[43] FDM. *Cheaper to own a car – especially an electric car – in 2024*, Jan 2024. Accessed on August 26, 2024.

[44] J. Felsenstein. *Inferring Phylogenies*. Sinauer, 2003.

[45] C. Fikar and P. Hirsch. Home health care routing and scheduling: A review. *Computers and Operations Research*, 77:86–95, 2017.

[46] M. Fırat and C. A. Hurkens. An improved mip-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15(3):363–380, 2012.

[47] M. Fischer. On the uniqueness of the maximum parsimony tree for data with up to two substitutions: An extension of the classic buneman theorem in phylogenetics. *Molecular Phylogenetics and Evolution*, 137:127–137, 2019.

[48] W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[49] P. Forsare Källman. Unsupervised anomaly detection and root cause analysis in hfc networks : A clustering approach. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2021. `https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1612886`.

[50] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch. Somvae: Interpretable discrete representation learning on time series. In *7th International Conference on Learning Representations (ICLR)*. ICLR, May 2019.

[51] B. Fortz, O. Oliveira, and C. Requejo. Compact mixed integer linear programming models to the minimum weighted tree reconstruction problem. *European Journal of Operational Research*, 256(1):242–251, 2017.

[52] L. Foulds and R. Graham. The steiner problem in phylogeny is np-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[53] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivariate time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[54] M. Gamst and D. Pisinger. Decision support for the technician routing and scheduling problem. *Networks*, 83(1):169–196, Sept. 2023.

[55] M. Gendreau, F. Guertin, and J.-Y. P. and. É. *Taillard: "Parallel tabu search for real-time vehicle routing and dispatching". Transportation science*, 33(4):381–390, 1999.

[56] E. Gibellini and C. E. Righetti. Unsupervised learning for detection of leakage from the hfc network. In *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, pages 1–8. IEEE, 2018.

[57] T. R. Gregory. Understanding evolutionary trees. *Evol Educ Outreach*, 1:121–137, 04 2008.

[58] H. Gu, Y. Zhang, and Y. Zinder. An efficient optimisation procedure for the workforce scheduling and routing problem: Lagrangian relaxation and iterated local search. *Computers & Operations Research*, 144:105829, 2022.

[59] J. A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29(1):53–65, 1973.

[60] G. Heiler, T. Gadermaier, T. Haider, A. Hanbury, and P. Filzmoser. Identifying the root cause of cable network problems with machine learning. *arXiv*, 3 2022. `https://arxiv.org/abs/2203.06989`.

[61] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36:396–405, 1993.

[62] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, New York, NY, 7th edition, 2001.

[63] J. Hu, Z. Zhou, X. Yang, J. Malone, and J. W. Williams. Cablemon: Improving the reliability of cable broadband networks via proactive network maintenance. In *Proceddings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020. `https://www.usenix.org/conference/nsdi20/presentation/hu-jiyao`.

[64] A. Hyvärinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3772–3780, Red Hook, NY, USA, 2016. Curran Associates Inc.

[65] INFORMS. Optimizing delivery routes, 2024. Accessed: 2024-09-21.

[66] Innovationsfonden. Kunstig intelligens og avancerede teknologier i udviklingen, 2024. Accessed: 2024-09-17.

[67] Intel. Intel advanced vector extensions programming reference. Technical Report 319433-011, Intel Corporation, 2011.

[68] International Organization for Standardization. Information technology —- Open Systems Interconnection -— Basic Reference Model: The Basic Model. Standard, International Organization for Standardization, Geneva, CH, 1994.

[69] International Telecommunication Union (ITU). The last-mile internet connectivity solutions guide: Sustainable connectivity options for unconnected sites, 2020. Accessed: 2024-10-09, Table 1, p. 14.

[70] H. Jingwen, J. Meng, H. Zheng, P. Parikh, and D. Guan. Achieving decent living standards in emerging economies challenges national mitigation goals for co2 emissions. *Nature Communications*, 14, 10 2023.

[71] P. Kapli, Z. Yang, and M. Telford. Phylogenetic tree building in the genomic age. *Nature Reviews Genetics*, 21:1–17, 05 2020.

