Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Metaheuristic Algorithms for Optimization Problems Sharing Permutative Representation

Disertation thesis

Ing. David Woller

Ph.D. programme: Informatics
Supervisor: RNDr. Miroslav Kulich, Ph.D.

Prague, March 2024

Acknowledgements

I would like to thank my supervisor, Mirek Kulich, for his expert guidance and always available assistance during the almost five years of my PhD studies. I am grateful to the Intelligent and Mobile Robotics (IMR) laboratory for co-funding my work and providing a friendly and supportive environment. I also appreciate my students and collaborators Jakub Rada, Jan Hrazdíra, Tomáš Hromada and David Pažout for implementing some of my ideas and trusting me with supervising their thesis. Special thanks to my closest colleagues Viktor Kozák, Jan Mikula, Luká Bertl, David Zahrádka, and Frank, who made my PhD a mostly pleasant experience. Finally, I would like to thank my family for all their support.

My PhD studies were funded by taxpayers' money through the PhD scholarship. The Czech Technical University in Prague supported my work with grants SGS18/206/OHK3/3T/37, SGS21/185/OHK3/3T/37 and SGS23/122/OHK3/2T/13. The presented research work was also supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (registration no. CZ.02.1.01/0.0/0.0/15003/0000470), European Union's Horizon 2020 research and innovation program under grant agreement No 688117, and by the Czech Science Foundation (GACR) under Grant Agreements 19-26143X and 23-05104S. Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic. My research internship was co-funded by the EU Erasmus+ program and Avignon University. Other activities abroad were supported by the "Nadační fond ČVUT Stanislava Hanzla" and "Nadace Nadání Josefa, Marie a Zdeňky Hlávkových" foundations.

Copyright

Abstract

Developing metaheuristic algorithms for computationally challenging combinatorial optimization problems is an important research area in operational research and related fields of computer science. In this thesis, we address a wide class of optimization problems with permutative representation, that is, problems whose solutions can be encoded as an ordered sequence of nodes from a given set. Their optimization lies in selecting the best subset and finding the best order of these nodes with respect to a problem-specific objective function and constraints. This thesis is a compilation of six core publications, three of which are journal articles. The thesis consists of two complementary research streams, each of which is covered by three core publications. The first stream is dedicated to developing problem-specific metaheuristic algorithms for various recently formulated problems with permutative representation. These problems are path planning for a mobile robot localizing radiation sources, route planning for a fleet of electric vehicles, and maintenance scheduling of a power transmission network. The first problem is motivated by an implemented robotic application, while the other two are novel optimization problems from international competitions, most notably the ROADEF Challenge. For each problem, we propose a specialized algorithm based on an established metaheuristic, such as Variable Neighborhood Search or Large Neighborhood Search, for which we design the necessary problem-specific components. The algorithms developed are either the first method addressing a highly specific newly formulated problem or belong among the state of the art, as documented by their success in the competitions. In the second research stream, we propose the generic metaheuristic solver for optimization problems with permutative representation, which is a tool that aims to combine the versatility of established Integer Programming solvers and the scalability of specialized metaheuristic algorithms for the studied class of problems. The solver is successfully benchmarked on several classical optimization problems and is subsequently deployed to some newly proposed problems in the follow-up work, inspired by an application in autonomous mining. In this work, we introduce the concept of self-deleting graphs and propose the variants of Travelling Salesperson Problem and Hamiltonian Cycle Problem on these graphs. The thesis is concluded by proposing and solving the problem of finding a placement of circles along a given tour, the Path-Conforming Circle Placement Problem (PCCP), which addresses another aspect of the motivating mining application.

Keywords: permutative representation, metaheuristic algorithm, generic solver, radiation search, Electric Vehicle Routing Problem, Transmission Maintenance Scheduling, self-deleting graph, Travelling Salesperson Problem, Hamiltonian Cycle Problem, autonomous mining, Path-Conforming Circle Placement.

Abstrakt

Vývoj metaheuristických algoritmů pro výpočetně náročné problémy z kombinatorické optimalizace je důležitou výzkumnou oblastí v operačním výzkumu a příbuzných oblastech počítačových věd. V této práci se věnujeme široké třídě optimalizačních problémů s permutativní reprezentací, tedy problémům, jejichž řešení lze reprezentovat uspořádanou sekvencí prvků z dané množiny. Optimalizace této třídy problémů spočívá ve volbě vhodné podmnožiny prvků a jejich uspořádání tak, aby byly splněny všechny omezující podmínky a minimalizována kriteriální funkce konkrétního problému. Tato práce je prezentována formou kompilace šesti hlavních publikací, z nichž tři jsou publikace časopisecké. Práce sestává ze dvou doplňujících se výzkumných směrů, z nichž každý je pokryt třemi články. První směr se věnuje návrhu specializovaných metaheuristiků algoritmů pro různé aktuálně formulované problémy s permutativní reprezentací. Tyto problémy jsou: plánování pohybu mobilního robotu lokalizujícího zdroje záření, plánování cest pro flotilu elektrických vozidel a rozvrhování údržby přenosové sítě. První problém je motivován realizací reálné robotické aplikace, zatímco zbylé dva jsou aktuální optimalizační problémy nově formulované v rámci mezinárodních soutěží, především ROADEF Challenge. Pro každý z těchto problémů navrhujeme specializovaný algoritmus postavený na zavedené metaheuristice, jako Variable Neighborhood Search nebo Large Neighborhood Search, pro kterou dále návrhujeme nezbytné specializované komponenty na míru danému problému. Vyvinuté algoritmy jsou buď první existující metodou řešící vysoce specifický nově formulovaný problém, nebo patří mezi nejlepší algoritmy pro řešený problém, jak dokládá jejich soutěžní umístění. Ve druhém výzkumném směru navrhujeme obecný metaheuristický řešič pro optimalizační problémy s permutativní reprezentací, což je nástroj mající za cíl zkombinovat pro danou třídu problémů univerzalitu zavedených řešičů postavených na celočíselném programování a škálovatelnost specializovaných metaheuristických algoritmů. Navržený řešič je úspěčně otestován na několika klasických optimalizačních problémech a je následně aplikován v navazujícím výzkumu, inspirovaným aplikací v autonomní povrchové těžbě. V tomto výzkumu definujeme koncept takzvaných samomazacích grafů a formulujeme na nich varianty problému obchodního cestujícího a problému hamiltonovské cesty. Práce je zakončena formulováním a řešením problému umístění kruhových překážek podél dané cesty, který se zabývá dosud neřešeným aspektem motivující těžební aplikace.

Klíčová slova: permutativní reprezentace, metaheuristický algoritmus, obecný řešič, hledání zdrojů radiace, problém směrování flotily elektrických vozidel, rozvrhování údržby přenosové sítě, samomazací graf, problém obchodního cestujícího, hamiltonovská cesta, autonomní těžba, problém umístění kruhových překážek podél dané cesty.

List of Acronyms

ACO Ant Colony Optimization. 6, 10, 36
ALNS Adaptive Large Neighborhood Search. 7, 54, 123
ASCHEA Adaptive Segregational Constraint Handling Evolutionary Algorithm. 90, 124
BKS Best-Known Solution. 35, 54
CMA-ES Covariance Matrix Adaptation Evolution Strategy. 10, 55
CP Constraint Programming. 1, 3
CTU Czech Technical University. ii, 126
CVRP Capacitated Vehicle Routing Problem. 4, 35, 90
EVRP Electric Vehicle Routing Problem. iv, 2, 3, 6, 7, 35, 90, 123, 126
GA Genetic Algorithm. 6, 9
GLNS Generalized Large Neighborhood Search. 7, 11, 123
GRASP Greedy Randomized Adaptive Search Procedure. 7, 35, 55, 123
GTSP Generalized Travelling Salesperson Problem. 7, 11
GVRP Green Vehicle Routing Problem. 35
HCP Hamiltonian Cycle Problem. iv, 4, 90, 126
HCP-SD Hamiltonian Cycle Problem on Self-Deleting graphs. 4, 103, 124
IG Iterated Greedy. 7
ILP Integer Linear Program. 8
ILS Iterated Local Search. 6, 7, 55, 123, 124
IMR Intelligent and Mobile Robotics. ii, 7
IP Integer Programming. iv, 1–3, 5, 6, 8, 90, 124, 126
IQCP Integer Quadratically Constrained Program. 8
IQP Integer Quadratic Program. 8
LNS Large Neighborhood Search. iv, v, 7
LP Linear Program. 8
MILP Mixed Integer Linear Program. 7–9, 55
MIQCP Mixed Integer Quadratically Constrained Program. 8
MIQP Mixed Integer Quadratic Program. 8
NPFS Non-Permutation Flowshop Scheduling Problem. 4, 90
NSGA Non-dominated Sorting Genetic Algorithm. 10
PCCP Path-Conforming Circle Placement Problem. iv, 4, 115, 124
PSO Particle Swarm Optimization. 10
Q3AP Three Dimensional Assignment Problem. 10
QAP Quadratic Assignment Problem. 90
ROADEF French Operations Research & Decision Support Society. iv, v, 6–8, 54, 123, 126
RTE Réseau de Transport d'Électricité. 54, 123
SA Simulated Annealing. 6
SOP Sequential Ordering Problem. 9
TMS Transmission Maintenance Scheduling. 90
TRP Travelling Repairman Problem. 9
TS Tabu Search. 6, 10
TSP Travelling Salesperson Problem. iv, 1, 3, 4, 7, 9, 90, 115, 126
TSP-CP Travelling Salesperson Problem with Circle Placement. 115, 124, 126
TSP-SD Travelling Salesperson Problem on Self-Deleting graphs. 4, 103, 115, 124
UAV Unmanned Aerial Vehicle. 2, 7, 11
UGV Unmanned Ground Vehicle. 2, 3, 11
VND Variable Neighborhood Descent. 35
VNS Variable Neighborhood Search. iv, v, 7, 10, 35, 36, 55, 123, 124
VRP Vehicle Routing Problem. 2, 9, 35, 123

Contents

Chapter 1

Introduction

With a history dating back to the 18^{th} century [15], combinatorial optimization is now a mature field with robust theoretical foundations and a broad portfolio of powerful methods. However, challenging applications are continually emerging in a wide range of diverse fields, such as resource allocation [16], machine learning [17], supply chain management [18], financial engineering [19], or robot routing [20]. New applications often require the formulation of new models, the scaling up of existing methods, or the design of entirely new algorithms. Moreover, the underlying optimization problems are often intractable. For these reasons, selecting the most suitable method typically requires a tradeoff between some of these criteria: optimality or approximation guarantee, solution quality, runtime, scalability, design time, and versatility.

Some problems are **well-solved**, meaning that exact polynomial-time algorithms exist. These are, for example, various shortest path problems, network flows, minimum spanning trees, or matching problems [21]. If the problem at hand can be reduced to any of these, it belongs to the $\mathcal{P}$ complexity class and can be considered efficiently solvable.

For other problems, which are known to be $\mathcal{NP}$-hard, the existence of such algorithms is unlikely, unless $\mathcal{P} = \mathcal{NP}$. Then, it comes to balancing the mentioned tradeoffs and picking the method most suitable, rather than perfect, for the given application. The first option might be an **approximation algorithm**, capable of providing a near-optimal solution in polynomial time, together with a provable guarantee of the solution quality, such as the classical Christofides algorithm [22] for the Travelling Salesperson Problem. The main advantage of such a polynomial algorithm lies in its performance and scalability. However, the provided quality guarantee might be too weak and easily surpassed by other methods in practice, and most importantly, such an algorithm might not even exist for the problem at hand.

When the optimality of the solution is required, the available design time is limited, and the instances are of moderate size, the most suitable choice is to use **Integer Programming (IP)**. Today, there are multiple solvers, such as Gurobi Optimizer [23], IBM ILOG CPLEX Optimizer [24], or FICO Xpress Solver [25], that provide highly efficient implementations of state of the IP art algorithms. The only requirement to the user is to formalize the problem and create a sensible model, which makes these solvers very attractive and widely used in practice. However, an explicit IP formulation of a high-dimensional problem may result in an immensely large model, which can be difficult to work with solely due to memory limitations. Additionally, the computational complexity of an exact solver is inherently exponential when solving an $\mathcal{NP}$-hard problem, which makes it intractable. For these reasons, scalability remains a major limitation of problem nonspecific IP methods. Moreover, IP modeling is limited to linear and quadratic (in)equalities. An alternative declarative programming paradigm with user properties similar to IP is **Constraint Programming (CP)**. CP is most suitable for finding any feasible solution in highly constrained problems, but can also be used for optimization. A major limitation of CP solvers is their inability to handle continuous variables.

Computationally challenging applications are often tackled by **metaheuristics**, high-level algorithmic frameworks that can be adapted to problem-specific metaheuristic algorithms. Metaheuristic algorithms often do not provide any guarantees about solution quality. They focus on performing an efficient local search in diverse promising regions of the solution space, without traversing it exhaustively. Despite being neither complete nor optimal, they are frequently used in applications where no other approach is computationally feasible. Moreover, metaheuristic algorithms are often documented to obtain solutions of similar quality as IP solvers in a significantly shorter time even on smaller problem instances. Concerning their major drawbacks, apart from the obvious lack of guarantees, most applications of metaheuristics require considerable design time and the implementations are not versatile. This is because essential components, such as various heuristics and operators, tend to be tailored to a particular problem. Using problem-specific information about the structure of the objective function or individual constraints can greatly increase the performance of these components, but makes the metaheuristic algorithm highly customized.

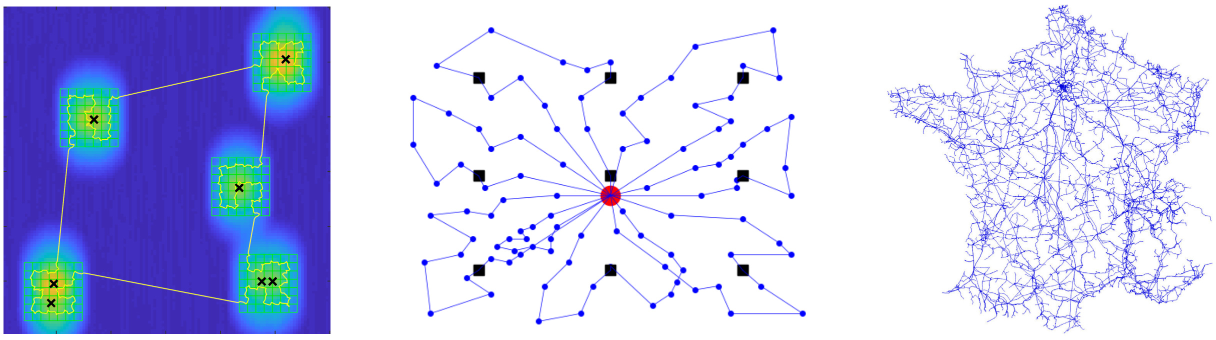
In this thesis, we focus on the design of **metaheuristic algorithms for problems with permutative representation**, that is, combinatorial optimization problems, whose solution can be encoded as an ordered sequence of possibly recurring nodes from a predefined set. Often, these problems differ only in the definition of the objective function and the set of constraints to be satisfied. Permutative representation naturally emerges in numerous frequently studied classes of problems, such as operation planning, vehicle routing, or resource scheduling. This thesis is a compilation of six core publications, three of which were published in impacted journals. Each core publication addresses a different problem with permutative representation, or the whole class of problems.

Problem-specific metaheuristic algorithms

In the first stream of work, consisting of the core publications [c1]–[c3], we propose various problem-specific metaheuristic algorithms tailored to a particular problem. The common goal was to develop a specialized algorithm that would be highly scalable and provide near-optimal solutions in a reasonable runtime. This was achieved primarily by designing custom components, such as local search operators, construction, destruction, or repair heuristics, and integrating them into suitable metaheuristics. These components typically exploit domain-specific knowledge to (i) efficiently and often incrementally evaluate the objective function, and (ii) take into account various constraints and restrict or navigate the algorithm towards searching only the space of valid solutions. From an application perspective, we tried to focus on recently formulated problems of practical relevance.

In the first core publication [c1], we proposed a metaheuristic algorithm for the path planning of an Unmanned Ground Vehicle (UGV) in complex robotic application. The objective was to pefrom an accurate localization of radiation sources by a heterogeneous robotic team. The essence of the planning problem lay in determining the order of inspection of individual regions of interest and selecting the most advantageous vehicle maneuver in each region, see Figure 1.1a. In this visualization, the yellow-green areas, identified by an UAV, are the areas of possible occurrence of a radiation source. These are then divided into smaller regions of interest (green squares), which are inspected by an UGV. Each region has to be inspected by performing a maneuver along a circular arc of given parameters in order to maximize localization accuracy. The yellow line then represents a near-optimal UGV trajectory, and the black crosses are the detected source locations.

The second core publication [c2] was motivated by an international competition organized within the IEEE WCCI 2020 conference [27]. The competition proposed the Electric Vehicle Routing Problem (EVRP), a novel variant of the classical Vehicle Routing Prob-



(a) Localization of radiation sources by UGV [c1]

(b) Electric Vehicle Routing Problem (EVRP) - solution [c2]

(c) Power transmission network [c3] (source: [26])

Figure 1.1: Illustrations to problems [c1]–[c3]

lem (VRP), for which we developed the winning algorithm. In this problem, the goal was to determine the order in which to serve individual customers using a fleet of electric vehicles while respecting constraints on the cargo load and range of the electric vehicles. Figure 1.1b shows the central depot (red circle), customers (blue circles), charging stations (black squares) and routes of individual vehicles (blue lines).

Similarly, the third core publication [c3] describes our submission to the ROADEF Challenge 2020 [28], a prestigious international competition with a 25-year history and dozens of participating teams from all over the world in each run. The 2020 problem, formulated by an industrial partner, focused on the maintenance scheduling of a large power transmission network (Figure 1.1c). The goal was to schedule the interventions necessary to carry out the preventive maintenance of individual power lines while respecting various constraints, such as mutual exclusiveness, limited workforce and resources, or seasonal risk factors. Our algorithm finished 2^{nd} out of 31 teams in the junior category and 8^{th} out of 74 in overall ranking. To summarize, for each of the three mentioned problems, we successfully developed a state of the art algorithm for the given problem, which is documented by the success in the competition of other methods, publication in an impacted journal, or both.

Generic metaheuristic solver for problems with permutative representation

The second stream of work is directly motivated by the first one and builds on the acquired experience. Instead of developing yet another specialized solver for the next problem that arises, our focus shifted towards designing a universal metaheuristic solver that could be readily applied to the whole class of problems with permutative representation. The idea was to design a solver that would require only the formulation of the problem in a simple unified formalism and not the implementation of specialized components, similarly to the even more generic IP and CP solvers based on exact methods. Performance-wise, the proposed generic metaheuristic solver's scalability should surpass the exact methods at the usual cost of no guarantees and get closer to the specialized metaheuristic algorithms.

In the fourth core publication [c4], we propose a formalism to describe problems with permutative representation. Together with this formalism, we create a generic metaheuristic solver capable of solving the problems defined in this formalism. The solver is highly modular, providing various alternative metaheuristics and low-level components. Thus, it can be automatically configured to fit a particular problem. The solver was successfully benchmarked against IP Gurobi Optimizer on several classical problems: Travelling

(a) Autonomous open-pit mining: Phase 1 of "drill and blast" method [c5] (source: [29])

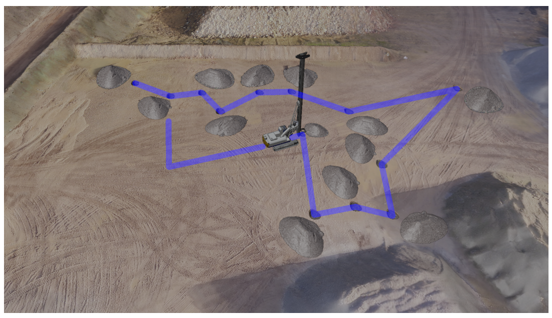

(b) Path-Conforming Circle Placement Problem (PCCP) - solution [c6]

Figure 1.2

Salesperson Problem (TSP), Capacitated Vehicle Routing Problem (CVRP) and Non-Permutation Flowshop Scheduling Problem (NPFS).

The following fifth core publication [c5] studies several novel optimization problems, originally motivated by autonomous drill rig routing in an open-pit mining application (Figure 1.2a). These problems are TSP, Hamiltonian Cycle Problem (HCP), and their relaxed variants, all of which are defined on self-deleting graphs. In self-deleting graphs, visiting a vertex results in removing a predefined set of edges, which makes the problem dynamic and highly constrained. The problems are named Travelling Salesperson Problem on Self-Deleting graphs (TSP-SD) and Hamiltonian Cycle Problem on Self-Deleting graphs (HCP-SD). They are studied both theoretically and experimentally, and a solution approach based on the generic metaheuristic solver from [c4] is proposed. The generic solver proves to be sufficiently scalable, although an exact construction procedure had to be added in order to guarantee solution feasibility even for highly constrained instances.

The last core publication [c6] extends the work [c5]. In this publication, we formulate the Path-Conforming Circle Placement Problem (PCCP) which addresses a previously unsolved aspect of the motivating mining application, namely the optimization of pile placement along a given vehicle tour (Figure 1.2b). Ultimately, TSP-SD and PCCP should be addressed simultaneously, which was beyond the capabilities of the proposed formalism for problems with permutative representation [c4] and is tackled in [r11], which is currently under review.

The rest of this thesis is organized as follows. In Chapter 2, we elaborate on the state of the art in metaheuristic algorithms in general and generic solvers similar to the one presented in [c4]. In Chapters 3-8, we briefly introduce individual core publications and present the full texts. Chapter 9 discusses the main results of individual contributions in the context of the entire thesis. Finally, Chapter 10 provides conclusions and possible future perspectives of the presented research.

Chapter 2

Related work

This chapter discusses related work relevant to the two streams of work presented in this thesis. The first stream, represented by the core publications [c1]–[c3], focuses on the design of problem-specific metaheuristic algorithms for various applications. The literature relevant to the specific applications is discussed in the core publications themselves. In Section 2.1, we provide a general overview of the development in the field of combinatorial optimization techniques suitable for our applications, especially metaheuristics. Then, we shift towards the second stream, the generic metaheuristic solver for problems with permutative representation. In Section 2.2, we discuss other existing metaheuristic algorithms for permutation-based problems and generic metaheuristic frameworks.

2.1 Selected optimization methods

The main goal of this section is to provide a non-exhaustive overview of the methods, which can be labeled highly scalable and generic, meaning that they can be applied or adapted to a wide portfolio of combinatorial optimization problems. The main areas of interest are metaheuristics and Integer Programming (IP). In both areas, we summarize the main contributions, their features, and explain the connections to this thesis.

Metaheuristics

The term "metaheuristic" was coined by Sörensen & Glover, who defined it as "a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms" [30]. In this definition, a heuristic optimization algorithm marks a problem-specific algorithm that provides an approximate solution to an optimization problem. Often, such a problem-specific instantiation of a metaheuristic is called a "metaheuristic for a particular problem" (e.g., Variable Neighborhood Search for Job Shop Scheduling Problem) or a "metaheuristic algorithm". Designing such an instantiation remains a challenging task, and a large amount of research is dedicated to applying established metaheuristics to newly formulated problems. For completeness, the term "heuristic" alone often refers to a problem-specific algorithmic component that implements a simple rule, such as, for example, a construction heuristic. In the rest of this section, we provide a non-exhaustive chronological overview of the main contributions in the field of metaheuristics. We explain the main features of individual methods, as selecting a suitable metaheuristic for a given application is crucial to the efficiency of the resulting metaheuristic algorithm. Each of these metaheuristics has countless applications in the literature. Thus, we emphasize only their applications to the problems of interest in this thesis, both ours and those of other authors.

According to [30], a **greedy constructive algorithm** is one of the oldest high-level algorithmic ideas that could, to an extent, be considered an early metaheuristic. This basic principle was instantiated in several algorithms dating as far back as the first half of the 20th century, such as the Borůvka-Sollin algorithm (1926, [31]) or the Prim-Dijkstra-

Jarník algorithm (1930, [32]). However, systematic research on metaheuristics followed after the introduction of IP and the first exact methods. Research on **evolutionary strategies** started as soon as the 1960s [33], although the first methods employed only the mutation operator and were designed as constraint satisfaction algorithms, rather than optimization ones. The modern form of **evolutionary algorithms** for optimization problems was introduced by [34] in 1975, together with the key concepts of population evolution, crossover, and selection. In this thesis, genetic and evolutionary algorithms are only rarely represented, as they are generally less scalable than single-solution metaheuristics that do not maintain a whole population of solutions. A Genetic Algorithm finished third in the IEEE WCCI competition on EVRP [27], which we address in Chapter 4.

Another early nature-inspired metaheuristic was the **Simulated Annealing (SA)** [35], introduced in 1983, which is based on performing random solution changes and accepting the resulting solutions with an acceptance probability inversely proportional to the difference between the best known and current cost value. Thus, even nonimproving solutions can be accepted with a small acceptance probability, which adds the ability to escape local optima and diversifies the search process. The acceptance probability is controlled by a so-called temperature parameter, which is decreasing according to a cooling schedule originally mimicking cooling processes in metallurgy. This mechanism typically favors diversification at the beginning of the search process and gradually shifts towards intensification. SA algorithm finished second in the IEEE WCCI competition on EVRP [27].

Tabu Search (TS) [36], introduced in 1986, added short-term memory to the search process. The memory is added using so-called tabu list, which is a fixed-size list of recently visited solutions or recently explored solution attributes. The purpose of the tabu list is to prevent cycling and repeated exploration of the same solution subspaces, which is beneficial both for increasing search efficiency and search diversification. Tabu Search is not applied to any of the thesis-related problems on its own, but it is an established mechanism that is frequently used in combination with other metaheuristics [37].

Ant Colony Optimization (ACO) [38], introduced in 1991, represents another approach how to incorporate memory into the search process. In ACO, a set of candidate solutions is iteratively constructed using a construction heuristic. This heuristic takes into account not only the heuristic information about the instance at hand, but also the information collected during the previous iterations. This information loosely corresponds to dynamically changing pheromone trails, which are used by real ant colonies seeking sources of nutrition. A metaheuristic algorithm based on ACO [39] was adapted for the EVRP, addressed in Chapter 4.

Iterated Local Search (ILS) [40], introduced in 1981, is one of the earliest neighborhood based metaheuristics. Basic ILS performs an exhaustive local search in a restricted neighborhood, which is defined by a local search operator. Here, a local search operator is a function that typically performs a minor modification of an incumbent solution, such as swapping two nodes at given positions in a solution sequence. The space of solutions reachable by all possible applications of a given operator is then called a neighborhood. Once a local optimum with respect to the neighborhood is reached, a perturbation operator is used to randomly modify the current solution and the local search is repeated. The perturbation should be strong enough to reach a different local optimum in the next local search run. Thus, ILS constantly alternates intensification and diversification. ILS is a metaheuristic repeatedly emerging in the problems addressed in this thesis. We based our solution method [s14] for the ROADEF Challenge 2022 [28] on ILS, and we also added it to the generic solver proposed in [c4]. A hybrid metaheuristic algorithm combining ILS and MILP [41] won the ROADEF Challenge 2020 [28] and a purely heuristic ILS approach finished in second place [42].

Variable Neighborhood Search (VNS) [43], introduced in 1997, substantially extended the basic ILS scheme. Rather than using a single neighborhood, the VNS performs local search sequentially in multiple neighborhoods, as the local optimum with respect to one neighborhood may not be a local optimum with respect to another one and better solutions can be found. In addition, VNS uses multiple perturbations of different strengths, which is useful for overcoming search stagnation. We won the IEEE WCCI competition on the EVRP [27] with a VNS metaheuristic algorithm, which was based on [c2] and later described in [r10]. Then, we added VNS to the generic solver proposed in [c4]. In another research stream in the IMR laboratory [44], it was also recently used to address the Multi-Agent Multi-Item Pickup and Delivery Problem [45], to schedule plan execution in the Multi-Agent Pathfinding Problem [46] or in mobile robot search planning [47].

Greedy Randomized Adaptive Search Procedure (GRASP) [48], introduced in 1995, was probably the first metaheuristic systematicaly exploiting repeated restarting. GRASP combines a greedy randomized construction heuristic with a local search. The GRASP repeatedly builds a good quality initial solution, which is then improved to local optimality by a local search engine. Randomization of the construction heuristic ensures search diversification and can be controlled by a tunable parameter. We applied GRASP in the thesis core publication [c2] and also in [r11], which extends the research presented in the core publication [c5]. GRASP was also successfully used in a hybrid metaheuristic with MILP by one of our competitors [49] in ROADEF Challenge 2020 [28], which finished 3^{rd} in the final phase. In the IMR laboratory, it was also recently used for inspection planning of structured areas by UAVs [50].

Similarly, **Large Neighborhood Search (LNS)** [51] introduced in 1998, successfully extended a basic metaheuristic often called Iterated Greedy (IG) [52], whose origin is unknown. IG repeatedly partially destroys and repairs the current solution, using a single destroy heuristic and a greedy repair heuristic. LNS follows the same principle, but employs several destroy and repair heuristics. An extension of the LNS is the Adaptive Large Neighborhood Search (ALNS), which we adapted to the ROADEF Challenge 2020 [28] competition problem described in the core publicaton [c3]. We also successfully applied GLNS [53], an adaptation of the LNS for the GTSP, to several planning problems in mobile robotics in the IMR laboratory, e.g., the thesis core publication [c1] described in Chapter 3, indoor inspection planning for a mobile robot [54] or TSP with neighborhoods in a polygonal world [55].

The presented methods are still widely used in practice, although some of them are more than 40 years old. An unfortunate ongoing trend is to invent novel metaheuristics based on various nature-inspired metaphors, such as wolf packs, bees, bacteria, bats, fireflies, or even mine blasts [30]. However, many of these attempts contribute very little to the development of the field and are often seen as a mere relabeling of existing principles by the research community [56]. A more promising current trend lies in creating hybrid metaheuristics, which may combine the advantages of different approaches. Two examples are memetic algorithms [57], combining evolutionary algorithms and local search, and matheuristics [58], combining exact and heuristic methods. Finally, there is an increasing interest in so-called hyperheuristics [59] - methods intelligently selecting or generating a suitable heuristic for a given problem.

Integer Programming (IP)

IP is a standard tool for solving combinatorial optimization problems. As mentioned in Chapter 1, IP methods have two major advantages: design efficiency and optimality guarantee. Solution methods are implemented in various solvers, and the user is only required to provide a mathematical formulation of his problem. The methods of IP are able to provide an optimality gap during the search and guarantee global optimality once the search is finished. As for the main drawbacks, IP provides complete and optimal algorithms only for problems of a certain structure, and scale worse compared to custom metaheuristic algorithms. The most common formalism is the Integer Linear Program (ILP), which can be described in canonical form as

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \\ \text{where} & \mathbf{x} \in \mathbb{Z}^n, \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n. \end{array}$$

If not all decision variables in $\mathbf{x}$ need to be integers, the problem is called the Mixed Integer Linear Program (MILP). There are also techniques for solving IP problems with a quadratic objective - Integer Quadratic Program (IQP), quadratic constraints - Integer Quadratically Constrained Program (IQCP), and their partially integral variants (MIQP, MIQCP). General cases of these problems are $\mathcal{NP}$-hard, which makes the complete IP algorithms computationally demanding and not scalable.

Concerning the solution methods, the cornerstone of an IP solver is an efficient algorithm for solving the Linear Program (LP) problem, as solving LP relaxations of ILP problems is a common step in ILP algorithms. Until today, many methods used in practice are based on two classical algorithms: the Dantzig's simplex algorithm [60] and Dikin's interior-point method [61]. Whereas the simplex algorithm has only average case polynomial-time complexity [62], the Karmarkar's algorithm [63] (a variant of the interior-point method) is provably polynomial.

When solving an ILP, there are two main classes of methods. The first one is the Branch and Bound approach (B&B), first defined by [64], which is based on partitioning the problem into smaller subproblems and eventually eliminating them through bounding. The second is the cutting-plane method, introduced by [65]. By applying additionally generated inequalities and constraints, the so-called cuts, the set of feasible solutions of the original problem is gradually reduced. These two methods are often used in combination, thus forming so-called Branch and Cut (B&C) algorithms [66]. Modern solvers implement a large number of different cuts, for example, Gomory cuts [65], clique cuts [67], cover cuts [68], disjunctive cuts [69], flow cover cuts [70], and many others [71].

In this thesis, the state of the art IP Gurobi optimizer [23] is benchmarked against our generic metaheuristic solver proposed in [c4]. It turns out that the IP solver on its own lags behind the heuristic one in terms of both scalability and solution quality on several textbook problems in fixed-time experiments, as is often the case in the literature. However, it emerges as a valuable component in several of the best-performing methods in ROADEF Challenge 2020 [28]. This competition addressed a complex industrial problem and motivated [c3]. A MILP solver was used in at least four of the 13 methods submitted to the final phase, including the winning one. MILP solver was used twice in a hybrid scheme with a metaheuristic algorithm, either to find a feasible solution or to refine the quality of the solution. In the other two cases, a MILP relaxation was used to speed up the optimization process.

2.2 Generic metaheuristic solvers and metaheuristic frameworks

In this section, we discuss the work related to the second research stream presented in this thesis, which is the generic metaheuristic solver for problems with permutative representation [c4], [s13] and its applications [c5], [c6]. First, we describe several existing metaheuristic algorithms for some subclasses of permutation-based problems and their usage specifics. Then, we discuss modular metaheuristic frameworks that are not restricted to a single problem class. The solver proposed in [c4] falls in between these two categories.

Metaheuristic algorithms for permutation-based problems

This section presents some metaheuristic algorithms that can be applied to multiple permutation-based problems with minimal adjustments. Their reusability usually relies on problem reductions, either optimum-preserving or approximation-preserving [72]. Therefore, they are similar to the generic metaheuristic solver proposed in [c4], but the representation of interest in [c4] aims to be more general and covers problems beyond the capabilities of all the methods mentioned.

An exceptionally mature solver is the LKH3 solver, which is an extension of the Lin-Kernighan-Helsgaun TSP solver [73]. It is able to solve a large number of variants of the Vehicle Routing Problem (VRP), Sequential Ordering Problem (SOP), Travelling Repairman Problem (TRP), Travelling Salesperson Problem (TSP), and others. LKH3 is based on transforming these problems into a standard TSP using a variety of optimum preserving reductions. Problem-specific penalty functions are implemented separately to handle additional constraints. LKH3 currently supports a total of 39 problems and is not designed with user-friendly extendability towards other problems in mind, as the necessary problem transformations to the TSP and back to the actually solved problem are hard-coded within the library. The objective function cannot be altered at all and is inherently tied to a static distance matrix, which greatly enhances the computational efficiency, but limits the method's versatility.

Another example is the Unified Hybrid Genetic Search for Multiattribute Vehicle Routing Problems [74]. It is a general-purpose solver consisting of problem-independent local search, genetic operators, and diversity management methods. Problem specifics are addressed by separate components for assignment, sequencing, and route evaluation. The solver is tested on 29 variants of the VRP and can be extended to others. For some problems, the solver is reported to match the performance of more specialized methods.

Addressing problems with fixed length permutation representation by Genetic Algorithms (GA) is well studied [75], [76]. Probably the most similar work to [c4] is [77], which introduced two parallel hybrid optimization methods for permutation-based problems. The addressed representation is a fixed length permutation with or without replacement. Both proposed methods combine GA with Branch & Bound technique and were built using the ParadisEO [78] framework. The work is more of a theoretical significance, as it provides an extensive overview of neighborhood and genetic operators for permutation-based problems. However, both proposed methods are tested only on the Three Dimensional Assignment Problem (Q3AP), and the implementation in its current form can be used only for other assignment problems.

Metaheuristic frameworks

This section provides examples of several existing metaheuristic frameworks. These frameworks are designed with maximal reusability in mind and usually implement a portfolio of templatized metaheuristics that can be adjusted to the problem at hand, rather than implemented from scratch. However, they still place requirements on the user going beyond problem modelling, as they are often modular and the formation of the final method is up to the user. In addition, they typically support only basic representations, such as vectors of fixed length. Custom representations can be added at the cost of implementing specialized operators or neighborhoods. A more detailed comparison covering multiple frameworks can be found in [79].

The most widely used framework according to [79] is the Java-based Evolutionary Computation Research System (ECJ, [80]). It implements a large number of state of the art algorithms such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Non-dominated Sorting Genetic Algorithm (NSGA), Ant Colony Optimization (ACO), or Particle Swarm Optimization (PSO) and supports parallelization. Problem representations are limited to several tree representations for genetic programming and various vector representations for genetic algorithms.

Another established framework is ParadiseEO [78], implemented in C++. Similarly to ECJ, ParadisEO is focused on evolutionary algorithms and enables parallelization. However, it also contains a separate module for local search algorithms and provides an automated parameter tuning. Regarding problem representations, it supports various vector representations, genetic programming tree representations, and problems representable by permutation.

More specialized frameworks exist as well. For example, the MOEA framework [81] is a Java-based framework focused on multiobjective evolutionary algorithms. It features 25 different algorithms and supports common vector representations and permutations.

Another example is the Java Metaheuristic Search Framework (JAMES, [82]), which specializes in local search metaheuristics. It implements algorithms such as Stochastic Hill Climbing, Tabu Search or Variable Neighborhood Search. The user can use a custom solution representation, but is required to provide at least one neighborhood for generating moves, which is an essential part of a local search algorithm.

Chapter 3

Path planning algorithm ensuring accurate localization of radiation sources

In this chapter, we present the first core publication called Path planning algorithm ensuring accurate localization of radiation sources [c1]. The presented research is a continuation of the author's diploma thesis [r7] and complements the research carried out in the Cybernetics and Robotics research group at CEITEC VUT Brno [83], which was presented in [84] and [85].

[c1] **Woller, D.**, Kulich, M., "Path planning algorithm ensuring accurate localization of radiation sources", *Applied Intelligence*, pp. 1–23, 2022, ISSN: 15737497. DOI: `10.1007/S10489-021-02941-Y`, **70% contribution, IF 5.3 (Q2 in Computer Science, Artificial Intelligence), citations: 1 in Web of Science, 1 in Scopus, 2 in Google Scholar.**

The paper addresses in greater detail a single subproblem of a complex robotic application. In the motivating application, a heterogeneous multirobot system is tasked with the localization of sources of gamma radiation, spread across an outdoor area. In the first phase, the entire area is scanned by a Unmanned Aerial Vehicle (UAV), and several regions are identified. In the second phase, these regions of interest are closely inspected by a Unmanned Ground Vehicle (UGV), which is tasked with highly accurate localization of radiation sources. The paper presents a path planning algorithm for the UGV that guarantees a successful localization of radiation sources and minimizes the UGV's trajectory length. The UGV is equipped with a high resolution gamma detector, which measures only the radiation count rate. To collect suitable data for accurate source localization, the UGV has to perform maneuvers along specific circular arcs in the regions of interest.

The proposed approach formulates the problem as the Generalized Travelling Salesperson Problem (GTSP) [86], which is defined in an unusual domain of specific robot maneuvers in $\mathbb{R}^2$. The goal is to visit exactly one node (perform a maneuver) from each given set of maneuvers covering certain area and minimize the total cost of the trajectory. To address this problem, the Generalized Large Neighborhood Search (GLNS) metaheuristic algorithm for the GTSP [53] is adapted. This paper presents the following contributions. First, an informed discretization procedure is proposed to transform the planning problem into the GTSP, where all sampled vertices correspond to maneuvers guaranteeing source localization in a corresponding subregion. Second, a reduction procedure that identifies and removes redundant maneuvers, not likely to be in any good quality solution, is proposed. Third, the original GLNS algorithm is adapted for the specifics caused by the use of maneuvers along circular arcs as vertices in the GTSP. The algorithm is also extended by the possibility to plan with polygonal obstacles and to take into account the robot's in-place rotation speed. Finally, a post-processing optimization procedure in the continuous space of maneuvers is proposed.

The experimental results document the effect of individual components and assess the scalability of the proposed approach on several artificial datasets.

Applied Intelligence (2022) 52:9574–9596
https://doi.org/10.1007/s10489-021-02941-y

Path planning algorithm ensuring accurate localization of radiation sources

David Woller[1,2] · Miroslav Kulich[1]

Accepted: 13 October 2021 / Published online: 7 January 2022

Abstract
An autonomous search for sources of gamma radiation in an outdoor environment is a domain suitable for the deployment of a heterogeneous robotic team, consisting of an Unmanned Aerial (UAV) and an Unmanned Ground (UGV) Vehicle. The UAV is convenient for fast mapping of the area and identifying regions of interest, whereas the UGV can perform highly accurate localization. It is assumed that the regions of interest are identified by the UAV during an initial reconnaissance, while performing a simple motion pattern. This paper proposes a path planning algorithm for the UGV, which guarantees accurate source localization in multiple preselected regions and minimizes the total path length. The problem is formulated as the Generalized Travelling Salesman Problem (GTSP) defined for discrete sets of suitable maneuvers (circular arcs), ensuring source localization in the given regions. The problem is successfully solved by a modified version of the state of the art GTSP solver, Generalized Large Neighborhood Search with Arcs (GLNS$_{\mathrm{arc}}$). Apart from adapting the GLNS, other aspects of the planning task are addressed: problem discretization and informed sampling of valid circular arcs, variants of weighting the nonrestricted trajectory segments between the arcs and postprocessing of the discretely planned trajectory in the continuous domain.

Keywords Search for radiation sources · Combinatorial optimization · Generalized Large Neighborhood Search · Generalized Travelling Salesman Problem · Heuristics · Metaheuristics

1 Introduction

Localization of radiation sources is a critical task repeatedly arising in various accidents of high seriousness. This paper focuses on localizing isolated point sources in otherwise uncontaminated areas or hot spots in large-scale accidents. Several types of incidents happened over the last century and are likely to occur again, no matter the level of technological progress and safety precautions [36]. First, there are cases of lost, stolen, or orphaned sources, commonly from an industrial or medical application. Sources were found at junkyards, abandoned factories, or even urban areas, often by unsuspecting citizens [31]. Another danger is related to the military use of nuclear energy and weapons. The U.S. alone admits 32 so-called broken arrow incidents (e.g., accidental nuclear detonation, contamination, loss in transit, or accidental jettisoning), including six cases of lost and never recovered nuclear weapons [35]. Finally, a number of accidents happened in the nuclear power industry, most notably the Chernobyl and Fukushima disasters. The initial postdisaster cleanup at Chernobyl included liquidation of highly radioactive debris, representing another relevant application.

Robotic systems are an obvious choice for radiation source localization due to the extreme risk the radiation presents to humans. As the environment can be urban or rural and indoor or outdoor, the commonly used systems are typically semiautonomous or entirely teleoperated. Unmanned Aerial Vehicles (UAVs) or Unmanned Ground Vehicles (UGVs) are deployed depending on the application, as each of the platforms has its advantages. UGVs are typically easier to navigate indoors, can operate for a longer time, and are capable of more accurate localization. On the other hand, UAVs are significantly faster and usable in

✉ David Woller
wolledav@cvut.cz

Miroslav Kulich
kulich@cvut.cz

[1] Czech Institute of Informatics, Cybernetics and Robotics, Czech Technical University in Prague, Prague, Czech Republic

[2] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, Praha 2, 121 35, Czech Republic

impassable terrain. In some applications, combining both platforms is beneficial, as their advantages can be combined.

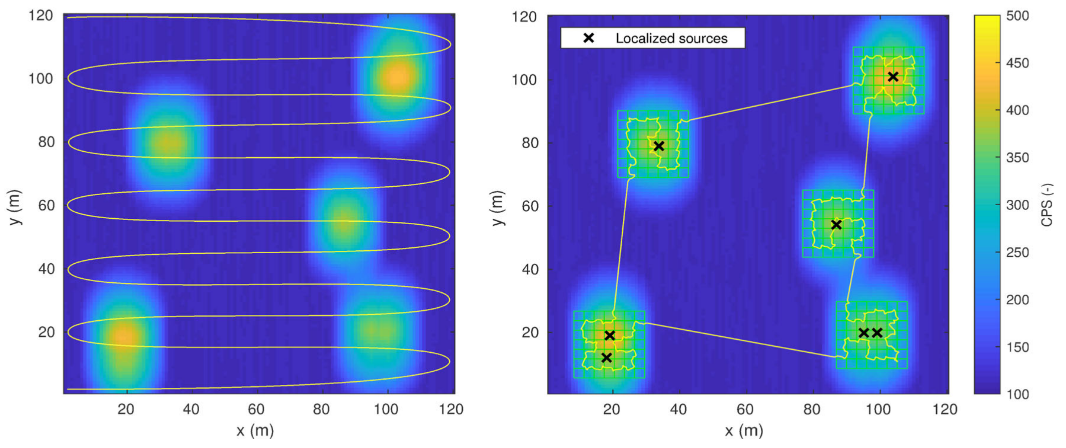
Such an approach to a semiautonomous search for sources of gamma radiation is presented in [11, 23] and [24]. The ultimate goal is to localize gamma radiation sources in an outdoor area, such as a place of a nuclear accident. The localization is to be carried out as quickly and precisely as possible, with subsequent use of an UAV and an UGV. The authors of [24] designed and constructed such a multirobot system, performed physical experiments, and evaluated the accuracy of the detection.

The considered scenario consists of two phases, which are illustrated in Fig. 1. First, the area is mapped by the UAV carrying a photogrammetry multisensor and a gamma detector. This phase's objective is to build a 3D map of the area surface and to pick regions with the potential presence of radiation sources. The UAV can carry only a lightweight gamma detector and its operation time is limited. Therefore, it is capable of detecting stronger radiation sources with insufficient precision (up to several meters). In case of weaker sources, the UAV might not be able to reliably distinguish the background radiation from source radiation, as documented in [11]. As there is no previous knowledge about the source position, the UAV performs an exhaustive search along a zig-zag trajectory while keeping an altitude of 10 meters above the terrain. The flight altitude was experimentally determined in [11] and it guarantees identifying all regions that could contain a source of activity relevant in the context of radiation protection. The minimal activity level of a potentially dangerous source was set to 10 MBq. The UAV trajectory and the discovered regions of interest are shown in Fig. 1a.

Second, the UGV with a more accurate gamma detector is deployed to inspect all previously discovered regions of interest. Thus, the scenario is not cooperative, as the UGV is deployed after the UAV finishes the initial mapping, not simultaneously. The UGV is substantially slower than the UAV, but it can operate for a longer time and locate the radiation sources more accurately. Multiple strategies such as simple zigzag pattern, Strong Source Search Algorithm [23], or Circular Algorithm [24] were proposed and tested. However, these algorithms are suitable only for single-source detection and do not utilize UAV-obtained information. This information was therefore processed by a human operator, who had to determine the regions of interest and manually define key segments (e.g., full circles of detection in the Circular Algorithm) of the UGV path. The success of this approach heavily depends on the operator experience. It does neither guarantee successful localization of all sources, nor does it optimize the UGV trajectory w.r.t. to any criteria. It is desirable to suppress the role of the operator and plan the UGV trajectory automatically and optimally w.r.t. some criteria, such as length or time.

This paper extends the preceding work by proposing a novel path planning algorithm for the UGV, which replaces the operator in the planning process and guarantees successful source detection. The goal is to plan an optimal UGV trajectory (e.g., with minimal length) or near-optimal one and to guarantee accurate localization of all radiation sources within the preselected regions of interest. An example of such a trajectory is shown in Fig. 1b, together with the detected positions of the radiation sources.