[72] Y. Kawahara and M. Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(2):114–127, 2012.

[73] K. K. Kidd and L. A. Sgaramella-Zonta. Phylogenetic analysis: concepts and methods. *American journal of human genetics*, 23(3):235, 1971.

[74] D. E. Knuth. Optimum binary search trees. *Acta Informatica*, 1:14–25, 1971.

[75] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[76] F. Lartey. Proactive network and technical facilities monitoring using standardized scorecards. In *Proceedings of the Fall Technical Forum*, Denver, CO, 2017. SCTE, ISBE, NCTA. `https://www.nctatechnicalpapers.com/Paper/2017/2017-proactive-network-and-technical-facilities-monitoring-using-standardized-scorecards`.

[77] P. H. Le-Khac, G. Healy, and A. F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[78] S. Lee, K. Levanti, and H. S. Kim. Network monitoring: Present and future. *Computer Networks*, 65:84–98, 2014.

[79] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon. Similarity preserving representation learning for time series clustering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 2845–2851. AAAI Press, 2019.

[80] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[81] H. Madsen. *Time series analysis*. Chapman & Hall, 2007.

[82] S. Makboul, S. Kharraja, A. Abbassi, and A. E. H. Alaoui. A multiobjective approach for weekly green home health care routing and scheduling problem with care continuity and synchronized services. *Operations Research Perspectives*, 12:100302, 2024.

[83] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *arXiv*, 2016. `https://arxiv.org/abs/1607.00148`.

[84] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff. Timenet: Pretrained deep recurrent neural network for time series classification. In *25th European Symposium on Artificial Neural Networks, ESANN 2017, Bruges, Belgium, April 26-28, 2017*, 2017.

[85] Y. Mao, H. Jamjoom, S. Tao, and J. M. Smith. Networkmd: topology inference and failure diagnosis in the last mile. In *Proceedings of the 7th*

ACM SIGCOMM Conference on Internet Measurement, IMC '07, page 189–202, New York, NY, USA, 2007. Association for Computing Machinery.

[86] P. Masse and R. Gibrat. Application of linear programming to investments in the electric power industry. *Management Science*, 3:149–166, 1957.

[87] I. Mathlouthi, M. Gendreau, and J.-Y. Potvin. Mixed integer linear programming for a multi-attribute technician routing and scheduling problem. *Information Systems and Operational Research*, 56(1):33–49, 2018.

[88] I. Mathlouthi, M. Gendreau, and J.-Y. Potvin. Branch-and-price for a multi-attribute technician routing and scheduling problem. *Operations Research Forum*, 2(1):1, 2021.

[89] I. Mathlouthi, M. Gendreau, and J.-Y. Potvin. Branch-and-price for multi-attribute technician routing and scheduling problem. *SN Operations Research Forum*, 2(1):1–35, 2021.

[90] I. Mathlouthi, M. Gendreau, and J.-Y. Potvin. A metaheuristic based on tabu search for solving a technician routing and scheduling problem. *Computers & Operations Research*, 125, 2021.

[91] McKinsey & Company. Unlocking success in digital transformations, 2023. Accessed: 2024-09-23.

[92] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, oct 1960.

[93] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[94] G. Monge. Memoir on the theory of excavations and embankments. pages 666–704, 1781.

[95] J. Määttä and T. Roos. Maximum parsimony and the skewness test: A simulation study of the limits of applicability. *PloS one*, 11:e0152656, 04 2016.

[96] L. Nakhleh, G. Jin, F. Z. F, and J. Mellor-Crummey. Reconstructing phylogenetic networks using maximum parsimony. *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, pages 93–102, 2005.

[97] M. Nei. Phylogenetic analysis in molecular evolutionary genetics. *Annual Review of Genetics*, 30(1):371–403, 1996. PMID: 8982459.

[98] H. Nguyen and R. Zheng. A binary independent component analysis approach to tree topology inference. *IEEE Transactions on Signal Processing*, 61(12):3071–3080, 2013.