The UGV is assumed to be mounted with relatively inexpensive high-resolution gamma detectors, which measure only the count rate. A single radiation source's position can be reliably detected with such a setup by performing a specifically constrained maneuver in its close



(a) UAV trajectory and identified regions (b) UGV trajectory, searched subregions and detected sources

Fig. 1 Radiation intensity map produced by the UAV and illustration of the two-phase search

neighborhood. If this maneuver is a circular arc, and multiple detectors are mounted on the robot, a directional profile of the radiation can be constructed and used for accurate localization. Measuring along circular arcs is advantageous, as it results in steeper characteristics of the measured data. Thus, the peaks in the measured radiation intensity are more prominent and can be better distinguished from fluctuations in background radiation. Moreover, measurements along circular arcs can be easily interpolated, which is useful for determining the source position from the measured data. This setup was successfully deployed in [27, 34], or [23]. Task-specific conditions on the usability of the circular arc maneuvers are described in Section 2.2.4.

As the UAV-preselected regions of interest can be arbitrarily large and one circular arc may not be sufficient for covering a whole region, each region is divided into smaller subregions of fixed size (green squares in Fig. 1b). An unlimited number of valid maneuvers for exploring a subregion exists; therefore, infinitely many valid circular arcs can be sampled. Besides that, the order of subregion exploration is not fixed and is also subject to optimization. Therefore, after appropriate discretization of individual subregions, the planning problem can be reformulated as NP-hard Generalized Travelling Salesman Problem (GTSP). Given n nodes (circular arcs) divided into m sets (subregions), the goal is to find such a trajectory that passes through exactly one node from each set and is optimal with respect to some criterion, e.g., minimum length. The modified variant of the GTSP with circular arcs as vertices is from now on referred to as the GTSP_{arc}.

This paper presents an approach to the discretization of the planning problem, introduces GLNS_{arc} algorithm solving the discrete GTSP_{arc}, evaluates its performance in several experiments and proposes two improvements to the GLNS_{arc} functionality.

The contribution lies in multiple aspects.

- **Problem formulation.** The planning task is formulated as a discrete optimization problem, specifically a variant of the GTSP and called GTSP_{arc}. It is shown how to tailor the state of the art GLNS metaheuristic to solve it.
- **Automation of the UGV path planning.** The current solution relies on a human operator when processing the UAV-obtained information and specifying critical segments of the UGV trajectory. The algorithm presented in this paper replaces the operator role in this phase and enables to generate the UGV path automatically. The algorithm is meant to be run on a base station, after the initial UAV reconnaissance.
- **Informed discretization.** An infinite number of valid vertices can be sampled for each subregion, which increases the computation and memory demands. A method for detecting sampled vertices that are valid, but not useful in any potential solution is presented. Thus, the discretized problem size can be reduced considerably, which speeds up the planning and allows for covering much larger areas.
- **Guarantee of detection in regions of interest.** It is shown how to sample valid vertices that ensure source detection in predefined regions of interest. Currently, this is reliant on the operator experience.
- **Post-processing in the continuous domain.** The proposed GLNS_{arc} algorithm is capable of finding a locally optimal solution in the discrete domain. However, this solution may not be locally optimal in the continuous domain. Therefore, a local search procedure in the continuous domain is introduced. This procedure is deployed once in postprocessing, after the GLNS_{arc} finishes.
- **Optimality criteria.** The algorithm is capable of minimizing a custom optimality criterion, which can be difficult for the human operator. This criterion is determined by the weights assigned to edges in the GTSP_{arc} planning graph. Besides the standard weighting based on Euclidean distance, a more realistic weighting is introduced, which also considers the robot's rotation speed. It is also demonstrated that GLNS_{arc} can be deployed in an environment with obstacles or impassable terrain, both indoor and outdoor.

The paper is structured as follows. Section 1 contains this introduction and state of the art summary in Section 1.1. Section 2 provides the theoretical description of the problem and describes the exact steps of the solution. Sections 2.1 and 2.2 give the formal definition of the planning problem, as well as the specifics introduced by the application. Section 2.3 presents the employed GTSP solver - GLNS. Finally, Sections 2.4 to 2.7 tackle various aspects of the solution - most notably the necessary GLNS modifications, variants of edge weighting, an approach to the problem discretization and a local optimization technique in continuous space called DenseOpt. Section 3 then examines the behavior of the algorithm and evaluates the contribution of individual components of the solution. A concise summary and conclusion is given in Section 4.

1.1 State of the art

Concerning gamma radiation monitoring and search for radiation sources, there are various scenarios and therefore, various respective approaches. For the long term, static monitoring in places with a higher risk of radiation leak (e.g., a nuclear power plant, medical or industrial facilities), permanent sensor networks are typically installed, while the current research is focused on developing reliable wireless sensor networks [7] or mobile sensor networks [26].

However, when the radiation source presence is not anticipated in the area or the area is too large or unsuitable for sensor network installment, a search mission deploying single or multiple measuring agents is a more adequate solution. Then, the motion planning becomes a critical factor in ensuring that the whole area is inspected and that the inspection is carried out in a reasonable time. The search process is often referred to as source seeking (SS) and determining the source position from the measured data as source term estimation (STE). Apart from radiation source localization, similar methods are also used for identifying leaks of gas or chemicals [25]. The SS motion planning methods can be classified either as adaptive or proactive [37]. In the case of adaptive methods, the motion planning is typically controlled by a feedback from the current output of STE, which is beneficial mainly for controlling the STE accuracy. Commonly used adaptive methods are gradient-based traversal, surging or casting [2]. These methods typically contain a simple mechanism for switching from one sensing location to another (towards another potential source), but this approach does not deal well with the intrinsic combinatorial problem, which is determining the order of exploration of individual regions. Therefore, they are suitable for localizing only a single source or for usage by a swarm of robots initially equally distributed across the area [8].

Proactive strategies are on the other hand designed to ensure full coverage of the area at the cost of higher time requirements and possibly lower accuracy. Several rather simple motion planning strategies, such as the zig-zag pattern [39] or contour mapping [13], were proposed for the outdoor environment. The main advantage of aerial spectrometry is the possibility of quickly exploring a relatively large area, while the main disadvantage is the low accuracy of source position estimation. For more accurate or indoor mapping, a ground-based agent such as a UGV can be employed. Compared to a UAV, the UGV typically moves at significantly lower speeds and has to deal with obstacles and impassable terrain. With no obstacles present or considered, some motion planning strategies used in aerial spectrometry were adapted to UGVs (e.g., the zig-zag pattern). However, these are not suitable in many real world applications and more complex motion planning strategies are needed. For example, [33] proposed an approach for an enclosed polygonal environment. This method first performs a convex polygon partitioning by splitting the environment into smaller regions explorable from a single position. Then, it determines the dwell time needed for staying in each of these positions using the Currie limit of detection [5] and plans a trajectory over all positions, ensuring exploration of the whole area. Similar approach was followed by [1], which formulated the multigoal mission as the Travelling Salesman Problem over a set of predefined measurement locations. The main difference to the GTSP$_{\text{arc}}$ is that the planning is performed over a set of predefined locations of static measurements, where the robot needs to stop. In the GTSP$_{\text{arc}}$, the robot takes useful measurements while constantly moving, so the GLNS$_{\text{arc}}$ plans over a set of maneuvers, rather than positions. In other words, the existing approaches directly solve the TSP or the GTSP over a fixed set of points while connecting these points with trajectory segments corresponding to a particular vehicle model, as in, e.g., the Dubins GTSP [19] are not directly applicable in GTSP$_{\text{arc}}$.

Another family of approaches, information sampling (IS) techniques [3, 17] extensively sample the robot configuration space to find a path/trajectory minimizing a given objective function. The number of samples and thus the computational complexity, however, grows exponentially with the number of dimensions. The configuration space of maneuvers in GTSP$_{\text{arc}}$ has six dimensions, in contrast to two or three dimensions in typical applications of IS. Therefore, even small GTSP$_{\text{arc}}$ instances will be hard to solve with current IS techniques. Moreover, the search in IS is restricted to a limited time horizon, while in the GTSP$_{\text{arc}}$ we plan a path that visits all regions.

When using both a UAV and a UGV in an outdoor environment, the advantages of both platforms can be combined. The UAV can be used for the fast yet inaccurate estimate of radiation source locations and the UGV for subsequent accurate localization. Therefore, there is no need for time-consuming exploration of the whole environment by the UGV. An example of such an application is presented in [4], where the area is first inspected by a UAV, which collects RGB images and radiation data of the area. Based on the collected data, a cost map for ground-based motion is created, and the UGV is then deployed to inspect regions of interest using repeatedly the A* algorithm (thus being suitable for detecting a single radiation source).

As the GTSP$_{\text{arc}}$ is an extension to the GTSP, and a GTSP solver was employed and modified in order to solve the GTSP$_{\text{arc}}$, this subsection mainly discusses state of the art in GTSP solvers. The GTSP is a combinatorial optimization problem extensively studied in operations research with many practical applications, such as location routing problems, material flow system design, post-box collection, stochastic vehicle routing or arc routing [21].

There are multiple methods of finding the optimal GTSP solution in exponential time. The GTSP can be modeled as an integer linear program (ILP) and solved with an ILP solver. One of the first formulations was introduced in [22]. Formulations studied by [10] then inspired the problem-specific exact branch-and-cut algorithm [9]. Another six different integer programming formulations are compared in [30], with an emphasis on 'compact' formulations (i.e., formulations, where the number of constraints and variables is a polynomial function of the number of nodes in the GTSP). A commonly used approach is to transform the GTSP into the TSP using the Noon-Bean transformation [28]. Then, an exact TSP solver such as the Concorde [18] can be deployed.

Due to GTSP NP-hardness, new approaches on how to find acceptable solutions to large problem instances in polynomial time are still being proposed. The problem can be transformed into the TSP and solved with a heuristic-based TSP solver, e.g., the LKH [14], but numerous heuristic approaches were adapted specifically for the GTSP. Among these, the following three algorithms can be considered as the most successful. The first one is the GLKH solver [16], which is based on the Lin-Kernighan k-opt heuristic used in the LKH. The GLKH is reported to be tested on large-scale instances with up to 17180 sets and 85900 vertices, and it is focused on finding high-quality solutions. The second one is a memetic GK heuristic proposed in [12], which combines genetic algorithms with a local search procedure. The GK yields high-quality solutions with excellent runtime on medium-size GTSP instances, but it does not scale well for problems with more than 200 sets [6]. Finally, Smith and Imeson presented an algorithm combining adaptive large neighborhood search, simulated annealing, and two local search procedures. The algorithm is called Generalized Large Neighborhood Search (GLNS), and it is documented to often outperform both GLKH and GK on several GTSP libraries [32]. It dominates the other two solvers, especially on highly constrained nonmetric instances, whereas the GLKH performs best on the largest clustered Euclidean instances and the GK on medium size metric instances. The proposed approach to GTSP$_{\text{arc}}$ is built on GLNS due to its most consistent performance over a broad portfolio of GTSP instances compared to the other two solvers.

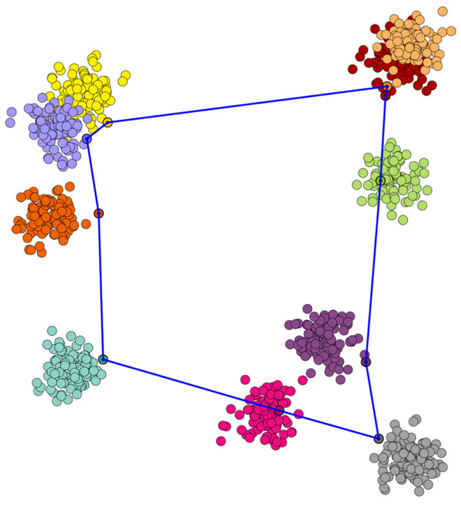


Fig. 2 Solved GTSP instance 10C1k.0 from MOM-lib [15]

2 Methods

2.1 Planning task formulation

This subsection provides a formal definition of the GTSP and elaborates on the complications arising from differences between the practical task of searching for radiation sources and the following standard GTSP definition.

Problem 1 (The Generalized Travelling Salesman Problem) Assume a complete weighted graph $G = (V, E, w)$ and a partition of V into m sets $P_V = \{V_1, ..., V_m\}$, where

- V is a set of n vertices
- $V_i \cap V_j = \emptyset$ for all $i \neq j$
- $\bigcup_{i=1}^{m} V_i = V$
- E is a set of edges such that all vertices are connected, apart from vertices from the same set
- w is a mapping assigning a weight to each edge $w : E \rightarrow \mathbb{R}$

Lets also define a tour T over graph G as a closed sequence of vertices and edges $T = (v_0, e_0, .., v_{m-1}, e_{m-1})$, where each edge connects two consecutive vertices - $e_i = (v_i, v_{i+1})$ and $e_{m-1} = (v_{m-1}, v_0)$. A set of vertices present in the tour T is denoted as V_T, a set of edges as E_T.

Then, the objective is to find a tour in G that contains exactly one vertex from each set and has a minimum length, i.e., it minimizes the tour length $w(T)$ defined as

$$w(T) = \sum_{e \in E_T} w(e).$$

An example of a solved GTSP instance is shown in Fig. 2, where vertices of the same color belong to the same set.

2.2 Application specifics

There are aspects of the planning task solved that prevent us from directly using the previously given GTSP formulation and already implemented solvers. Due to the following, the formulation has to be slightly changed, and the solver is appropriately modified.

2.2.1 Vertex definition

Contrary to GTSP, where a vertex is typically a point in 2D, a single vertex in the GTSP$_{\text{arc}}$ represents a circular arc -

Table 1 Vertex parameters in the GTSP$_{\text{arc}}$

Symbol	Parameter description
x, y	planar coordinates of circular arc center (m, m)
r	arc radius (m)
α	angle between coordinate frame x-axis and arc axis (rad)
ω	angular size of the arc (rad)

a special trajectory segment such that passing by it allows a precise radiation source detection in a corresponding subregion. A minimal vertex representation consists of the parameters given in Table 1, which are also depicted in Fig. 3a.

2.2.2 Vertex weighting

As each vertex represents a segment of a robot trajectory, its length influences the total trajectory cost. In the standard GTSP definition given in Section 2.1 only edges, not vertices, have weights assigned. Therefore, this difference must be taken into consideration while implementing various metrics in the algorithm. In GLNS$_{\text{arc}}$, the weight w of each vertex is defined as the length of the arc: $w = \omega r$.

2.2.3 Vertex connecting

There is no restriction on the direction in which a vertex is to be passed, therefore connecting two vertices is ambiguous. The original algorithm considers only the possibility that edge weights depend on the vertex order - an edge from a to b might have a different weight than an edge from b to a (asymmetric GTSP). However, in the GTSP$_{\text{arc}}$, even with a fixed order, there are still four ways to connect two consecutive vertices, as shown in Fig. 3b.

2.2.4 Vertex validity constraint

While detecting sources of radiation along circular arcs, only some arcs are potentially useful. Here, the vertices are generated to fit the experimental setup presented in [24], which uses a UGV fitted with a GPS unit, two mutually shielded gamma detectors, and counting electronics. The detection system is capable of measuring the rate of gamma radiation in counts per second (CPS) and recording the position of the measurement. It is partially directional due to the use of two detectors - when moving along a curve, it can be determined on which side of the curve the radiation source is located.

The STE procedure used for determining source locations based on the measurements taken is called the Circular Algorithm; it is described in [23] in detail and works as follows. The theoretical shape of the radiation intensity I_m measured in CPS along a circle near a radiation source can be described (according to the inverse square law and the law of cosines) as

$$I_m(\phi) = \frac{I_s}{a^2 + r^2 - 2dr\cos\phi}, \tag{1}$$

where I_s states the source intensity in CPS in a distance of 1 meter from the source, a is the distance from the source to the circle's center, r is the radius of the circle, ϕ is the difference between the angular coordinates of the UGV and the source with respect to the circle center and d is the distance from the source to the closest circle point. The function given in (1) can be used to interpolate measurements taken along a circular segment, although according to [23], quadratic interpolation is sufficient for determining the location of the intensity peak and thus the direction towards the radiation source. As it is also known, whether the source lies inside or outside of the circular segment (or better, the corresponding full circle) and which subregion is covered by this particular segment (see Section 2.6.1), an initial estimate of the source position

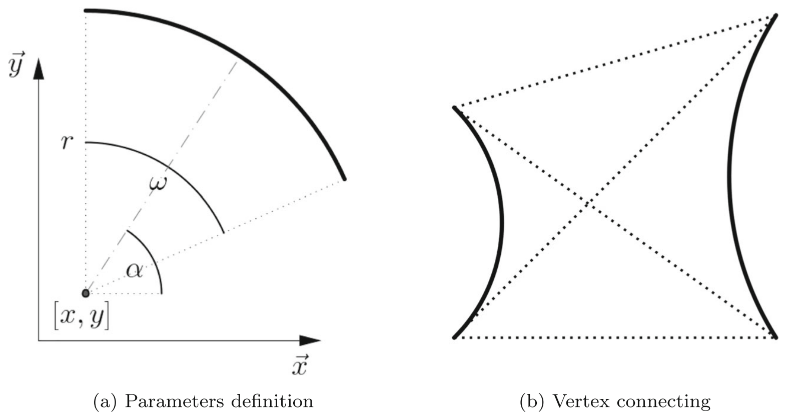


Fig. 3 Vertices (arcs) in the GTSP$_{\text{arc}}$

Table 2 Vertex validity constraint - parameters

Symbol	Parameter description
r	radius of circular arc (m)
d	distance from the source to the closest segment point (m)
l	arc length (m)
θ	angle between arc axis and line from arc center to source (rad)
I_s	intensity of the radiation source in CPS (-)
I_B	background radiation in CPS (-)
K	experimentally set constant (-)

and intensity can be made. This estimate is then iteratively improved by applying the Gauss-Newton method, which is reported to achieve an average accuracy of 17.8 cm RMS (root mean square) in real-world experiments carried out by the authors of [23].

It must be distinguished whether the peak in radiation intensity measured along a circular segment corresponds to a radiation source or fluctuations in the radiation background. Given a source $s = [x_s, y_s]$ with an intensity I_s, a robot position $p = [x_p, y_p]$ and a level of background radiation I_B, the intensity I_m measured by the robot is equal to

$$I_m = \frac{I_s}{(x_s - x_p)^2 + (y_s - y_p)^2} + I_B$$

according to the inverse square law. Then, the ratio between the minimal and maximal intensity I_m measured along a circular segment must be higher than a constant K called prominence [24] to identify a radiation source. For a circular segment, this constraint can be rewritten as

$$\frac{\frac{I_s}{2r(r+d)\left(1-\cos\left(\frac{l}{2r}-|\theta|\right)\right)+d^2} + I_B}{\frac{I_s}{d^2} + I_B} < K. \qquad (2)$$

Individual parameters are described in Table 2 and their numerical values used for vertex generation are given in Table 3 (if constant). The values of I_s and I_m are in practice estimated from the initial UAV measurements. In Fig. 1, I_s corresponds to the highest CPS values at the yellow hot spots and I_B to the lowest readings in the dark blue areas. The measured intensity I_m is taken along the yellow line, marking the robot trajectory in Fig. 1b.

Table 3 Parameter values used for V_{val} generation

Parameter	Values
$x(m)$	-5 to 5 sampled by 1
$y(m)$	-5 to 5 sampled by 1
$r(m)$	1 to 5 sampled by 1
$\alpha(rad)$	$\pi/6$ to 2π sampled by $\pi/6$
$\omega(rad)$	$\pi/6$ to π sampled by $\pi/6$
$I_s(-)$	6000
$I_B(-)$	150
$K(-)$	0.7

2.3 Generalized Large Neighborhood Search (GLNS) description

GLNS is a GTSP solver introduced by S. L. Smith and F. Imeson [32]. This subsection gives only a brief overview of its functioning. A full description of the algorithm, including a thorough performance comparison with other approaches, can be found in [32].

Algorithm 1 GLNS.

```
Data: A GTSP instance $(G, P_V)$
Result: GTSP tour $T$
1  for $i \leftarrow 1$ to cold_restarts do
2      $T \leftarrow$ initial_tour$(G, P_V)$
3      $T_{best,i} \leftarrow T$
4      Initialize the acceptance temperature $\tau$
5      repeat
6          Select a removal heuristic $R$ and an insertion heuristic $I$
7          Select number of vertices to remove $N_r$
8          $T_{new} \leftarrow T$
9          Remove $N_r$ vertices from $T_{new}$ using $R$
10         Insert $N_r$ vertices to $T_{new}$ using $I$, one from each set not visited by $T_{new}$
11         Locally re-optimize $T_{new}$ (MoveOpt, ReOpt)
12         if $w(T_{new}) < w(T_{best,i})$ then
13             $T_{best,i} \leftarrow T_{new}$
14         end
15         if accept$(T_{new}, T, \tau)$ then
16             $T \leftarrow T_{new}$
17             Record improvement made by $R$ and $I$
18         end
19     until stop criterion met
20     Update selection weights of heuristics
21     Update the acceptance temperature $\tau$
22 end
23 return tour $T_{best,i}$ that attains $\min_i w(T_{best,i})$
```

GLNS implements an adaptive large neighborhood search, which is a metaheuristic planning approach based on the iterative application of constructive and destructive procedures to a current solution. These procedures are being selected randomly, using the roulette wheel selection mechanism. The selection weights of individual procedures are dynamically adjusted throughout the search process, according to their success in previous iterations. Each time a procedure improves the currently best solution, its selection weight after the next restart is increased and vice versa.

GLNS pseudocode is given in Algorithm 1. First, an initial random tour is generated (line 2). Then, the following process runs iteratively. A pair of removal and insertion heuristics is selected according to their selection weights (line 6). These heuristics are then applied to remove N_r vertices from the current tour T and insert different N_r vertices, thus creating a modified tour T_{new} (lines 7 to 10). The modified tour T_{new} is subject to the local optimization techniques MoveOpt and ReOpt (line 11) and is consequently accepted or declined, while using a simulated annealing criterion (line 15). This process repeats until one of the stop criteria is met (line 19). After that, the planner updates the selection weights of the heuristics (line 20) and either starts the whole process again with a new cold restart or returns the best tour found overall.

The acceptance criterion (line 15) uses a simulated annealing procedure, which allows for accepting nonimproving tours T_{new} with a small probability depending on the temperature τ. The temperature τ is initialized at the beginning of each cold restart (line 4) and then gradually decreases in so-called initial descent. Then, after a certain number of nonimproving iterations, the temperature is increased again. This is called a warm restart, which also ends after a fixed number of nonimproving iterations. The temperature updates are performed at the end of each iteration (line 21). Each cold restart then consists of an initial descent and several warm restarts.

2.4 Proposed GLNS modifications towards GLNS$_{\text{arc}}$

As described in Section 2.2, graph vertex in the GTSP definition corresponds to a circular arc in the GTSP$_{\text{arc}}$. This arc represents a part of a trajectory and has certain properties, that have to be taken into account while employing GLNS to solve the GTSP$_{\text{arc}}$. The main issues arise from the fact, that the arc has a nonzero length and that connecting two vertices is ambiguous. Necessary modifications to the algorithm solving these issues are described in detail in this subsection. To prevent confusion, the word vertex is used when talking about circular arcs (i.e., the graph vertices in GTSP$_{\text{arc}}$) from now on.

2.4.1 Vertex duplication

In Section 2.2, a vertex is described by this tuple of parameters - $\langle x, y, r, \alpha, \omega\rangle$. This representation is sufficient for problem formulation but impractical for implementation, as it requires distinguishing between various ways of vertex connecting. Instead of doing that, an additional parameter $sign \in \{\pm 1\}$ is added. This parameter determines in which direction the vertex is to be passed through (-1 for clockwise passage, +1 for anticlockwise). Naturally, this doubles the total number of vertices in GLNS$_{\text{arc}}$ algorithm, as each vertex is inserted with both possible $sign$ values. On the other hand, the problem with edge connecting is solved, because the original GLNS allows for solving asymmetric GTSP, which is the case now.

2.4.2 Tour weight

Let $T = (v_0, e_0, v_1, e_1, ..., v_{m-1}, e_{m-1})$ be a tour and $w(T)$ its weight. In GLNS$_{\text{arc}}$, this weight is calculated as

$$w(T) = \sum_{i=0}^{m-1} w(e_i) + \sum_{j=0}^{m-1} w(v_j), \tag{3}$$

where v_i, e_j are the vertices and edges from T and $w(v)$, $w(e)$ are their respective weights. The bold expression in (3) and in all the following equations is newly added in GLNS$_{\text{arc}}$, and it reflects the fact that vertices have nonzero weight.

2.4.3 Cheapest insertion and unified insertions

GLNS contains several insertion heuristics that add vertices from currently unused sets to an incomplete tour T according to some simple rules. In all of these heuristics, the insertion cost of a vertex $v_{new} \in V_i$, $V_i \in P_V \setminus P_T$ is minimized. Here, P_T contains those sets, that are already visited by T. In the cheapest insertion, this cost is minimized to select both V_i and v_{new}, whereas, in the remaining three unified insertions, V_i is already selected by a different mechanism and only v_{new} is sought. In all cases, the insertion cost c_{ins} has to be modified to take the weight of v_{new} into account:

$$c_{ins} = w(v_j, v_{new}) + w(v_{new}, v_{j+1}) - w(e_j) + w(v_{new}).$$

Here, v_j and v_{j+1} are two consecutive vertices in T before insertion and $w(v_j, v_{new})$, $w(v_{new}, v_{j+1})$ and $w(e_j)$ weights of corresponding edges.

2.4.4 Worst removal

Similarly to insertion heuristics, GLNS contains several removal heuristics, removing vertices from a tour T while using some simple rules. Worst removal removes such vertex $v_j \in V_T$ from tour T, that maximizes the removal cost c_{rem}. GLNS$_{\text{arc}}$ modification is again rather straightforward:

$$c_{rem} = w(e_{j-1}) + w(e_j) - w(v_{j-1}, v_{j+1}) + w(v_j).$$

2.4.5 ReOpt

Re-Opt, which is a local optimization subroutine, attempts to optimize the choice of vertices while keeping the set order fixed. This is achieved by performing a graph search through all sets, in which only edges between two consecutive sets are considered. When expanding from vertex $x \in V_i$ to vertex $y \in V_{i+1}$, current score in y is calculated as

$$score(y) = score(x) + w(x, y) + w(y).$$

Moreover, the first vertex $a \in V_1$ is initialized with $score(a) = w(a)$, instead of zero.

2.4.6 MoveOpt

MoveOpt subroutine attempts to optimize the set order by randomly removing a vertex v_i from a tour T and reinserting another vertex v_j from the same set to any position in the tour so that the insertion cost is minimized. This cost c_{ins} is modified the same way as in the cheapest and unified insertions, i.e.

$$c_{ins} = w(v_j, v_{new}) + w(v_{new}, v_{j+1}) - w(e_j) + w(v_{new}).$$

2.4.7 Remarks

Some parts of GLNS were not formally modified, although the original idea behind them might have changed in GLNS$_{\text{arc}}$ due to task reformulation.

Set-vertex distance from a set V to a vertex u is still defined as

$$dist(V, u) = \min_{v_i \in V}(\min(w(u, v_i), w(v_i, u))).$$

In GLNS$_{\text{arc}}$, these distances are precomputed after vertex duplication. Therefore, the value obtained corresponds to the shortest path to or from u to V, no matter the $sign$ of u, v (= their orientation) or the edge (u, v) direction.

A different situation arises in the distance removal heuristic. The motivation is to remove vertices from a current tour T, which are "close to each other". At each iteration, a vertex v_{seed} is selected randomly from the set of already removed vertices V_{rem}. The next vertex v_j to be removed from T is obtained as

$$v_j = \arg\min_{v_j \in T}(\min(w(v_{seed}, v_j), w(v_j, v_{seed}))).$$

Here, GLNS$_{\text{arc}}$ considers only edges between vertices v_{seed}, v_j and not their oppositely oriented variants available in the GTSP$_{\text{arc}}$ instance, as these variants are not present in T.

2.5 Edge weighting variants

Two variants of edge weighting in the GTSP$_{\text{arc}}$ graph were designed in the GLNS$_{\text{arc}}$. The first one ($line$) calculates

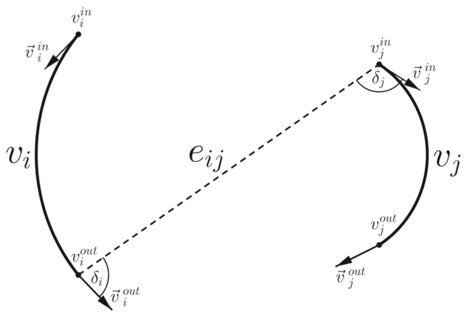


Fig. 4 Edge parameters

the edge weight as the Euclidean distance between the connected vertices endpoint and entry point. This metric is commonly used in GTSP and similar problems, but it may be of limited value in GTSP$_{\text{arc}}$, as it does not reflect the rotation time needed at the vertex endpoints. Two edges with a similar weight may thus result in considerably different trajectory execution times. The second variant ($lineWA$) is designed to reflect this, as it also considers the rotation time needed at each endpoint.

Let v be a vertex in GTSP$_{\text{arc}}$, with parameters as in Table 1. The vertex is oriented according to $sign$. Therefore, it has an entry point v^{in} and a leaving point v^{out}. The robot orientation in these points is then given by the tangent vector $\mathbf{v}^{in}$, respectively $\mathbf{v}^{out}$, as shown in Fig. 4. Definitions of individual edge weighting variants follow.

2.5.1 line

Let us consider vertices v_i, v_j connected by an edge e_{ij} (from v_i to v_j). The edge weight is then calculated as

$$w(e_{ij}) = \|v_i^{out} - v_j^{in}\|$$

2.5.2 lineWA - line with weighted angles

Similarly to the $line$ type, this variant connects vertex endpoints with a straight line. Let us again consider vertices v_i, v_j and an edge e_{ij}. Then, let δ_i be the smaller angle between vector $\mathbf{v}_i^{out}$ and e_{ij} (and δ_j between $\mathbf{v}_j^{in}$ and e_{ij} respectively), as shown in Fig. 4. The edge weight is then defined as

$$w(e_{ij}) = \|v_i^{out} - v_j^{in}\| + k(\delta_i + \delta_j), \tag{4}$$

where k is an application specific constant, reflecting the robot in-place turning speed.

2.6 Problem discretization

As was explained in Section 1, a single set of vertices in the GTSP$_{\text{arc}}$ corresponds to a subregion with the potential presence of a radiation source. A vertex in the GTSP$_{\text{arc}}$ then corresponds to a maneuver along a specific circular arc. Equation (2) restricts the parameters of the circular arc, so that accurate localization of the radiation source is guaranteed. This constraint needs the source position to be known; therefore, it cannot be directly applied when sampling valid arcs covering the whole subregion - precise source position is not known yet. Thus, the following straightforward approach was adapted.

2.6.1 Sampling sets - covering subregions with vertices

The procedure of sampling valid vertices in GTSP$_{\text{arc}}$ is described in Algorithm 2.

Algorithm 2 Sampling of valid vertices.

```
   Data: Subregion $R$, ranges of $x$, $y$, $r$, $\alpha$ and $\omega$
   Result: Set of valid vertices $V_{val}$
1  $V_{val} \leftarrow \emptyset$
2  Uniformly sample $p$ sources $s$ within $R$
3  for $x \in$ range($x$) do
4      for $y \in$ range($y$) do
5          for $r \in$ range($r$) do
6              for $\alpha \in$ range($\alpha$) do
7                  for $\omega \in$ range($\omega$) do
8                      Generate vertex
                       $v \leftarrow v(x, y, r, \alpha, \omega)$
9                      $valid \leftarrow true$
10                     for $i \leftarrow 1$ to $p$ do
11                         if $\neg$ constraint($s_i, v$) then
12                             $valid \leftarrow false$
13                             break
14                         end
15                     end
16                     if $valid$ then
17                         $V_{val} \leftarrow V_{val} \cup \{v_{sign=1}\}$
18                         $V_{val} \leftarrow V_{val} \cup \{v_{sign=-1}\}$
19                     end
20                 end
21             end
22         end
23     end
24 end
25 return $V_{val}$
```

The algorithm takes the following inputs: finite sets of possible vertex parameter values x, y, r, α, and ω (described in Table 1) and a subregion R. First, the subregion R is uniformly densely covered with p potential source positions s (line 2). Second, a circular arc is generated for each combination of given parameter values (line 8). If the newly generated vertex satisfies (2) for all p source positions, the arc is assumed to be valid for the whole subregion (line 9 to 13). The newly generated valid vertex v is then added to the set of valid vertices V_{val} in both orientations given by $sign$ (lines 16 to 18).

2.6.2 Reducing set size

The sampling procedure described in Section 2.6.1 generates a set of valid vertices V_{val}, whose size grows rapidly, proportionally to the range of input parameters x, y, r, α and ω. However, some vertices sampled may not be actually useful. A procedure for utilization-based V_{val} reduction is proposed in this section.

The contribution of some vertex v_i from a set V_{val} to the total tour weight $w(T)$ depends only on its neighbors v_{i-1} and v_{i+1} in a tour T. If some vertex $v_i \in V_{val}$ does not minimize the partial cost

$$c_{par} = w(v_{i-1}, v_i) + w(v_i) + w(v_i, v_{i+1}) \tag{5}$$

for any possible configuration of its neighbors v_{i-1} and v_{i+1}, it is redundant and can be removed from V_{val}.

To perform the vertex utilization analysis, a square grid of points is generated around the set V_{val}. An example of such a grid applicable for edge type $line$ is shown in Fig. 5a. Points in this grid correspond to the endpoints of vertices v_{i-1}, v_{i+1} and they are sufficient for determining the relevant edge weights $w(v_{i-1}, v_i), w(v_i, v_{i+1})$. A sufficient set of potentially usable vertices V_{suf} is then extracted from V_{val} by testing all possible combinations of these endpoints and keeping only those vertices from V_{val} minimizing (5) for any of these combinations. This process is described in Algorithm 3.

Algorithm 3 Utilization-based set reduction.

```
   Data: Original set $V_{val}$, grid parameters $P$
   Result: Reduced set $V_{suf}$
1  $V_{suf} = \emptyset$
2  Generate grid $G = G(P)$ around $V_{val}$
3  for $p_1, p_2 \in G \times G$ do
4      $c_{best} \leftarrow \infty$
5      $v_{best} \leftarrow null$
6      for $v \in V_{val}$ do
7          $c_{par} \leftarrow w(p_1, v) + w(v) + w(v, p_2)$
8          if $c_{par} < c_{best}$ then
9              $c_{best} \leftarrow c_{par}$
10             $v_{best} \leftarrow v$
11         end
12         $V_{suf} \leftarrow V_{suf} \cup v_{best}$
13     end
14 end
15 return $V_{suf}$
```

The accuracy of this method depends on the set of grid parameters P. For the edge type $line$, where edge weight

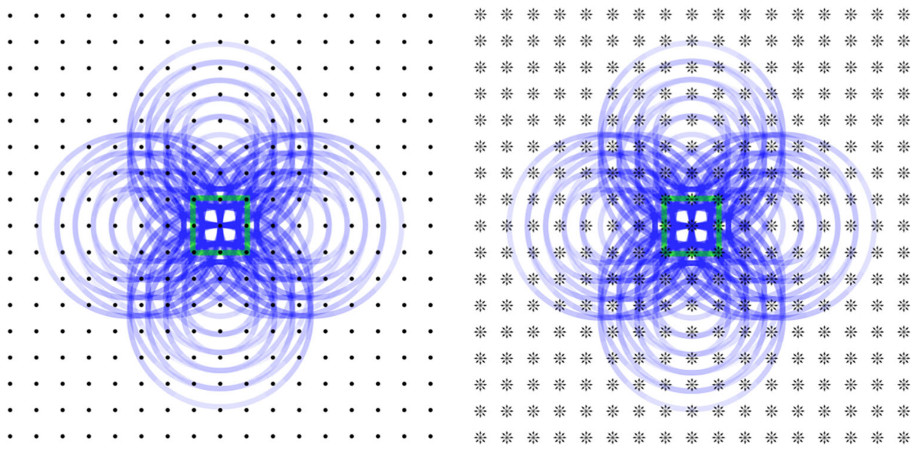


Fig. 5 Grids used for reducing V_{val} size

is calculated as the Cartesian distance between vertices endpoints, these parameters are grid size $size$ and grid resolution $posRes$. A point in the grid then corresponds to a point in 2D. In the case of the $lineWA$ type, which also considers the amount of rotation needed in each endpoint, a point in the grid consists of a point in 2D and a vector, as shown in Fig. 5b. Thus, there must be an additional parameter $angleRes$, which determines the angular resolution used when sampling multiple vectors in different directions from each point.

Suitable values of these parameters depend on the set V_{val} and on the edge type used. A simple method for extracting a sufficient set V_{suf} while refining the parameters P until the size of V_{suf} converges is proposed and described in Algorithm 4. First, V_{suf} is extracted using the reduction procedure described in Algorithm 3 with initial parameters P (line 1). Each parameter from P is then repeatedly refined according to the refinement step from Table 4 (line 5) and the V_{suf} is extracted again (line 6). Refining of the current parameter stops when there is no increase in the size of V_{suf} (line 8).

As the parameters are tuned sequentially, it may happen that refining the latter parameter enables further refining and subsequent increase in V_{suf} size for some of the preceding parameters. To cover this possibility, Algorithm 4 can be run repeatedly with the previously found parameters used as an initial solution in the next run, until there is no change in V_{suf} size over one whole run.

Algorithm 4 Set reduction with parameters refining.

```
Data: Initial parameters $P$, original set $V_{val}$
Result: Tuned parameters $P$, reduced set $V_{suf}$
1  $V_{suf} \leftarrow$ reduce($V_{val}$, $P$)
2  for $i \leftarrow 1$ to $|P|$ do
3      repeat
4          $size = |V_{suf}|$
5          Refine value of $P[i]$
6          $V_{suf} \leftarrow$ reduce($V_{val}$, $P$)
7          $newSize = |V_{suf}|$
8      until $size = newSize$
9  end
10 return $P$, $V_{suf}$
```

The approach presented does not guarantee that the generated set V_{suf} fully substitutes the original set V_{val} - especially when the refinement steps or the initial values of P are poorly chosen. However, the experiments presented in Section 3.4 show that there is no significant decrease in the quality of the solution obtained when using V_{suf} instead of V_{val} in GTSP$_{\text{arc}}$ instances, whereas the runtime is reduced dramatically and much larger areas can be covered.

2.7 DenseOpt optimization

As explained in Section 1, the planning task is originally continuous, as each set can be covered by infinitely many vertices, given that they respect the constraint in (2). Sampling a finite number of vertices is enforced by using GLNS, which is designed for discrete problems. The inevitable consequence is that the solution obtained by GLNS$_{\text{arc}}$ will probably never be optimal in the original continuous domain. This drawback is accepted, as GLNS$_{\text{arc}}$

Table 4 Parameter values used for V_{suf} generation

Parameter	Initial value	Refinement step
$size(m)$	10	$size = size + 1$
$posRes(m)$	1	$posRes = posRes/2$
$angleRes(rad)$	$\pi/6$	$angleRes = angleRes/2$

is a metaheuristic-based planner, which does not give any guarantees about solution optimality as well. However, the quality of the final solution is negatively influenced by the sampling density. Apart from that, once the GLNS$_{\text{arc}}$ reaches a good quality solution, further convergence tends to be slower and slower. In this phase, it may be more beneficial to perform a simple solution refining in the continuous domain, rather than spending more time solving the optimization problem in the discrete problem. This is the exact purpose of the proposed DenseOpt procedure.

DenseOpt is a newly proposed intensification optimization technique which performs local search in the continuous domain, after the solution in the discrete domain is returned by GLNS$_{\text{arc}}$. It is named to match the other two local optimization techniques defined in GLNS - MoveOpt, and ReOpt. Given a tour T obtained by GLNS$_{\text{arc}}$, DenseOpt searches the close neighborhood of each vertex in T and randomly samples new admissible vertices, not present in the discrete GTSP$_{\text{arc}}$ formulation $G = (V, E, w)$. If the newly sampled vertex improves the weight of T, it replaces the originally present vertex from the same set. The process is described in Algorithm 5.

Algorithm 5 DenseOpt.

```
   Data: Tour $T = (v_0, e_0, v_1, e_1, ..., v_{m-1}, e_{m-1})$
   Result: Locally improved tour $T$
1  $indices \leftarrow [0, 1, ..., m-1]$
2  repeat
3      $stop \leftarrow$ True
4      Uniformly randomly shuffle $indices$
5      for $j \leftarrow 0$ to $m-1$ do
6          $index \leftarrow indices[j]$
7          $v \leftarrow v_{index} \in T$
8          for $k \leftarrow 1$ to $N_s$ do
9              Sample valid vertex $v_{new}$ close to $v$
10             if $v_{new}$ improves $w(T)$ then
11                 $stop \leftarrow$ False
12                 Replace $v_{index}$ in $T$ by $v_{new}$
13                 Update edges $e_{index-1}, e_{index}$ in $T$
14             end
15         end
16     end
17 until $stop$ = True
18 return $T$
```

The whole tour T is repeatedly optimized until there is no improvement in its weight. In each iteration, a random order of resampling is created by shuffling the array $indices$ (line 4). Then, each vertex in the tour T is resampled N_s times (lines 8-9) as v_{new}. Sampling of v_{new} is limited to a predefined range of parameters r, α, and ω. The original density of sampling in the GTSP$_{\text{arc}}$ instance solved determines this range. Vertex v_{new} is sampled anywhere between its closest neighbors. If v_{new} improves tour cost $w(T)$, it is added to T and v_{index} is removed (lines 10-12). Moreover, the edges $e_{index-1}$ and e_{index} are newly generated, so that the newly added v_{new} is connected to the rest of the tour T (line 13). As the vertex v_{index} in T is being replaced continuously, its original position is stored in copy v, so that new vertices v_{new} are sampled in the same subregion (line 8). Parameter N_s was experimentally tuned (see Section 3.5) and set to $N_s = 300$.

3 Results

This chapter documents and interprets the experiments carried out. It is focused on thoroughly demonstrating GLNS$_{\text{arc}}$ capabilities, performance, and assessing the contribution of the additional solution components. It does not provide a comparison with other methods, which is due to several reasons. First, the problem formulation is new and was not fully addressed before. Second, the application previously relied on a human operator input, which is not a feasible approach for thorough experimental comparison. Third, the relevant methods for similar problems are not directly applicable in the considered application without nontrivial adaptation.

The proposed algorithm considers three criteria:

1. source localization in a preselected region is to be guaranteed,
2. all preselected regions are to be inspected,
3. the total trajectory length over all regions is to be optimized.

Adaptive methods (such as [2, 23] or [24]) fully respect only the first criterion and can be iteratively applied to satisfy the second, given that the already discovered sources are removed before continuing the search. Proactive methods (e.g., the zig-zag pattern [39]) are designed to satisfy only the first criterion and are suitable for localizing multiple sources without extraction. However, no adaptation of a proactive method that would consider also the second and third criteria is known to us, apart from the proposed one. In the preceding work [23], the UGV trajectory was created manually and meeting all three criteria depended on a human operator experience. For obvious reasons, this approach is not suitable for experimental comparison.

All instances solved in fast and default mode were solved 50 times so that the results could be processed statistically. This number was reduced to 5 in case of the slow mode due to its excessive time requirements. All experiments were carried out on a single core of Lenovo P330 desktop PC with an Intel Core i7-8700, 3.2 GHz CPU, and 32 GiB of RAM. The GLNS$_{\text{arc}}$ was implemented in C++.

3.1 Generating problem instances

It was described in Section 2.6.1, how to sample a set of valid vertices V_{val}, covering a subregion R. In all generated problem instances, the subregion R was set to a square with the dimensions of 3×3 meters. The size of the subregion is limited by the application-dependent parameters I_s, I_B, and K. Actual values used for generating V_{val} are given in Table 3, as well as values of constants used in the `constraint` function. In real-world experiments, the values of I_s and I_B are estimated from the UAV collected data.

The obtained set V_{val} is shown in Fig. 6a, where the subregion R is marked by the green line, and the generated vertices are displayed in blue. Vertices are partially transparent, so the darker shade of some segments indicates that multiple vertices are overlapping. In total, the set contains 3824 vertices.

The set V_{val} was subsequently reduced by performing the set reduction described in Section 2.6.2. An example of a reduced set V_{suf} is shown in Fig. 6b. This particular set is generated for edge type $line$ by reducing the set V_{val} from Fig. 6a. It contains only 120 vertices (compared to 3824 vertices in V_{val}). Initial values and refining steps for the parameters P are given in Table 4. Parameters common for $line$ and $lineWA$ edge types have identical values. Final values of grid parameters P are $size = 16$ meters and $posRes = 0.125$ meters.

If a region of interest discovered by the UAV in the initial phase is larger than 3×3 meters, or if there are multiple regions, they need to be split into smaller subregions of an appropriate size. E.g., a region of size 18×18 meters can be covered by 36 subregions of size 3×3 meters, as shown in Fig. 7, so that the successful radiation source detection is guaranteed in any point of the area. Each subregion corresponds to a set of vertices in the GTSP$_{\text{arc}}$, and vertices from the same set are displayed in the same color.

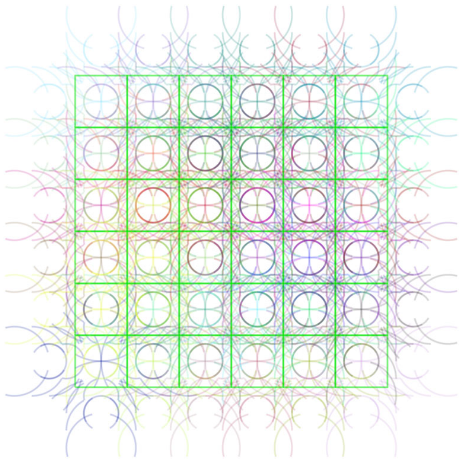


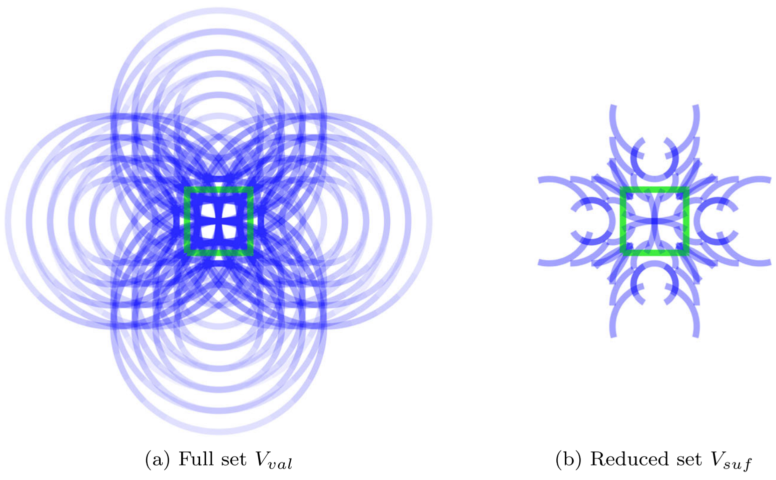
Fig. 7 Problem instance `tiles_suf/6x6_4320`

3.2 Datasets description

Three datasets were created to evaluate the GLNS$_{\text{arc}}$ performance, behaviour and tune the denseOpt parameter: `tiles_full`, `tiles_suf` and `patterns`. The datasets are available at [38].

Dataset `tiles_full` consists of 14 problems of increasing size, which were created by placing the full set of 3824 valid vertices V_{val} described in Section 2.6.2 in a tiled pattern, similarly to the problem displayed in Fig. 7. The largest problem in this dataset covers an area of 12×15 meters, which corresponds to 20 sets (each covering 3×3 meters) and 76480 vertices.