[99] J. Ni and S. Tatikonda. Network tomography based on additive metrics. *IEEE Transactions on Information Theory*, 57(12):7798–7809, 2011.

[100] C. C. Nielsen. *Tactical Routing and Scheduling Optimisation for Dynamic Field Service*. Phd thesis, Technical University of Denmark, 2024.

[101] C. C. Nielsen and D. Pisinger. Tactical planning for dynamic technician routing and scheduling problems. *Transportation Research Part E: Logistics and Transportation Review*, 177:103225, 2023.

[102] C. C. Nielsen and D. Pisinger. Tactical planning for dynamic technician routing and scheduling problems with skills and depots. *European Journal of Operational Research*, submitted, 2024.

[103] C. C. Nielsen and D. Pisinger. Workforce scheduling and routing with substitution skills and workload balancing. *Transportation Research Part E: Logistics and Transportation Review*, (submitted), 2024.

[104] M. Nowak and P. Szufel. Technician routing and scheduling for the sharing economy. *European Journal of Operational Research*, 314(1):15–31, 2024.

[105] U. N. F. C. on Climate Change (UNFCCC). Paris agreement, 2015. Adopted December 12, 2015. Entered into force November 4, 2016.

[106] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4):431–448, 2018.

[107] R. Padmanabhan, A. Schulman, D. Levin, and N. Spring. Residential links under the weather. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 145–158, New York, NY, USA, 2019. Association for Computing Machinery. `https://dl.acm.org/doi/10.1145/3341302.3342084`.

[108] D. C. Paraskevopoulos, G. Laporte, P. P. Repoussis, and C. D. Tarantilis. Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, 263(3):737–754, 2017.

[109] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances*

in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

[110] R. Patterson and E. Rolland. Hybrid fiber coaxial network design. *Operations Research*, 50:538–551, 06 2002.

[111] E. Pekel. A simple solution to technician routing and scheduling problem using improved genetic algorithm. *Soft Computing*, 26(14):6739–6748, 2022.

[112] D. L. Pereira, J. C. Alves, and M. C. de Oliveira Moreira. A multiperiod workforce scheduling and routing problem with dependent tasks. *Computers & Operations Research*, 118, 2020.

[113] D. Pham and G. Kiesmüller. Hybrid value function approximation for solving the technician routing problem with stochastic repair requests. *Transportation Science*, 58, 01 2024.

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

[115] V. Pillac, C. Guéret, and A. Medaglia. On the Dynamic Technician Routing and Scheduling Problem. Research report, Ecole des Mines de Nantes, 2012.

[116] V. Pillac, C. Guéret, and A. Medaglia. A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7):1525–1535, 2013.

[117] V. Pillac, C. Guéret, and A.-L. Medaglia. A fast reoptimization approach for the dynamic technician routing and scheduling problem. *Recent developments in metaheuristics*, pages 347–367, 2018.

[118] D. Pisinger. Consensus fixing for two-stage stochastic optimization problems. *Submitted*, 2024.

[119] D. Pisinger and S. Sørensen. Topology reconstruction using time series data in telecommunication networks. *Networks*, 83(2):408–427, 2024.

[120] E. Pourjavad and E. Almehdawe. Optimization of the technician routing and scheduling problem for a telecommunication industry. *Annals of Operations Research*, 315:371–395, 2022.

[121] W. Powell. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. Wiley, 2022.

[122] W. Powell. *Sequential Decision Analytics and Modeling: Modeling with Python*. Foundations and Trends(r) in Technology, Information and Ope. Now Publishers, 2022.

[123] W. B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019.

[124] I. Promptlink Communications. What is broadband network noise and why is it difficult to find? `https://www.promptlink.com/media-library/blog/what-is-broadband-network-noise-and-why-is-it-difficult-to-find.html`, 2019. [Online; accessed 13-December-2023].

[125] I. Promptlink Communications. What is cable ingress. `https://www.promptlink.com/media-library/blog/what-is-cable-ingress.html`, 2023. "[Online; accessed 13-December-2023]".