Fig. 6 Generated sets of valid vertices



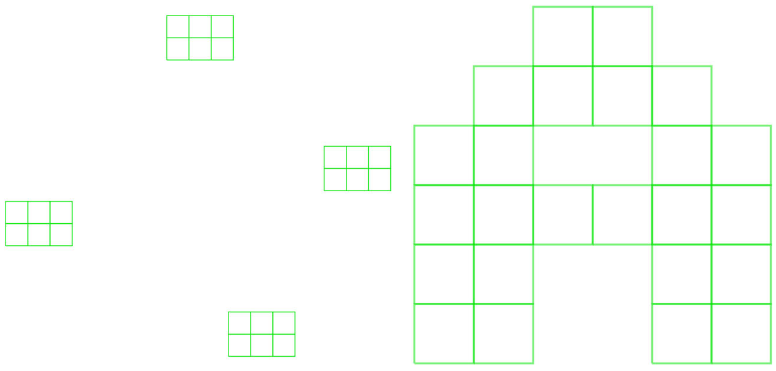


Fig. 8 `patterns` dataset - examples of instances

(a) `clust6_range=40_v3` (b) `bold_a`

Dataset `tiles_suf` was created in the same manner, but it is based on the reduced set V_{suf} containing only 120 vertices, which was generated for edge type *line*. The dataset contains a total number of 53 problems, while the largest problem covers an area of 30 × 30 meters, corresponding to 100 sets and 12000 vertices.

Finally, dataset `patterns` is also based on the V_{suf} set, and it contains problems of medium size ranging between 24 and 32 sets. These problems are created with the intention to capture the behavior of the algorithm on structurally varied problems. It contains fifteen problems, where the sets are clustered by 3, 4, or 6, six problems with

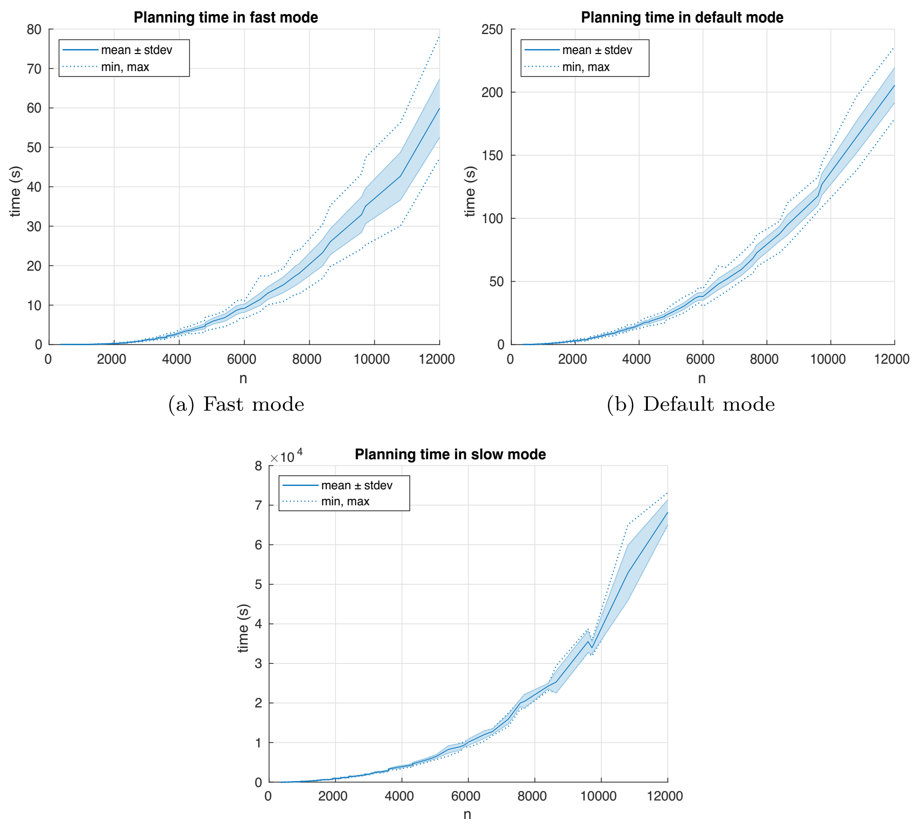


Fig. 9 Planning time across planner modes

(a) Fast mode (b) Default mode (c) Slow mode sets arranged in a letter-shaped pattern, and ten problems with sets distributed randomly. Two examples are shown in Fig. 8.

3.3 GLNS$_{arc}$ modes of operation

GLNS comes with three sets of parameters corresponding to the following planner modes - fast, medium (default), and slow. These modes differ most notably in the number of cold and warm restarts, parameters determining the total number of iterations, and the frequency of applying the local optimization techniques MoveOpt and ReOpt. A full list of all values can be found in [32]. Individual settings were not further modified or tuned, but they were compared in terms of quality of solution and runtime on the `tiles_suf` dataset. The problems in this dataset have a very similar structure, but they gradually increase in size; thus they are suitable for scalability and applicability assessment of individual modes.

Figure 9 shows the planning time needed by individual modes in relation to the problem size, respectively, the number of vertices n. Consistently with the runtime analysis provided in [32], all dependencies plotted are polynomial.

Instances of up to 2700 vertices were always solved within 1 second in the fast mode; thus the GLNS$_{arc}$ can be used as an online planner for smaller problems. Mean planning time for the largest problem of 12000 vertices is 59.9 seconds, while in the worst case, the planning took 78.3 seconds. Fast mode time demands are therefore moderate even for larger instances. The best and worst-case planning times are circa 30% from the mean value, while the standard deviation is up to 10% of the mean. Planning times in the default mode (Fig. 9b) are about one order higher than in the fast mode and show slightly lower deviations (with extrema within 15% from the mean and standard deviation up to 6% of the mean). As for the slow mode (Fig. 9c), planning times are about three orders higher than in the fast mode, thus solving problems with more than 4000 vertices in terms of hours. Individual problems were solved at most 5 times due to the excessive time demands of the slow mode; therefore, the remaining statistical properties are not conclusive and comparable to the other modes.

As for the final tour weight obtained - GLNS$_{arc}$ performance in the fast mode on the `tiles_suf` is visualized in Fig. 10. The resulting weights show very low diversity - the worst tour weight is always within 2.5% from the best weight found and within 1.2% from the mean.

In the case of the default and slow mode, the diversity is even lower and not visually apparent when plotted in the same manner.

Figure 11 compares the relative difference of the mean best weight found across all three modes. Weights obtained in the slow mode are generally the best; thus they are being taken as a baseline. The difference is then calculated as $100\frac{w-w_{slow}}{w_{slow}}$, where w_{slow} is the mean final weight for the particular problem in the slow mode and w is the same value for the currently compared mode. Slow mode results are not displayed in the plot, as they would be compared to themselves, and the corresponding markers would naturally all lie on the line marking zero relative difference. The graph shows that the default mode is at most by 1.5% worse than the slow mode and the fast mode at most by 2.5%. The interpolated mean weight difference initially increases with problem size but appears to stabilize or even slightly decrease for the largest problem instances.

In conclusion, both the fast and default GLNS$_{arc}$ mode performance are sufficient for the intended application, as the planning requires several minutes at worst. The planning

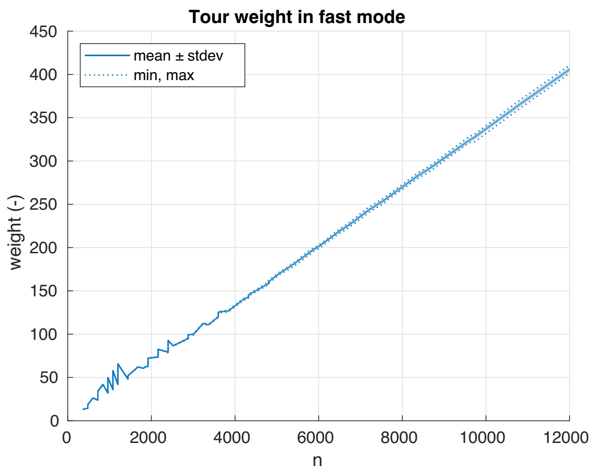


Fig. 10 Final tour weight in fast mode

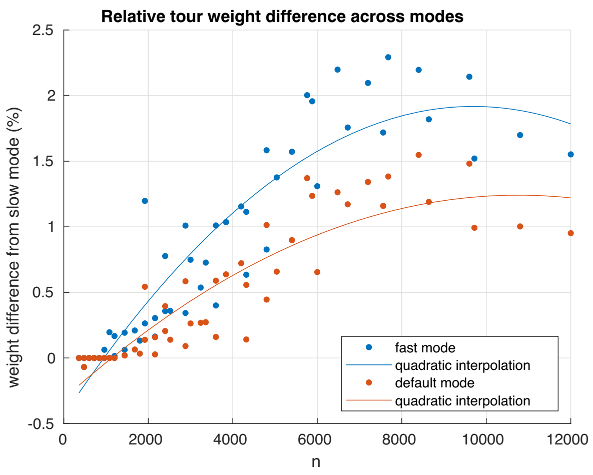


Fig. 11 Final tour weight comparison across planner modes

is to be carried out on a base station between the UAV and the UGV operation, so the plan is not needed instantly and minutes are acceptable for the operating staff. On the other hand, using the slow mode can be considered infeasible, as the planning times exceed hours. In terms of solution quality, the default mode produces solutions worse by up to 1.5% than the slow mode, but this gap can be easily closed by the proposed DenseOpt postprocessing technique, as documented in Section 3.5.

3.4 Planning with full vs. reduced sets

This subsection documents and evaluates the impact of reducing the set size (described in Section 2.6.2) on the planning time and the quality of the solution. For this purpose, the datasets `tiles_suf` and `tiles_full` are used. All problems from the `tiles_full` are present in the `tiles_suf`, meaning that the same area is being covered. However, the sampling density in the `tiles_suf` problems is much lower, as each subregion of 3 × 3 meters is covered by the reduced V_{suf} set (120 vertices), rather than by the full and naively sampled V_{val} set (3824 vertices).

Figure 12 shows the planning time needed for solving all problems in both datasets in the fast mode. The planning time is plotted against the number of sets m, as the number of vertices n differs greatly for the same problems. First, even though the planning times for the largest problems in the `tiles_full` dataset are not unacceptable (mean planning time is at most 167 seconds), the corresponding graph ends at $m = 20$, as the dataset does not contain larger problems. This is due to the fact that the limiting factor of the GLNS$_{\text{arc}}$ are the memory requirements of storing all edge weights, not the planning time (at least in the fast mode). The current implementation could handle instances with circa $8-9 \times 10^4$ vertices on the hardware used. Second, the planning time needed for solving the `tiles_full` problems is significantly higher than for the `tiles_suf`. According to [32], GLNS time complexity in the fast mode in $O(mn)$. The number of vertices n is a linear function of m in both datasets, so the time complexity is $O(m^2)$, and the planning times should differ by a constant factor c, which can be estimated to $c = 300$.

Figure 13 shows the difference in the final tour weight for those problems in both datasets that are identical in terms of the number of sets and their distribution in space. Its value is calculated as $100\frac{w_{suf}-w_{full}}{w_{full}}$, where w_{full} is the mean final weight for a problem from `tiles_full` and w_{suf} is the mean final weight for the corresponding problem from `tiles_suf`. There is no apparent trend, but it can be said that the mean final tour weight of a problem from `tiles_suf` is at most by 0.1% worse than its counterpart from `tiles_full` and that this happens only for three problems. In the case of the remaining instances, the score obtained with V_{suf} is equal to or better than the score obtained with V_{val}. When compared to the relative differences across different planner modes shown in Fig. 11, the effects of using V_{suf} instead of V_{val} have a negligible impact on the quality of solution and the set size reduction described in 2.6.2 can be considered highly beneficial. Given that problems with $n = 75000$ are solvable, the GLNS$_{\text{arc}}$ can be applied to problems with 625 V_{suf} sets, thus cover an area of 75 × 75 meters (5625 m^2). Without the reduction, the GLNS$_{\text{arc}}$ could store at most 20 V_{val} sets covering an area of modest 180 m^2. Naturally, the size of the area depends on the parameters I_s and I_B, which are estimated during the initial UAV reconnaissance.

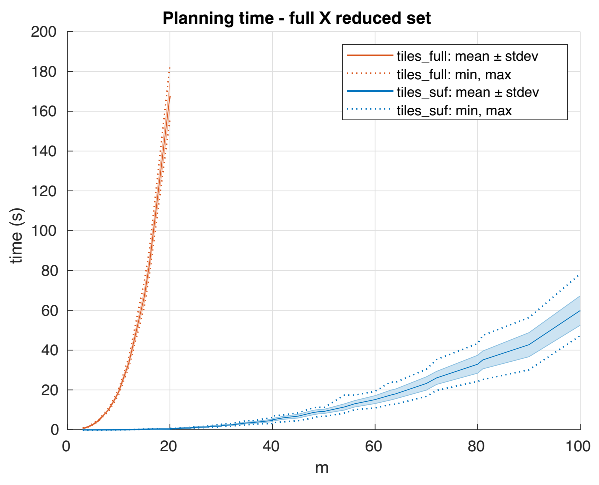


Fig. 12 Planning time comparison - full vs. reduced set

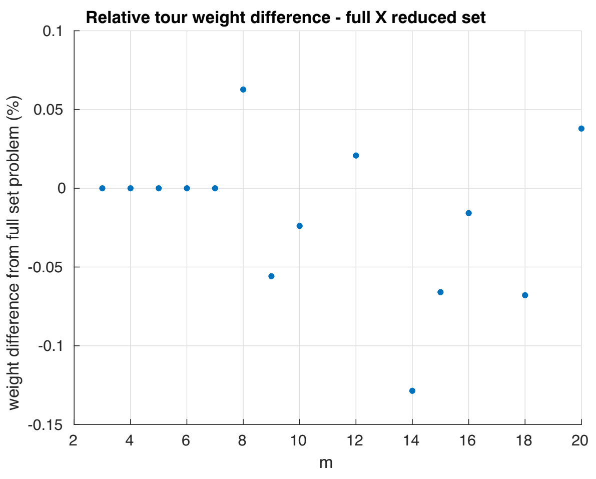


Fig. 13 Final tour weight comparison - full vs. reduced set

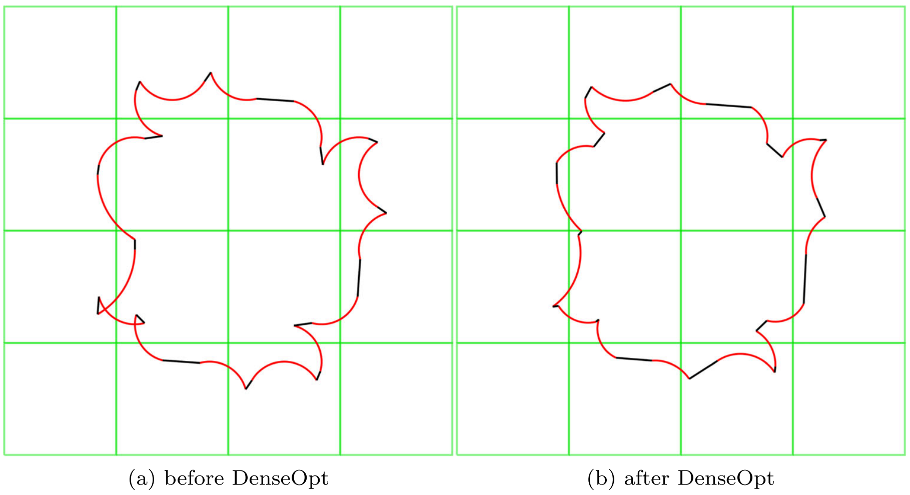


Fig. 14 Solved problem `tiles_suf/4x4_1920`

Their values in Table 3 used for V_{val} generation are taken from [23], where the experiments were carried out in an area of cca 436 m^2.

In conclusion, the proposed set reduction technique is necessary for solving instances of reasonable size. It has only a negligible effect on the solution quality, compared to planning with unreduced data.

3.5 DenseOpt

This subsection documents the performance of the DenseOpt optimization described in Section 2.7. DenseOpt is performed once the GLNS$_{\text{arc}}$ finishes planning on the discretely defined problem and attempts to improve the tour weight by sampling new previously unconsidered vertices in a close neighborhood of the vertices present in the tour.

Figure 14 shows the solved problem `4x4_1920` before and after performing DenseOpt. In this particular case, DenseOpt improves the tour weight by about 18%. Interestingly, it also almost entirely eliminates the mutual crossing of neighboring circular segments, even though the results were obtained with edge type *line*.

DenseOpt is run until there is no improvement in the tour weight and has only one parameter N_s. This parameter determines how many times is every vertex resampled in each DenseOpt iteration. Figure 15 shows the progress of tuning this parameter on the dataset `patterns` in the fast mode, where the relative mean tour weight improvement is plotted against the value of N_s. The improvement is calculated as $\frac{w(T)-w_{dense}(T)}{w(T)}$; here, $w(T)$ is the original mean tour weight and $w_{dense}(T)$ is the mean weight after performing DenseOpt. The plot shows the seemingly logarithmic growth of the relative improvement w.r.t. N_s. However, the logarithmic interpolation plotted along the data reveals that the improvement tends to slow down and lies below the interpolating function for $N_s \geq 250$. This is not surprising, as the relative improvement is bounded to be less than 100%, while the limit of a logarithm on an arbitrary base is infinity. Figure 16 shows the time requirements of the DenseOpt w.r.t. N_s. The dependency turns out to be linear, thus the higher value of N_s (number of resampling attempts per vertex) does not accelerate the convergence of the DenseOpt optimization towards a local optimum. Instead, it presumably enables to reach the local optimum in the continuous domain with higher precision, thus resulting in a slightly better final score.

Based on these observations, the parameter value is set to $N_s = 300$ in the following experiments, which corresponds to a mean improvement of circa 7% and an average duration of 1 second on the `patterns` dataset.

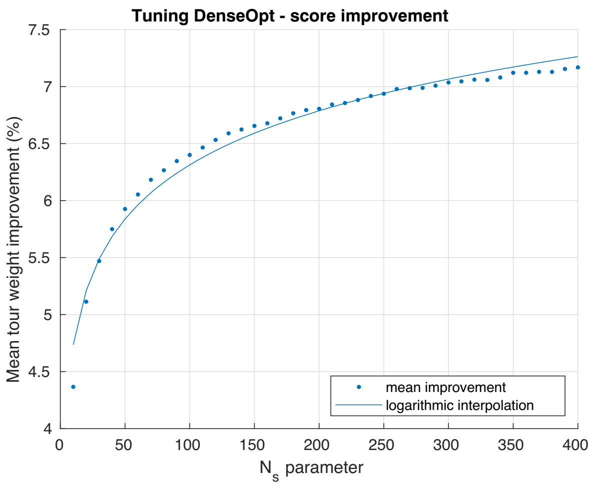


Fig. 15 Tuning DenseOpt - score

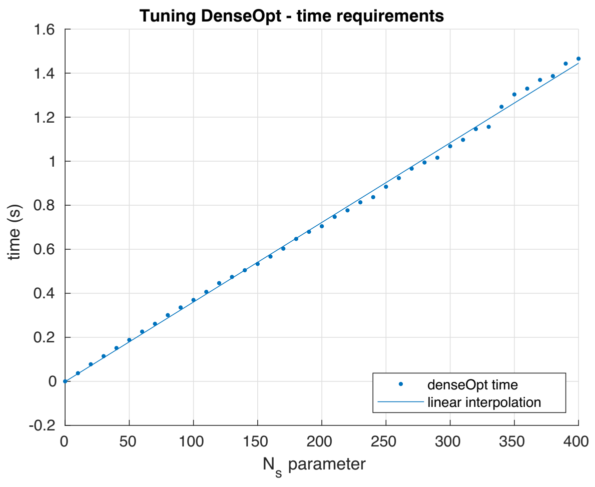


Fig. 16 Tuning DenseOpt - time

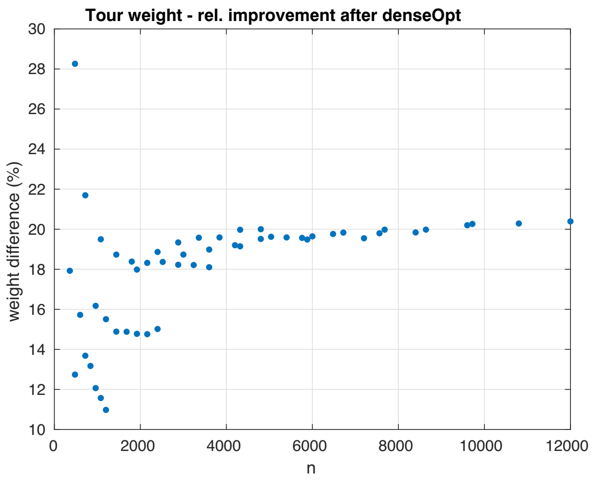


Fig. 18 Tuned DenseOpt performance - relative

DenseOpt performance after parameter tuning is evaluated on the dataset tiles_suf. Figure 17 shows the final tour weight before and after DenseOpt across the whole dataset. It can be observed that the plotted statistical properties (minimum, maximum, and standard deviation) are not affected by the DenseOpt in terms of distance from the mean. Figure 18 then shows the relative improvement. The improvement is at least 10%, and it approaches 20% with increasing problem size. When averaged across the whole dataset, the mean improvement slightly exceeds 18%. In contrast, the mean improvement on the patterns dataset shown in Fig. 15 in only about 7%. Therefore, the effect of DenseOpt is heavily dependent on the problem structure. Individual sets in the tiles_suf problems are placed close together, whereas the sets in the patterns dataset are often sparsely distributed across a larger area in small clusters and thus smaller improvements can be achieved through local resampling of vertices.

Finally, Fig. 19 shows the time requirements of DenseOpt compared to the GLNS$_{\text{arc}}$ planning time w.r.t. m. DenseOpt times are interpolated with a quadratic function; thus, the time complexity can be estimated as $O(m^2)$, given that all sets are the same size. The same applies to the planning time, which is $O(mn)$, therefore $O(m^2)$ for fixed size sets. For smaller instances, DenseOpt requires more time than planning. At 32 sets, the time requirements of planning and DenseOpt are both circa 2.5 seconds, and for larger instances, planning time becomes dominant.

It was shown in Section 3.3, that solving the tiles_suf instances in slow mode yields at most by 2.5% better score than the fast mode and by 1.5% than the default mode. In both cases, the improvement attainable by DenseOpt

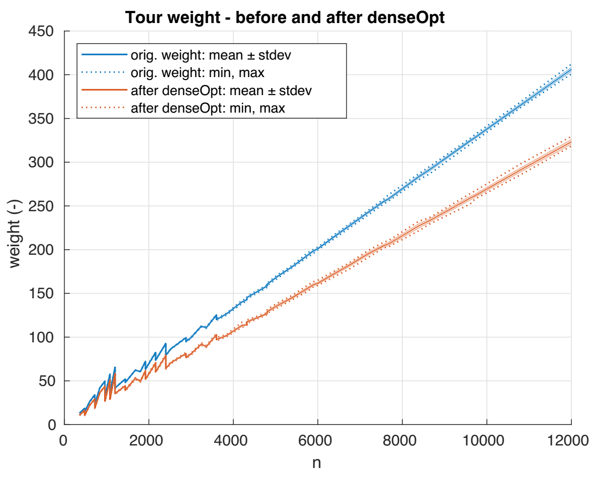


Fig. 17 Tuned DenseOpt performance - absolute

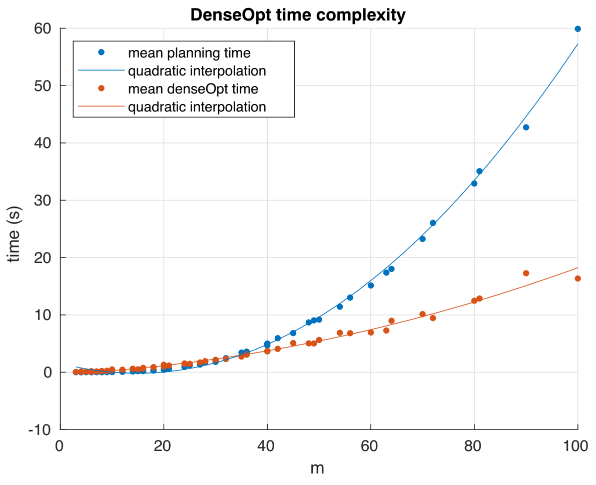


Fig. 19 Tuned DenseOpt time requirements

is several times higher and obtained at a fraction of the additional planning time required by a slower planner mode. Therefore, using the DenseOpt is highly beneficial, as it enables to fundamentally reduce the computation time dedicated for solving the underlying discrete optimization problem and obtain a better quality solution in the continuous domain.

3.6 Set reduction for various edge types

The procedure for sampling valid vertices described in Section 2.6.1 produces a set of 3824 vertices, denoted as V_{val}. This set was then reduced to V_{suf} (Section 2.6.2), a sufficient set of valid vertices generated for edge type $line$. Using V_{suf} instead of V_{val} was then experimentally shown to be highly beneficial in Section 3.4, while planning with the edge type $line$. This subsection presents the results of the set reduction for the edge type $lineWA$ with various angle-weighting constants k.

While generating V_{suf} for edge type $line$, the reduction procedure terminated as proposed, i.e., when there was no further increase in the set size after refining both parameters $size$ and $posRes$ of the reduction grid (Section 2.6.2). In the case of $lineWA$, the set reduction procedure turned out to be excessively time-consuming, as the reduction grid has one more parameter $angleRes$ and is effectively 3-dimensional. Therefore, the reduction for $lineWA$ variants was terminated after $posRes$ was refined to 0.5 meter and $angleRes$ remained at initial value $\frac{\pi}{6}$.

Table 5 V_{suf} size for various edge types

Edge type	Angle weight k	V_{suf} size
$line$	0	120
$lineWA$	0.1	184
$lineWA$	0.5	416
$lineWA$	1	634
$lineWA$	5	2366
$lineWA$	10	3684

The generated sets are shown in Fig. 20 and their sizes given in Table 5. Figure 20 shows the previously used V_{suf} generated for edge type $line$. Figure 20b-f show the reduced set for $lineWA$ with various values of angle weight k. It can be observed that the generated set size increases together with k. For $k = 0.1$, the generated set contains 184 vertices, whereas for $k = 10$, it contains 3684 vertices out of 3824 vertices originally present in V_{val}. An explanation for this trend is that the $lineWA$ edge is weighted according to (4). As the angles δ_i and δ_j are assigned greater weight k, the influence of the distance between neighboring vertex endpoints and vertex length decreases. The edge weights then no longer correspond to Euclidean distances, and every vertex is potentially usable given that its endpoints are suitably oriented. Thus, no significant reduction can be achieved for high values of k.

In conclusion, the proposed set reduction technique enables a significant reduction of the vertex sets, which is

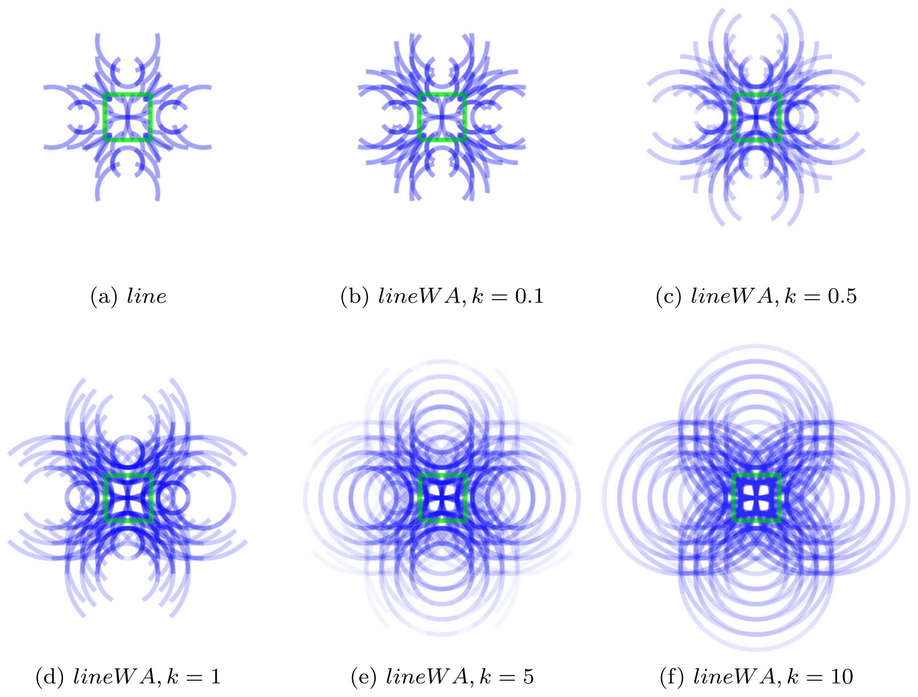


Fig. 20 V_{suf} for various edge types

crucial for solving larger instances. However, the reduction is most effective for metric edge weighting.

3.7 Planning with various edge types

Planning with the edge type $lineWA$ was tested on one problem from the dataset `tiles_suf` and on the whole `patterns` dataset. These datasets are based on the V_{suf} set generated for the edge type $line$. In a real application, problems should be based on V_{suf} generated for the edge type used later in planning. However, creating a separate dataset for each edge type would distort the following comparison of planner performance, as the equivalent problems in different datasets would greatly vary in the number of vertices.

Figure 21 shows the problem `tiles_suf/4x4_1920` solved with four different angle weights k in edge type $lineWA$ after performing DenseOpt. In the case of the smallest value $k = 0.5$, the final tour does not differ much from the solution for edge type $line$, which is shown in Fig. 14. With the increasing value of the constant k, the tour is being straightened at the cost of increasing its Cartesian length, as the algorithm minimizes costly turning maneuvers. For $k = 10$, the black straight line segments (edges) smoothly connect to the red circular segments (vertices), and the trajectory resembles a Dubins path, even though this property is generally not guaranteed.

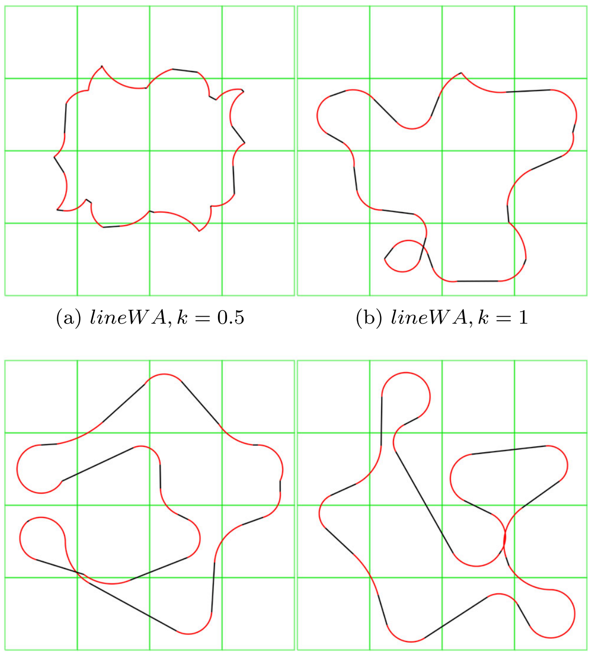


Fig. 21 Solved `4x4_1920` with various edge types

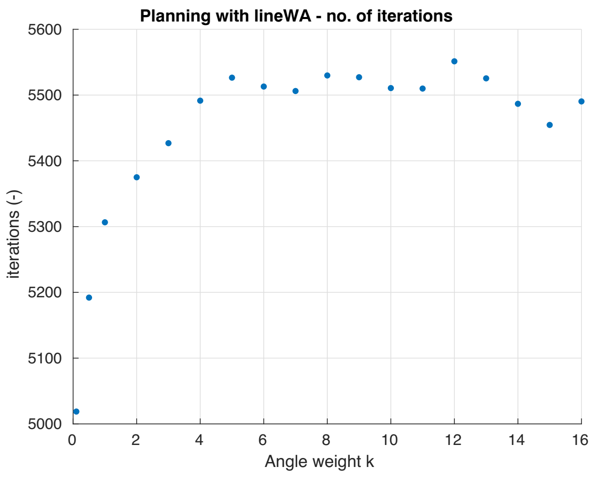


Fig. 22 Planning time for various edge types

Another interesting trend is revealed in Fig. 22, which shows the mean number of iterations needed per problem averaged over the whole `patterns` dataset for different values of k. All problems in the `patterns` dataset have similar size, ranging between 24 and 32 sets of 120 vertices. The mean number of iterations starts at circa 5000 for $k = 0.1$ and increases slightly above 5500, where it settles for $k > 5$. The GLNS$_{\mathrm{arc}}$ terminates after a fixed number of nonimproving iterations in each warm restart, and the results presented indicate that for higher values of k, more iterations are needed to achieve that point. In other words, the local optimum is more difficult to reach.

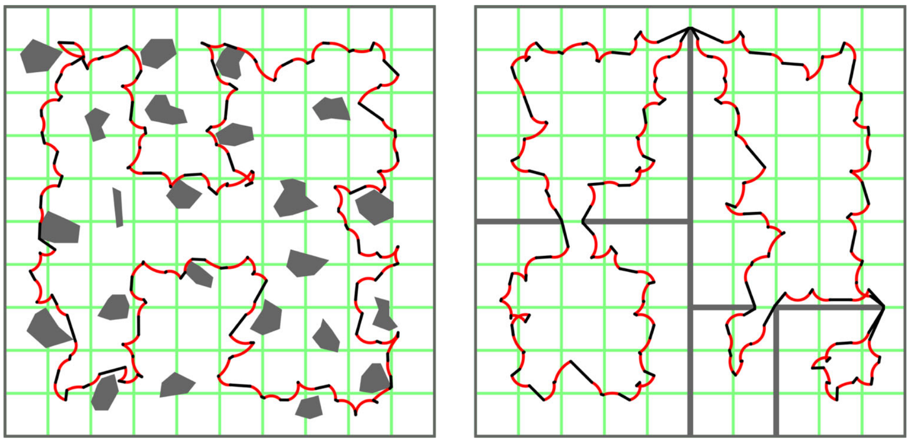
An explanation for this is that the sets in the dataset `patterns` are spatially clustered, and the edge weights are close to Euclidean distances for small values of k. Both of these properties gradually cease to apply, and for $k > 5$, the edge weights are determined primarily by the mutual orientation of vertex endpoints. Thus, the GTSP$_{\mathrm{arc}}$ instances become nonmetric, i.e., they do not satisfy the triangle inequality. Thus, the problems are more difficult to solve, and the planning time increases proportionally to the number of iterations.

In summary, planning with various edge types can be used to produce smooth trajectories, although without guarantee. Using nonmetric edge weighting increases the computational requirements, but not significantly.

3.8 Planning with obstacles

The experimental work [24] motivating the development of GLNS$_{\mathrm{arc}}$ assumed that the terrain is obstacle-free, as its primary focus was on determining the accuracy of the STE. This assumption is generally too strong for practical deployment. However, GLNS$_{\mathrm{arc}}$ can be directly used for

Fig. 23 Planning with obstacles



(a) outdoor map (b) indoor map

planning in an environment with obstacles, given that a map of the environment is available. There are two extra steps needed before the actual planning, both concerning the preparation of input data.

First, vertices colliding with obstacles in the map must be removed from the instance. Second, the edges connecting the vertices must be planned to avoid the obstacles. For both steps, the VisiLibity1 [29] C++ library was used. This library allows for collision checking and shortest path planning in a provided polygonal map. A single edge then corresponds either to a straight line or to a sequence of multiple straight lines avoiding obstacles, whereas its weight is determined as the length of the line or the sequence of lines. In the following examples, the angle weighting constant k is set to zero; thus, sharp turning is not penalized here.

Figure 23a shows a planned path on an outdoor map `potholes` from the dataset [20]. Similarly to the previous figures, the black segments correspond to edges, red segments to vertices (circular arcs, where the measurements are taken), and green squares to subregions covered by a single vertex. The grey polygons then correspond to an impassable terrain. It is assumed here that these polygons do not affect the radiation propagation. Therefore, the sources can be located anywhere on the map, including the polygons.

Figure 23b shows a planned path in an indoor environment. Here, the walls are considered impassable for radiation. Thus, only such vertices that do not collide with a wall and are located in the same room as the covered green subregion are used for planning.

These examples are meant to illustrate that the GLNS$_{\text{arc}}$ is not limited to planning in an obstacle-free outdoor environment. Application in more realistic environments is straightforward and the only additional step is the extraction of invalid vertices from the problem instance.

4 Conclusions

A new planning problem with a background in the search of sources of gamma radiation by a UGV in an outdoor environment was formulated as a GTSP variant and named GTSP$_{\text{arc}}$. GTSP$_{\text{arc}}$ is a combinatorial optimization task in the space of maneuvers guaranteeing source detection in preselected regions. To solve this problem in the discrete domain, a state of the art GTSP solver called GLNS was modified and adapted for the application - the new solver is referred to as GLNS$_{\text{arc}}$.

The paper describes all necessary modifications of the GLNS and evaluates the GLNS$_{\text{arc}}$ performance in multiple experiments on three generic datasets. These datasets are made publicly available at [38]. Method performance is documented to be sufficient for deployment in the motivating application, both in terms of time requirements and scalability. To achieve this, two additional components are proposed. The first one is a preprocessing technique, which significantly reduces the GLNS$_{\text{arc}}$ input data size based on vertex utilization analysis. The technique is shown to have a negligible effect on the solution quality and enables solving instances an order of magnitude larger, thus exploring a larger area. The second one is a postprocessing technique called DenseOpt, which refines the GLNS$_{\text{arc}}$ obtained solution in the continuous domain. The DenseOpt proves to be a more time-efficient way of further improving the solution quality than using a slower GLNS$_{\text{arc}}$ mode or denser vertex sampling. Moreover, two variants of edge weighting are considered and compared - Euclidean (*line*) and Euclidean with weighted angles (*lineWA*). It is also demonstrated that GLNS$_{\text{arc}}$ can be used for planning in the polygonal domain with obstacles. These setups document the applicability of GLNS$_{\text{arc}}$ while considering different vehicle models or environments.

The development of GLNS$_{\text{arc}}$ was motivated by the experimental work described in [24]. The authors of [24] designed a multirobotic system consisting of a UAV and a UGV, deployed it in real-world experiments, and evaluated the accuracy of the detection. The UAV was used for fast identification of regions of interest, and the UGV subsequently performed accurate localization in the preselected regions. However, planning the UGV trajectory in this scenario relied on a human operator, which does neither guarantee the source detection nor does it produce optimized trajectories. The GLNS$_{\text{arc}}$ is designed to automate the planning of the UGV trajectory, while guaranteeing the detection in all preselected regions and producing a near-optimal trajectory. The GLNS$_{\text{arc}}$ is unique in this combination of criteria.

Concerning future work - the GLNS$_{\text{arc}}$ can be directly applied to planning with splines, Dubins curves, or in environments with obstacles that are not polygonal, given that the trajectory segment weights are precomputed. The main issue arising here is the time demand of the weight precomputing. Comparing these variants thoroughly would provide a valuable insight into the applicability of discrete optimization techniques in similar robot routing problems.

Acknowledgements This work has been supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (registration no. CZ.02.1.01/0.0/0.0/15 003/0000470). The work of David Woller has been also supported by the Grant Agency of the Czech Technical University in Prague, grant SGS18/206/OHK3/3T /37.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Arain MA et al (2015) Global coverage measurement planning strategies for mobile robots equipped with a remote gas sensor. In: Sensors (switzerland), vol 15.3, pp 6845–6871. issn: 14248220, https://doi.org/10.3390/s150306845
2. Bourne JR, Pardyjak ER, Leang KK (2019) Coordinated Bayesian-Based bioinspired plume source term estimation and source seeking for mobile robots. IEEE Trans Robot 35.4:967–986. issn: 19410468. https://doi.org/10.1109/TRO.2019.2912520
3. Chen W, Liu L (2019) Pareto monte carlo tree search for multi-objective informative planning. In: Robotics: Science and systems XV. https://doi.org/10.15607/rss.2019.xv.072
4. Christie G et al (2017) Radiation search operations using scene understanding with autonomous UAV and UGV. J Field Robo 34.8:1450–1468. issn: 15564959. https://doi.org/10.1002/rob.21723
5. De Geer LE (2004) Currie detection limits in gamma-ray spectroscopy. Appl Radiat Isotopes 61.2-3:151–160. issn: 09698043. https://doi.org/10.1016/j.apradiso.2004.03.037
6. Drexl M, Gutenberg J (2012) On the generalized directed rural postman problem. Tech. rep Gutenberg School of Management and Economics
7. Ebenezer J, Murty S (2016) Deployment of wireless sensor network for radiation monitoring. In: 2015 International conference on com- puting and network communications (coconet 2015). Institute of Electrical and Electronics Engineers Inc., pp 27–32. isbn: 9781467373098. https://doi.org/10.1109/CoCoNet.2015.7411163
8. Ferri G et al (2007) Explorative particle swarm optimization method for gas/odor source localization in an indoor environment with no strong air-flow. In: 2007 IEEE International conference on robotics and biomimetics, ROBIO. IEEE computer society, pp 841–846. isbn: 9781424417582. https://doi.org/10.1109/ROBIO.2007.4522272
9. Fischetti M, González JJS, Toth P (1997) A branchand-cut algorithm for the symmetric generalized traveling salesman problem. Oper Res 45.3:378–394. issn: 0030364x. https://doi.org/10.1287/opre.45.3.378
10. Fischetti M, González JJS, Toth P (1995) The symmetric generalized traveling salesman polytope. Networks 26.2:113–123. issn: 10970037. https://doi.org/10.1002/net.3230260206
11. Gabrlik P, Lazna T (2018) Simulation of gamma radiation mapping using an unmanned aerial system. In: IFAC-Papersonline 51.6:256–262. issn: 24058963. https://doi.org/10.1016/j.ifacol.2018.07.163
12. Gutin G, Karapetyan D (2010) A memetic algorithm for the generalized traveling salesman problem. Natural Comput 9.1:47–60. issn: 15677818. https://doi.org/10.1007/s11047-009-9111-6. arXiv:0804.0722
13. Han J et al (2013) Low-cost multi-UAV technologies for contour mapping of nuclear radiation field. J Intell Robot Syst: Theory Appl 70.1-4:401–410. issn: 09210296. https://doi.org/10.1007/s10846-012-9722-5
14. Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res 126:106–130
15. Helsgaun K (2013) GTSP problem libraries BAF, MOM and GTSP+. http://akira.ruc.dk/~keld/research/GLKH/. accessed 2020-02-03
16. Helsgaun K (2015) Solving the equality generalized traveling salesman problem using the Lin-Kernighan-Helsgaun Algorithm. Math Programm Comput 7.3:269–287. issn: 18672957. https://doi.org/10.1007/s12532-015-0080-8
17. Hollinger G, Sukhatme G (2016) Sampling-based motion planning for robotic information gathering. In: Robotics: Science and systems. https://doi.org/10.15607/rss.2013.ix.051
18. Hoos HH, Thomas S (2014) On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem. Eur J Oper Res 238.1:87–94. issn: 03772217. https://doi.org/10.1016/j.ejor.2014.03.042
19. Isaacs JT, Hespanha JP (2013) Dubins traveling salesman problem with neighborhoods: A graph-based approach. Algorithms 6.1:84–99. issn: 19994893. https://doi.org/10.3390/a6010084. http://www.mdpi.com/1999-4893/6/1/84
20. Kalisiak M, Faigl J (2013) Motion planning maps - dataset. http://agents.fel.cvut.cz/~faigl/planning/. accessed 2020-07-07
21. Laporte G, Asef-Vaziri A, Sriskandarajah C (1996) Some applications of the generalized travelling salesman problem. J Oper Res Soci 47.12:1461–1467. issn: 14769360. https://doi.org/10.1057/jors.1996.190

22. Laporte G, Nobert Y (1983) Generalized traveling salesman problem through n sets of nodes: an integer programming approach. INFOR: Inf Syst Oper Res 21.1:61–75. issn: 03155986. https://doi.org/10.1080/03155986.1983.11731885
23. Lazna T Optimizing the localization of gamma radiation point sources using a UGV. In: 2018 ELEKTRO Conference Proceedings. Institute of Electrical and Electronics Engineers Inc., pp 1–6. (2018) https://doi.org/10.1109/ELEKTRO.2018.8398368
24. Lazna T, et al. (2018) Cooperation between an unmanned aerial vehicle and an unmanned ground vehicle in highly accurate localization of gamma radiation hotspots. Int J Adv Robot Syst 15.1:172988141775078. issn: 17298814. https://doi.org/10.1177/1729881417750787
25. Lilienthal A, Loutfi A, Duckett T (2006) Airborne chemical sensing with mobile robots. Sensors 6.11:1616–1678. issn: 1424-8220. https://doi.org/10.3390/s6111616
26. Liu Z, Abbaszadeh S, Sullivan CJ (2018) Spatial-temporal modeling of background radiation using mobile sensor networks. PLOS one, vol 13. Ed. by Raghuraman Mudumbai. issn, pp 1932–6203. https://doi.org/10.1371/journal.pone.0205092
27. Miller A., Machrafi R., Mohany A. (2015) Development of a semi-autonomous directional and spectroscopic radiation detection mobile platform. Radiat Measur 72:53–59. issn: 13504487. https://doi.org/10.1016/j.radmeas.2014.11.009
28. Noon CE, Bean JC (1993) An efficient transformation of the generalized traveling salesman problem. INFOR: Inf Syst Oper Res 31.1:39–44. issn: 0315-5986. https://doi.org/10.1080/03155986.1993.11732212
29. Obermeyer KJ, Contributors (2008) VisiLibity: A C++ Library for Visibility Computations in Planar Polygonal Environments. http://www.VisiLibity.org. accessed 2020-07-07
30. Pop PC. (2007) New integer programming formulations of the generalized travelling salesman problem. Amer J Appl Sci 4.11:932–937. issn: 15543641. https://doi.org/10.3844/ajassp.2007.932.937
31. de Julio Rozental J (2002) Two decades of radiological accidents direct causes, roots causes and consequences. Braz Arch Biol Technol 45.spe:125–133. issn: 1516-8913. https://doi.org/10.1590/s1516-89132002000500018
32. Smith SL, Frank I (2017) GLNS An effective large neighborhood search heuristic for the generalized traveling salesman problem. Comput Oper Res 87:1–19
33. Soin PK et al (2019) Application of a novel search method to handheld gamma radiation detectors. IEEE Sens J:1–1. issn: 1530-437X. https://doi.org/10.1109/jsen.2019.2945314
34. Uher J. et al (2007) Directional radiation detector. IEEE Nuclear Sci Symp Conf Rec 2:1162–1166. isbn: 1424409233. https://doi.org/10.1109/NSSMIC.2007.4437213
35. Wendorf M (2020) Broken Arrows - The World's Lost Nuclear Weapons. shorturl.at/ryBLO
36. Wheatley S, Sovacool BK, Sornette Didier (2016) Reassessing the safety of nuclear power. Energy Res Social Sci 15:96–100. issn: 22146296. https://doi.org/10.1016/j.erss.2015.12.026
37. Wiedemann T, Shutin D, Lilienthal AJ (2019) Modelbased gas source localization strategy for a cooperative multi-robot system— A probabilistic approach and experimental validation incorporating physical knowledge and model uncertainties. Robot Auton Syst 118:66–79. issn: 09218890. https://doi.org/10.1016/j.robot.2019.03.014
38. Woller D (2019) GTSP with arcs - 3 datasets. http://imr.ciirc.cvut.cz/Datasets/GTSP-arc. accessed 2020-02-03
39. Zakaria AH et al (2017) Development of autonomous radiation mapping robot. In: Procedia Computer Science. vol 105. Elsevier B.V., pp 81–86. https://doi.org/10.1016/j.procs.2017.01.203

David Woller is a Ph.D. student at the Czech Institute of Informatics, Cybernetics, and Robotics, Czech Technical University (CTU) in Prague. He obtained his Master's degree in 2019 in Cybernetics and Robotics at CTU in Prague. He spent 3 months at a research internship at the University of Avignon, Laboratory of Informatics, France. His main research interests are route planning in robotic applications, metaheuristics, and local search techniques in combinatorial optimization.