[126] PwC. Global telecom outlook 2023-2027, 2023. Accessed: 2024-09-22.

[127] T. E. Rasmussen. *Machine Learning and Time Series Analysis for Optimization and Anomaly Detection in Field Service*. Phd thesis, Technical University of Denmark, 2024. Part I: Background and Theory.

[128] T. E. Rasmussen and S. Sørensen. Encoding binary events from continuous time series in rooted trees using contrastive learning. *arXiv*, 2024. `https://arxiv.org/abs/2401.01242`.

[129] T. E. Rasmussen, S. Sørensen, D. Pisinger, T. M. Jørgensen, and A. Baum. Topology reconstruction in telecommunication networks: Embedding operations research within deep learning, 2024. Preprint available at SSRN: `https://ssrn.com/abstract=4757707`.

[130] ReportLinker. Global and China RF coaxial cable industry report, 2019-2025. Technical report, ReportLinker, 2019.

[131] H. Rezgui. An overview of optical fibers. *Global Journal of Science Frontier Research*, 21:14–20, 01 2022.

[132] H. Ritchie and M. Roser. Co_2 emissions per capita, 2024. Accessed: 2024-09-16.

[133] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[134] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986.

[135] J. Rupe and J. Zhu. Kickstarting proactive network maintenance with the proactive operations platform and example application. In *Proceedings of the Fall Technical Forum*, New Orleans, LA, 2019. SCTE, ISBE, NCTA. `https://www.nctatechnicalpapers.com/Paper/2019/2019-kickstarting-proactive-network-maintenance`.

[136] A. Rzhetsky and M. Nei. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular biology and evolution*, 10(5):1073–1095, 1993.

[137] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 07 1987.

[138] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975.

[139] M. W. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations research*, 4:285–305, 1985.

[140] Seagate Technology. Rethink data: Put more of your data to work – from edge to cloud, 2020. Accessed: 2024-09-23.

[141] A. Shapiro, D. Dentcheva, and A. Ruszczynki. *Lectures on stocastic programming: Modeling and theory (2nd)*. SIAM, 2014.

[142] M. Simakovic and Z. Cica. Detection and localization of failures in hybrid fiber-coaxial network using big data platform. *Electronics*, 10(23), 2021. https://www.mdpi.com/2079-9292/10/23/2906.

[143] M. N. Simaković, I. B. Masnikosa, and C. G. Zoran. Performance monitoring challenges in hfc networks. In *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, pages 385–388, 2017.

[144] SKAT. *Taxable and tax-free tansport allowance*, Jul 2024. Accessed on August 26, 2024.

[145] Society of Cable Telecommunications Engineers, Inc. *Test Procedure for Common Path Distortion (CPD)*. SCTE and ISBE, 2019. `https://www.scte.org/standards/library/catalog/scte-109-test-procedure-for-common-path-distortion/`.

[146] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[147] Y. Song, M. Ulmer, B. Thomas, and S. Wallace. Building trust in home services - stochastic team-orienteering with consistency constraints. *Transportation Science*, 06 2019.

[148] S. Souyris, C. Cortés, F. Ordonez, and A. Weintraub. A robust optimization approach to dispatching technicians under stochastic service times. *Optimization Letters*, 7, 10 2013.

[149] S. Sridhar, F. Lam, G. E. Blelloch, R. Ravi, and R. Schwartz. Mixed integer linear programming for maximum-parsimony phylogeny inference. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(3):323–331, 2008.

[150] M. Steel and D. Penny. Parsimony, Likelihood, and the Role of Models in Molecular Phylogenetics. *Molecular Biology and Evolution*, 17(6):839–850, 06 2000.

[151] S. Sørensen. *Integrated Fault Localization and Field Service Logistics*. Phd thesis, Technical University of Denmark, 2024. Chapter 4: Fault Estimation in Telecommunication Networks: A Maximum Parsimony Approach.

[152] E. Tsang and C. Voudouris. Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119–127, Mar. 1997.

[153] M. Ulmer, J. Goodson, D. Mattfeld, and B. Thomas. On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics*, 9:100008, 06 2020.