Miroslav Kulich is an assistant professor at the Czech Institute of Informatics, Cybernetics, and Robotics, Czech Technical University (CTU) in Prague. He received a Ph.D. degree in Artificial Intelligence and Biocybernetics at CTU in Prague, and an RNDr. degree at Charles University in Prague, Faculty of Mathematics and Physics. He also spent 6 months at a research fellowship at Helsinki University of Technology, Automation Technology Laboratory, Finland. Miroslav Kulich serves as a reviewer for several impacted journals as well as a program committee member of relevant international conferences. His research interests include planning for single and multi-robot systems, especially in exploration and search&rescue scenarios.

Chapter 4

The GRASP Metaheuristic for the Electric Vehicle Routing Problem

In this chapter, we present the second core publication called The GRASP Metaheuristic for the Electric Vehicle Routing Problem [c2], which was published together with a co-authored publication [r8]. The research presented was motivated by the Competition on Electric Vehicle Routing Problem, organized within the 2020 IEEE Congress on Computational Intelligence [27]. Our team (David Woller, Václav Vávra, Viktor Kozák and Miroslav Kulich) won the competition, and both conference publications present the main components and preliminary results. The final method implemented the VNS metaheuristic and was described in the publication [r10], which is still under review in the Q2 journal *Operational Research - An International Journal*, with the implementation available online [87].

[c2] **Woller, D.**, Kozák, V., Kulich, M., "The GRASP Metaheuristic for the Electric Vehicle Routing Problem", English, in *Modelling and Simulation for Autonomous Systems*, ser. 1, Cham, CH: Springer, 2020. DOI: `10.1007/978-3-030-70740-8_12`, **50% contribution, citations: 0 in Web of Science, 1 in Scopus, 4 in Google Scholar.**

The Electric Vehicle Routing Problem (EVRP) is a variant of the classical Capacitated Vehicle Routing Problem (CVRP), which was formulated only recently. In the existing literature, the probably most similar problem is the Green Vehicle Routing Problem (GVRP). The goal of the CVRP is to plan a set of routes for a fleet of vehicles while satisfying the demand of each customer and respecting the limited capacity of individual vehicles. In the GVRP, a limited range of vehicles is considered, together with the option to recharge or refuel at predefined locations. Although the properties of EVRP were considered separately in the literature, the fundamental formulation proposed in the competition [27] has not yet been studied.

In this paper, we propose a metaheuristic algorithm for EVRP based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic [88], which iteratively alternates a construction phase and a local search phase. For the local search phase, a local search heuristic Variable Neighborhood Descent (VND) [89] is deployed. As the main contribution, we designed both the construction procedure and problem-specific components for the local search phase. The crucial component of the construction procedure is a custom repair procedure, which guarantees to transform any sequence of customers into a valid EVRP tour by sequentially fixing battery and load constraints. Multiple alternative construction and repair procedures are proposed and compared in [r8]. The proposed local search operators then inspect neighborhoods commonly used in other Vehicle Routing Problems, but are designed to minimize the number of cost evaluations, since the competition defined the stop condition by a fixed budget of evaluations.

The experimental results in [c2] document the process of tuning the proposed GRASP method and show that it improved the best-known solution BKS on a majority of previously solved instances. A detailed comparison of the final VNS method with four other competing algorithms is available online [27]. Another method, based on the Ant Colony Optimization (ACO) metaheuristic, was proposed after the competition [39]. The experiments carried out in [87] show, that our VNS method outperforms even the newest ACO metaheuristic algorithm in a fair experiment.

The GRASP Metaheuristic for the Electric Vehicle Routing Problem

David Woller(✉), Viktor Kozák, and Miroslav Kulich

Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Jugoslávskách partyzánů 1580/3 160 00 Praha 6, Prague, Czech Republic
{wolledav,viktor.kozak,kulich}@cvut.cz,
http://imr.ciirc.cvut.cz

Abstract. The Electric Vehicle Routing Problem (EVRP) is a recently formulated combination of the Capacitated Vehicle Routing Problem (CVRP) and the Green Vehicle Routing Problem (GVRP). The goal is to satisfy all customers' demands while considering the vehicles' load capacity and limited driving range. All vehicles start from one central depot and can recharge during operation at multiple charging stations. The EVRP reflects the recent introduction of electric vehicles into fleets of delivery companies and represents a general formulation of numerous more specific VRP variants. This paper presents a newly proposed approach based on Greedy Randomized Adaptive Search Procedure (GRASP) scheme addressing the EVRP and documents its performance on a recently created dataset.GRASP is a neighborhood-oriented metaheuristic performing repeated randomized construction of a valid solution, which is subsequently further improved in a local search phase. The implemented metaheuristic improves multiple best-known solutions and sets a benchmark on some previously unsolved instances.

Keywords: Electric vehicle routing problem · Greedy randomized adaptive search · Combinatorial optimization

1 Introduction

This paper addresses the Electric Vehicle Routing Problem (EVRP) recently formulated in [14]. The EVRP is a challenging $\mathcal{NP}$-hard combinatorial optimization problem. It can be viewed as a combination of two variants of the classical Vehicle Routing Problem (VRP) - the Capacitated Vehicle Routing Problem (CVRP) and the Green Vehicle Routing Problem (GVRP). In the VRP, the goal is to minimize the total distance traveled by a fleet of vehicles/agents, while visiting each customer exactly once. In the CVRP, the customers are assigned an integer-valued positive demand, and the vehicles have limited carrying capacity. Thus, an additional constraint of satisfying all customers' demands while respecting the limited vehicle capacity is added to the VRP. Concerning the GVRP, the terminology is not completely settled, but the common idea among different formulations aims at minimizing the environmental impact, typically by taking into

J. Mazal et al. (Eds.): MESAS 2020, LNCS 12619, pp. 189–205, 2021.
https://doi.org/10.1007/978-3-030-70740-8_12

consideration the limited driving range of alternative fuel-powered vehicles and the possibility of refueling at rarely available Alternative Fuel Stations (AFSs). The EVRP has the same objective as the VRP while incorporating the additional constraints from CVRP and GVRP. It is sometimes alternatively named CGVRP, while the name is EVRP often used for other variants of the GVRP (e.g., with considering the non-linear time characteristic of the recharging process or the influence of carried load on the energy consumption).

A method based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic, together with the preliminary results, is presented in the paper. A novel dataset was introduced in [13], and it is the first publicly available dataset of EVRP instances. A subset of this dataset is used in the competition [12]. This paper is the first to give results to the whole dataset and thus represents a valuable benchmark for future researchers. As the competition results were not known at the time of writing, the results are compared only to the organizers-provided best-known values for the smallest instances.

1.1 Related Works

Various approaches were successfully applied to the numerous variants of the VRP, and many of these can also be adapted to the EVRP formulation solved. For example, a recent survey [5] focused only on the variants of EVRP presented a total number of 79 papers. Most of these consider additional constraints intended to reflect real-world conditions, such as using heterogeneous or mixed vehicle fleet, hybrid vehicles, allowing partial recharging or battery swapping, considering different charging technologies and the non-linearity of charging function, dynamic traffic conditions, customer time windows, and others. The problem formulation introduced in [14] stands out, as it leaves out all but the most fundamental constraints on battery and load capacity. Thus, addressing it might produce a universal method adjustable for more specific variants.

According to [5], the most commonly applied metaheuristics are Adaptive Large Neighborhood Search (ALNS), Genetic Algorithms (GA), Large Neighborhood Search (LNS), Tabu Search (TS), Iterative Local Search (ILS) and Variable Neighborhood Search (VNS). The GRASP metaheuristic deployed in this paper is, therefore, not a commonly used one. However, it follows similar principles as other neighborhood-oriented metaheuristics, such as TS, ILS, or VNS. Exact methods such as Dynamic Programming or various Branch-and-Bound/Branch-and-Cut techniques are frequently used as well, but these are generally not suitable for solving larger instances in a reasonable time.

According to [14], the EVRP variant solved was first formulated in [9]. So far, only a few papers are dealing with this problem, and for each one of them, the exact formulation slightly varies. The first one is [18], which additionally limits the maximum number of routes. It presents a solution method based on the Ant Colony System (ACS) algorithm. The second one is [15], which considers the maximum total delivery time. It presents a Simulated Annealing (SA) algorithm operating with four different local search operators (swap, insert, insert AFS, and delete AFS). Then, [17] proposes a novel construction method and a memetic algorithm consisting of a local search and an evolutionary algorithm. Similarly to [15], the local search combines the Variable Neighborhood Search (VNS) with the Simulated Annealing acceptance criterion. The operators used in the local search are 2-opt, swap, insert, and inverse. Unlike the previous approaches, the proposed algorithm is memetic, thus maintains a whole set of solutions. The organizers of [12] themselves also presented a solution method together with the new dataset in [13]. Similarly to [18], they employ an Ant Colony Optimization metaheuristic. However, the problem formulation in [13] differs from [14] and the previously mentioned methods, as it evaluates the energy consumption as a function of the current load. Thus, their results are not directly comparable.

2 Methods

2.1 Problem Formulation

The EVRP can be described as follows: given a fleet of EVs, the goal is to find a route for each EV, such that the following requirements are met. The EVs must start and end at the central depot and serve a set of customers. The objective is to minimize the total distance traveled, while each customer is visited exactly once, for every EV route the total demand of customers does not exceed the EV's maximal carrying capacity and the EV's battery charge level does not fall below zero at any time. All EVs begin and end at the depot, EVs always leave the AFS fully charged (or fully charged an loaded, in case of the depot), and the AFSs (including the depot) can be visited multiple times by any EV. An example of a solved EVRP instance is shown in Fig. 1. Here, the depot, the AFSs, and the customers are represented by a red circle, black squares, and blue circles, respectively. The blue line represents the planned EVRP tour.

The EVRP mathematical formulation as introduced in [14] follows.

$$\min \sum_{i\in V, j\in V, i\neq j} w_{ij}x_{ij}, \tag{1}$$

s.t.

$$\sum_{j\in V, i\neq j} x_{ij} = 1, \forall i \in I, \tag{2}$$

$$\sum_{j\in V, i\neq j} x_{ij} \leq 1, \forall i \in F', \tag{3}$$

$$\sum_{j\in V, i\neq j} x_{ij} - \sum_{j\in V, i\neq j} x_{ji} = 0, \forall i \in V, \tag{4}$$

$$u_j \leq u_i - b_i x_{ij} + C(1 - x_{ij}), \forall i \in V, \forall j \in V, i \neq j, \tag{5}$$

$$0 \leq u_i \leq C, \forall i \in V, \tag{6}$$

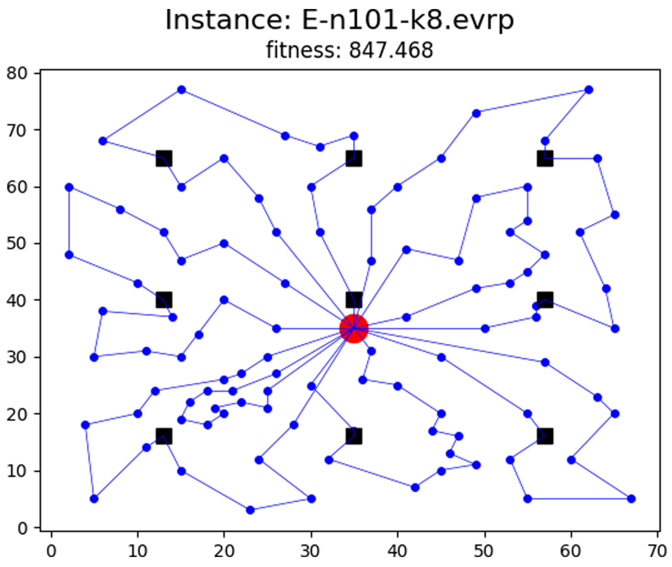


Fig. 1. Solved EVRP instance

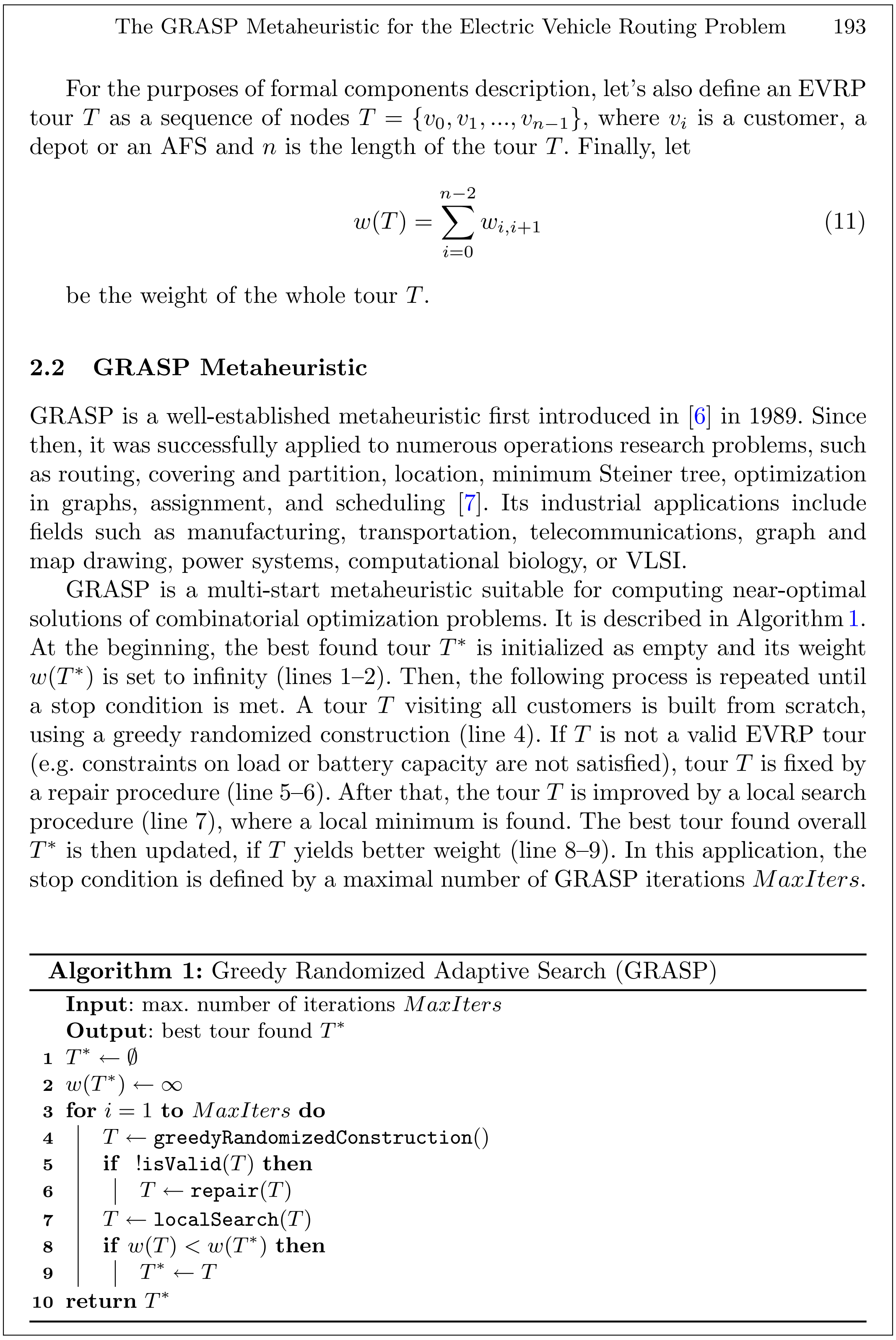
$$y_j \leq y_i - hw_{ij}x_{ij} + Q(1 - x_{ij}), \forall i \in I, \forall j \in V, i \neq j, \tag{7}$$

$$y_j \leq Q - hw_{ij}x_{ij}, \forall i \in F' \cup D, \forall j \in V, i \neq j, \tag{8}$$

$$0 \leq y_i \leq Q, \forall i \in V, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \forall j \in V, i \neq j, \tag{10}$$

where $V = \{D \cup I \cup F'\}$ is a set of nodes. Set I denotes the set of customers, set F' denotes set of β_i node copies of each AFS $i \in F$ (i.e., $|F'| = \sum_{i \in F} \beta_i$) and D denotes the central depot. Lets also define $E = \{(i, j) | i, j \in V, i \neq j\}$ as a set of edges in the fully connected weighted graph $G = (V, E)$. Then, x_{ij} is a binary decision variable corresponding to usage of the edge from node $i \in V$ to node $j \in V$ and w_{ij} is the weight of this edge. Variables u_i and y_i denote, respectively, the remaining carrying capacity and remaining battery charge flevel of an EV on its arrival at node $i \in V$. Finally, the constant h denotes the consumption rate of the EVs, C denotes their maximal carrying capacity, Q the maximal battery charge level, and b_i the demand of each customer $i \in I$.



For the purposes of formal components description, let's also define an EVRP tour T as a sequence of nodes $T = \{v_0, v_1, ..., v_{n-1}\}$, where v_i is a customer, a depot or an AFS and n is the length of the tour T. Finally, let

$$w(T) = \sum_{i=0}^{n-2} w_{i,i+1} \tag{11}$$

be the weight of the whole tour T.

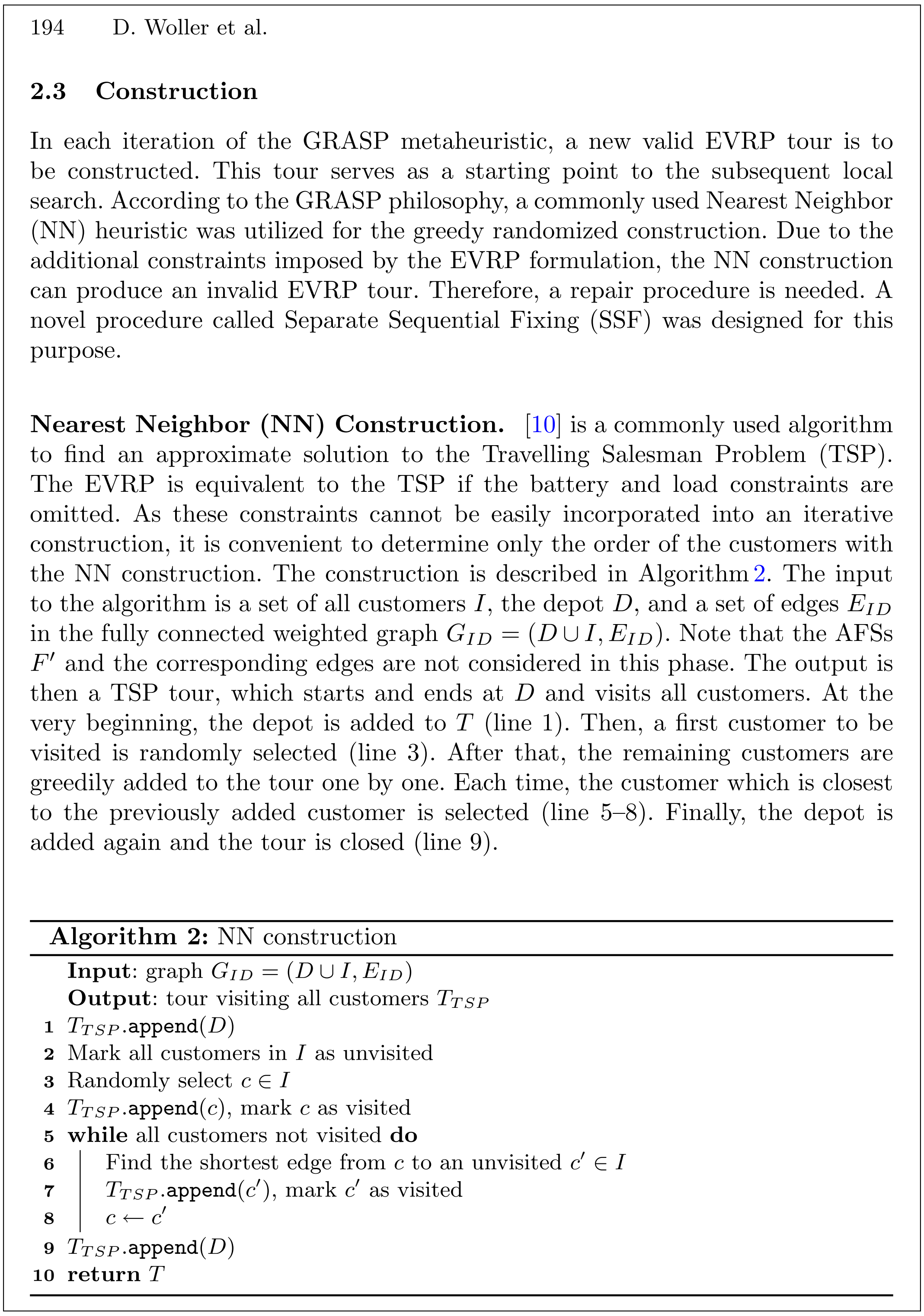
2.2 GRASP Metaheuristic

GRASP is a well-established metaheuristic first introduced in [6] in 1989. Since then, it was successfully applied to numerous operations research problems, such as routing, covering and partition, location, minimum Steiner tree, optimization in graphs, assignment, and scheduling [7]. Its industrial applications include fields such as manufacturing, transportation, telecommunications, graph and map drawing, power systems, computational biology, or VLSI.

GRASP is a multi-start metaheuristic suitable for computing near-optimal solutions of combinatorial optimization problems. It is described in Algorithm 1. At the beginning, the best found tour T^* is initialized as empty and its weight $w(T^*)$ is set to infinity (lines 1–2). Then, the following process is repeated until a stop condition is met. A tour T visiting all customers is built from scratch, using a greedy randomized construction (line 4). If T is not a valid EVRP tour (e.g. constraints on load or battery capacity are not satisfied), tour T is fixed by a repair procedure (line 5–6). After that, the tour T is improved by a local search procedure (line 7), where a local minimum is found. The best tour found overall T^* is then updated, if T yields better weight (line 8–9). In this application, the stop condition is defined by a maximal number of GRASP iterations $MaxIters$.