[154] M. W. Ulmer. Horizontal combinations of online and offline approximate dynamic programming for stochastic dynamic vehicle routing. *Central European Journal of Operations Research*, 28(1):279–308, 2020.

[155] United Nations, Department of Economic and Social Affairs, Population Division. World population prospects 2024, 2024. Accessed: 2024-09-16.

[156] United Nations Development Programme. Human development index (hdi), 2024. Accessed: 2024-09-16.

[157] United Nations Environment Programme (UNEP). Emissions gap report 2023: Turning up the pace of climate action, 2023. Accessed: 2024-09-16.

[158] UPS. Ups to enhance orion with continuous delivery route optimization, 2024. Accessed: 2024-09-21.

[159] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, page 125, 2016.

[160] A. van den Oord, O. Vinyals, and k. kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[161] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American statistical association*, 91(433):365–377, 1996.

[162] V. Verroios, V. Efstathiou, and A. Delis. Reaching available public parking spaces in urban environments using ad hoc networking. In *2011 IEEE 12th International Conference on Mobile Data Management*, volume 1, pages 141–151, 2011.

[163] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operations Research*, 234(3):658–673, 2014.

[164] B. Wang, W. Wei, H. Dinh, W. Zeng, and K. R. Pattipati. Fault localization using passive end-to-end measurements and sequential testing for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 11(3):439–452, 2011.

[165] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017.

[166] T. H. Williams. System and method to locate common path distortion on cable systems. Patent US 2004O245995A1, Arcom Digital Patent LLC, 12 2004. `https://patents.google.com/patent/US20080319689`.

[167] L. Wolcott, M. O'Dell, P. Kuykendall, and V. Gopal. A pnm system using artificial intelligence, hfc network impairment, atmospheric and weather data to predict hfc network degradation and avert customer impact. In *Proceedings of the Fall Technical Forum*. SCTE, ISBE, 2018. `https://www.nctatechnicalpapers.com/Paper/2018/2018-a-pnm-system`.

[168] World Bank. Measuring the emissions & energy footprint of the ict sector: Implications for climate action, 2023. Accessed: 2024-09-22.

[169] E. H.-K. Wu, J. Sahoo, C.-Y. Liu, M.-H. Jin, and S.-H. Lin. Agile urban parking recommendation service for intelligent vehicular guiding system. *IEEE Intelligent Transportation Systems Magazine*, 6(1):35–49, 2014.

[170] A.-E. Yahiaoui, S. Afifi, and H. Allaoui. Enhanced iterated local search for the technician routing and scheduling problem. *Computers & Operations Research*, 160:106385, 2023.

[171] K. Yamanishi and J.-i. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 676–681, New York, NY, USA, 2002. Association for Computing Machinery.

[172] I. Yaqoob, I. A. T. Hashem, A. Gani, S. Mokhtar, E. Ahmed, N. B. Anuar, and A. V. Vasilakos. Big data: From beginning to future. *International Journal of Information Management*, 36(6, Part B):1231–1247, 2016.

[173] M. Yu, A. Greenberg, D. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim. Profiling network performance for multi-tier data center applications. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011. `https://www.usenix.org/conference/nsdi11/profiling-network-performance-multi-tier-data-center-applications`.

[174] B. Yuan, R. Liu, and Z. Jiang. A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. *International Journal of Production Research*, 53:1–15, 09 2015.

[175] E. Zamorano and R. Stolletz. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257(1):55–68, 2017.

[176] J. Zhu, K. Sundaresan, and J. Rupe. Proactive network maintenance using fast, accurate anomaly localization and classification on 1-d data series. In *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2020. `https://ieeexplore.ieee.org/document/9187045`.

[177] J. Zhu, K. Sundaresan, and J. Rupe. Proactive network maintenance using fast, accurate anomaly localization and classification on 1-d data series. In *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–11. IEEE, 2020.

Technical
University of
Denmark

Akademivej, Building 358
2800 Kgs. Lyngby
Tlf. 45 25 48 00

www.man.dtu.dk