Algorithm 1: Greedy Randomized Adaptive Search (GRASP)

```
   Input: max. number of iterations $MaxIters$
   Output: best tour found $T^*$
1  $T^* \leftarrow \emptyset$
2  $w(T^*) \leftarrow \infty$
3  for $i = 1$ to $MaxIters$ do
4      $T \leftarrow$ greedyRandomizedConstruction()
5      if !isValid($T$) then
6          $T \leftarrow$ repair($T$)
7      $T \leftarrow$ localSearch($T$)
8      if $w(T) < w(T^*)$ then
9          $T^* \leftarrow T$
10 return $T^*$
```



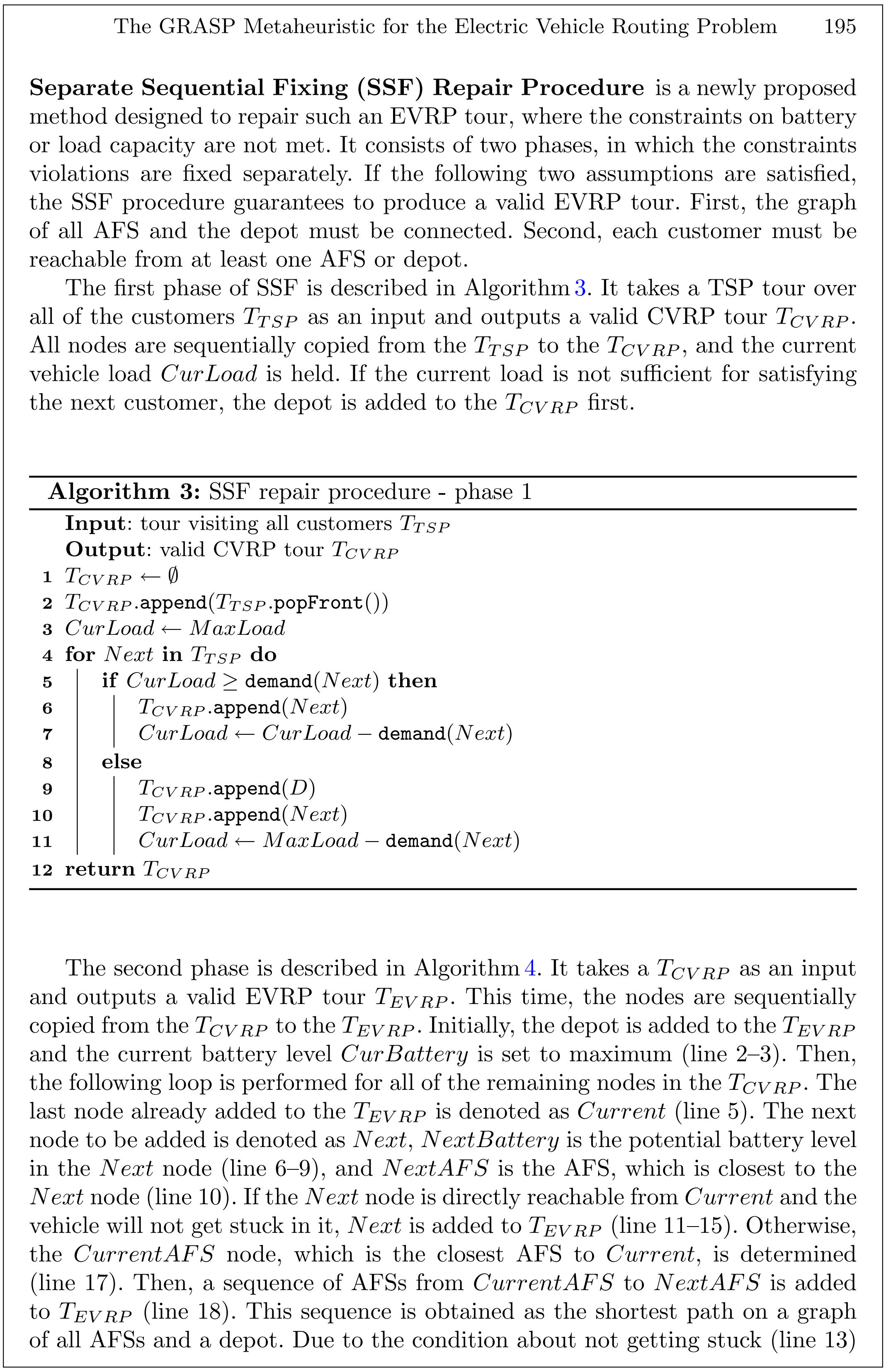
2.3 Construction

In each iteration of the GRASP metaheuristic, a new valid EVRP tour is to be constructed. This tour serves as a starting point to the subsequent local search. According to the GRASP philosophy, a commonly used Nearest Neighbor (NN) heuristic was utilized for the greedy randomized construction. Due to the additional constraints imposed by the EVRP formulation, the NN construction can produce an invalid EVRP tour. Therefore, a repair procedure is needed. A novel procedure called Separate Sequential Fixing (SSF) was designed for this purpose.

Nearest Neighbor (NN) Construction. [10] is a commonly used algorithm to find an approximate solution to the Travelling Salesman Problem (TSP). The EVRP is equivalent to the TSP if the battery and load constraints are omitted. As these constraints cannot be easily incorporated into an iterative construction, it is convenient to determine only the order of the customers with the NN construction. The construction is described in Algorithm 2. The input to the algorithm is a set of all customers I, the depot D, and a set of edges E_{ID} in the fully connected weighted graph $G_{ID} = (D \cup I, E_{ID})$. Note that the AFSs F' and the corresponding edges are not considered in this phase. The output is then a TSP tour, which starts and ends at D and visits all customers. At the very beginning, the depot is added to T (line 1). Then, a first customer to be visited is randomly selected (line 3). After that, the remaining customers are greedily added to the tour one by one. Each time, the customer which is closest to the previously added customer is selected (line 5–8). Finally, the depot is added again and the tour is closed (line 9).

Algorithm 2: NN construction

```
   Input: graph $G_{ID} = (D \cup I, E_{ID})$
   Output: tour visiting all customers $T_{TSP}$
1  $T_{TSP}$.append($D$)
2  Mark all customers in $I$ as unvisited
3  Randomly select $c \in I$
4  $T_{TSP}$.append($c$), mark $c$ as visited
5  while all customers not visited do
6      Find the shortest edge from $c$ to an unvisited $c' \in I$
7      $T_{TSP}$.append($c'$), mark $c'$ as visited
8      $c \leftarrow c'$
9  $T_{TSP}$.append($D$)
10 return $T$
```



Separate Sequential Fixing (SSF) Repair Procedure is a newly proposed method designed to repair such an EVRP tour, where the constraints on battery or load capacity are not met. It consists of two phases, in which the constraints violations are fixed separately. If the following two assumptions are satisfied, the SSF procedure guarantees to produce a valid EVRP tour. First, the graph of all AFS and the depot must be connected. Second, each customer must be reachable from at least one AFS or depot.

The first phase of SSF is described in Algorithm 3. It takes a TSP tour over all of the customers T_{TSP} as an input and outputs a valid CVRP tour T_{CVRP}. All nodes are sequentially copied from the T_{TSP} to the T_{CVRP}, and the current vehicle load $CurLoad$ is held. If the current load is not sufficient for satisfying the next customer, the depot is added to the T_{CVRP} first.

Algorithm 3: SSF repair procedure - phase 1

```
   Input: tour visiting all customers $T_{TSP}$
   Output: valid CVRP tour $T_{CVRP}$
1  $T_{CVRP} \leftarrow \emptyset$
2  $T_{CVRP}$.append($T_{TSP}$.popFront())
3  $CurLoad \leftarrow MaxLoad$
4  for $Next$ in $T_{TSP}$ do
5      if $CurLoad \geq$ demand($Next$) then
6          $T_{CVRP}$.append($Next$)
7          $CurLoad \leftarrow CurLoad -$ demand($Next$)
8      else
9          $T_{CVRP}$.append($D$)
10         $T_{CVRP}$.append($Next$)
11         $CurLoad \leftarrow MaxLoad -$ demand($Next$)
12 return $T_{CVRP}$
```

The second phase is described in Algorithm 4. It takes a T_{CVRP} as an input and outputs a valid EVRP tour T_{EVRP}. This time, the nodes are sequentially copied from the T_{CVRP} to the T_{EVRP}. Initially, the depot is added to the T_{EVRP} and the current battery level $CurBattery$ is set to maximum (line 2–3). Then, the following loop is performed for all of the remaining nodes in the T_{CVRP}. The last node already added to the T_{EVRP} is denoted as $Current$ (line 5). The next node to be added is denoted as $Next$, $NextBattery$ is the potential battery level in the $Next$ node (line 6–9), and $NextAFS$ is the AFS, which is closest to the $Next$ node (line 10). If the $Next$ node is directly reachable from $Current$ and the vehicle will not get stuck in it, $Next$ is added to T_{EVRP} (line 11–15). Otherwise, the $CurrentAFS$ node, which is the closest AFS to $Current$, is determined (line 17). Then, a sequence of AFSs from $CurrentAFS$ to $NextAFS$ is added to T_{EVRP} (line 18). This sequence is obtained as the shortest path on a graph of all AFSs and a depot. Due to the condition about not getting stuck (line 13) and the two SSF assumptions, $CurrentAFS$ is always reachable from $Current$. After $NextAFS$, $Next$ can be added as well, and the loop is repeated until the T_{CVRP} is not empty.

Algorithm 4: SSF repair procedure - phase 2

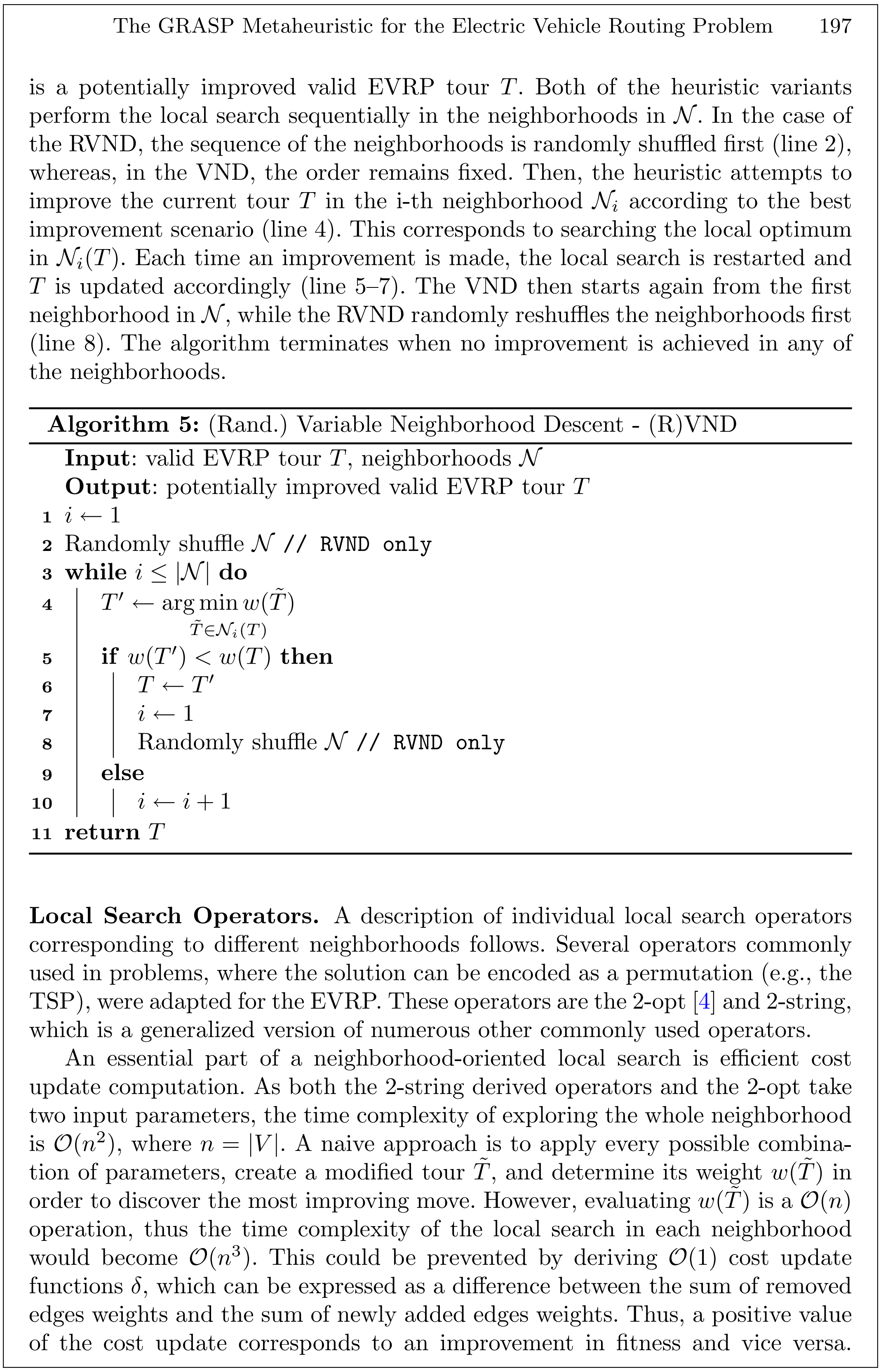
```
   Input: valid CVRP tour $T_{CVRP}$
   Output: valid EVRP tour $T_{EVRP}$
1  $T_{EVRP} \leftarrow \emptyset$
2  $T_{EVRP}$.append($T_{CVRP}$.popFront())
3  $CurBattery \leftarrow MaxBattery$
4  for $Next$ in $T_{CVRP}$ do
5      $Current \leftarrow T_{EVRP}$.back()
6      if isAFS($Next$) then
7          $NextBattery \leftarrow MaxBattery$
8      else
9          $NextBattery \leftarrow CurBattery -$ getConsumption($Current, Next$)
10     $NextAFS \leftarrow$ getClosestAFS($Next$)
11     $Reachable \leftarrow CurBattery \geq$ getConsumption($Current, Next$)
12     $Stuck \leftarrow NextBattery <$ getConsumption($Next, NextAFS$)
13     if $Reachable$ & !$Stuck$ then
14         $T_{EVRP}$.append($Next$)
15         $CurBattery \leftarrow NextBattery$
16     else
17         $CurAFS \leftarrow$ getClosestAFS($Current$)
18         $T_{EVRP}$.append(getPath($CurAFS, NextAFS$))
19         $T_{EVRP}$.append($Next$)
20         $CurBattery \leftarrow MaxBattery -$ getConsumption($T_{EVRP}$.back(), $Next$)
21 return $T_{EVRP}$
```

2.4 Local Search

This section provides a detailed description of the local search that is performed within the GRASP metaheuristic described in Sect. 2.2. The local search uses several local search operators, corresponding to different neighborhoods of an EVRP tour. The application of these operators is controlled by a simple heuristic. For this purpose, the Variable Neighborhood Descent (VND) and its randomized variant (RVND) were selected [3].

(Randomized) Variable Neighborhood Descent - (R)VND is a heuristic commonly used as a local search routine in other metaheuristics. It has a deterministic variant (VND) and a stochastic one (RVND). Both variants are described in Algorithm 5.

The input is a valid EVRP tour T and a sequence of local search operators $\mathcal{N}$, corresponding to different neighborhoods in the search space. The output



is a potentially improved valid EVRP tour T. Both of the heuristic variants perform the local search sequentially in the neighborhoods in $\mathcal{N}$. In the case of the RVND, the sequence of the neighborhoods is randomly shuffled first (line 2), whereas, in the VND, the order remains fixed. Then, the heuristic attempts to improve the current tour T in the i-th neighborhood $\mathcal{N}_i$ according to the best improvement scenario (line 4). This corresponds to searching the local optimum in $\mathcal{N}_i(T)$. Each time an improvement is made, the local search is restarted and T is updated accordingly (line 5–7). The VND then starts again from the first neighborhood in $\mathcal{N}$, while the RVND randomly reshuffles the neighborhoods first (line 8). The algorithm terminates when no improvement is achieved in any of the neighborhoods.

Algorithm 5: (Rand.) Variable Neighborhood Descent - (R)VND

```
   Input: valid EVRP tour $T$, neighborhoods $\mathcal{N}$
   Output: potentially improved valid EVRP tour $T$
1  $i \leftarrow 1$
2  Randomly shuffle $\mathcal{N}$ // RVND only
3  while $i \leq |\mathcal{N}|$ do
4      $T' \leftarrow \arg\min_{\tilde{T} \in \mathcal{N}_i(T)} w(\tilde{T})$
5      if $w(T') < w(T)$ then
6          $T \leftarrow T'$
7          $i \leftarrow 1$
8          Randomly shuffle $\mathcal{N}$ // RVND only
9      else
10         $i \leftarrow i + 1$
11 return $T$
```

Local Search Operators. A description of individual local search operators corresponding to different neighborhoods follows. Several operators commonly used in problems, where the solution can be encoded as a permutation (e.g., the TSP), were adapted for the EVRP. These operators are the 2-opt [4] and 2-string, which is a generalized version of numerous other commonly used operators.

An essential part of a neighborhood-oriented local search is efficient cost update computation. As both the 2-string derived operators and the 2-opt take two input parameters, the time complexity of exploring the whole neighborhood is $\mathcal{O}(n^2)$, where $n = |V|$. A naive approach is to apply every possible combination of parameters, create a modified tour $\tilde{T}$, and determine its weight $w(\tilde{T})$ in order to discover the most improving move. However, evaluating $w(\tilde{T})$ is a $\mathcal{O}(n)$ operation, thus the time complexity of the local search in each neighborhood would become $\mathcal{O}(n^3)$. This could be prevented by deriving $\mathcal{O}(1)$ cost update functions δ, which can be expressed as a difference between the sum of removed edges weights and the sum of newly added edges weights. Thus, a positive value of the cost update corresponds to an improvement in fitness and vice versa.

As the operators can produce an invalid EVRP tour, each local optimum candidate $\tilde{T} \in \mathcal{N}_i(T)$ is determined and checked for validity before acceptance as T'.

2-Opt is an operator commonly used in many variants of classical planning problems such as TSP or VRP. It takes a pair of indices i, j, and a tour T as an input and returns a modified tour $\tilde{T}$, where the sequence of nodes from i-th to j-th index is reversed. It must hold, that $i < j$, $i \geq 0$ and $j < n$.

The cost update function δ_{2-opt} can be evaluated as

$$\delta_{2-opt} = w_{i-1,i} + w_{j,j+1} - w_{i-1,j} - w_{i,j+1}, \tag{12}$$

where the indices are expressed w.r.t. to the tour T.

2-String and Its Variants is a generalized version of several commonly used operators, which can be obtained by fixing some of the 2-string parameters. The 2-string operator takes five parameters: a tour T, a pair of indices i, j valid w.r.t. to T, and a pair of non-negative integers X, Y. It returns a modified tour $\tilde{T}$, where the sequence of X nodes following after the i-th node in T is swapped with the sequence of Y nodes following after the j-th node. It must hold, that $i \geq 0$, $j \geq i + X$ and $j + Y \leq n - 1$. The following operators can be derived by fixing the values of X and Y:

- 1-point: $X = 0, Y = 1$
- 2-point: $X = 1, Y = 1$
- 3-point: $X = 1, Y = 2$
- or-opt2: $X = 0, Y = 2$
- or-opt3: $X = 0, Y = 3$
- or-opt4: $X = 0, Y = 4$
- or-opt5: $X = 0, Y = 5$

When performing the local search, the complementary variants of these operators (e.g., 1-point with $X = 1, Y = 0$) are considered as well.

The cost update function $\delta_{2-string}$ can be evaluated as

$$\delta_{2-string} = cut_1 + cut_2 + cut_3 + cut_4 - add_1 - add_2 - add_3 - add_4, \tag{13}$$

where cut_1 corresponds to the edge weight after i-th node in T, cut_2 to the edge after $i + X$, cut_3 to the edge after j and cut_4 to the edge after $j + Y$. Then, add_1 is the weight of the edge added after the index i-th node in T, add_2 of the edge added after the reinserted block of Y nodes, add_3 of the edge added after j and add_4 of the edge added after the reinserted block of the X nodes. For some combinations of the parameters, some of these values evaluate to zero, which must be carefully treated in the implementation. For example, if $X \neq 0$, then $cut_2 = w_{i+X,i+X+1}$, otherwise $cut_2 = 0$.

3 Results and Discussion

This section documents the parameters tuning and evaluates the performance of the proposed GRASP metaheuristic on the dataset introduced in [13]. Note that the CGVRP formulation used in [13] slightly differs from the EVRP formulation used in this paper and [14]. In [13], the energy consumption depends on the traveled distance and the current vehicle load, whereas in [14], it depends only on the traveled distance. Thus, the best-known values provided in [13] are not relevant when using the more general formulation from [14].

The stop condition of the metaheuristic is also adopted from [14]. An individual run terminates after $25000n$ fitness evaluations of a tour T, where $n = |V|$ is the actual problem size, that is, the total number of unique nodes. A single call to a distance matrix is counted as $1/n$ of an evaluation. Consistently with [14], each problem instance is solved 20 times, with random seeds ranging from 1 to 20. The experiments were carried out on a computer with an Intel Core i7-8700 3.20GHz processor and 32 GB RAM.

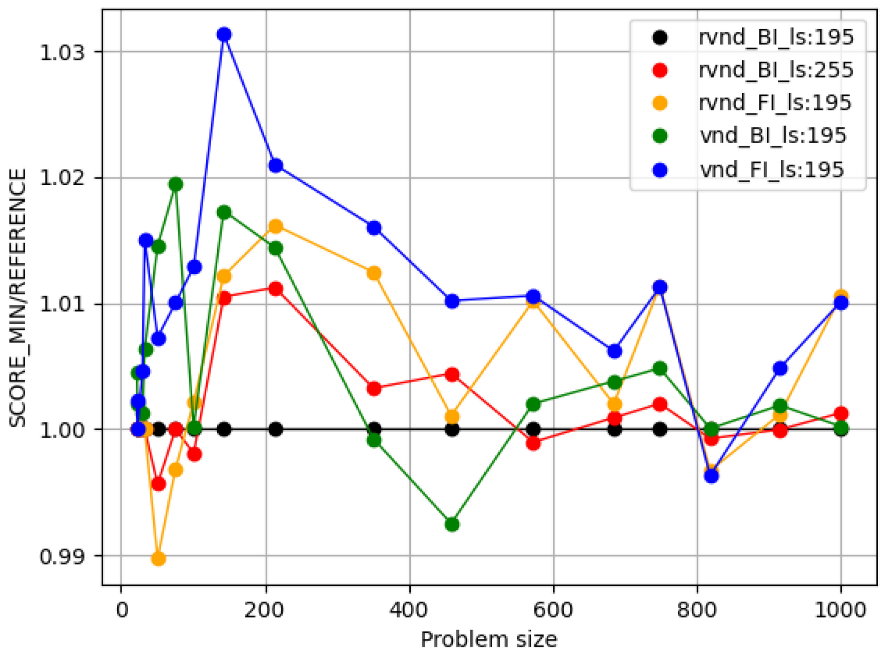
The rest of this section is structured as follows. Section 3.1 provides detailed information about the used dataset [13]. The process of tuning GRASP parameters on a subset of the dataset is described in Sect. 3.2. The final results of the tuned metaheuristic on the whole dataset are given in Sect. 3.3.

3.1 Dataset Description

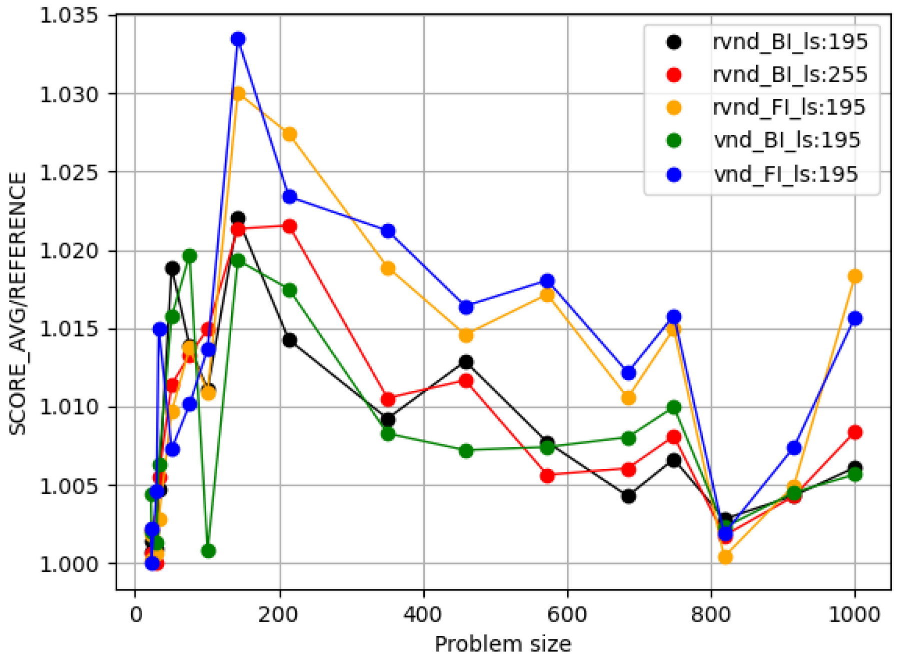
The dataset consists of 4 types of instances, denoted as E, F, M, and X. These EVRP instances were created from already existing CVRP instances by a procedure described in [13], which adds a minimum number of charging stations such that all the customers are reachable from at least one charging station. The E instances are generated from the CVRP benchmark set from [1], the M instances from [2], the F instances from [8] and the X instances from [16]. The E and F instances are small to medium-sized, as they contain between 21 and 134 customers. The M instances are medium-sized (100 to 199 customers), and the X instances are large (143 to 1000 customers). Only instances X and E are addressed in [12]. Some parameters used in [12] instances (e.g., energy consumption rate and maximal energy capacity) slightly differ from the values used in [13]. Here, values from [12] are used when solving the X and E instances.

3.2 Parameters Tuning

The proposed GRASP metaheuristic was implemented in C++ and tuned on the E and X instances from [12]. Three settings of the local search were addressed: randomization of the local search (VND or RVND), neighborhood descent strategy (best improvement - BI or first improvement - FI), and selection of the best performing subset of the operators. The individual setups' names encode the parameter settings. For example, setup rvnd_BI_ls:255 stands for RVND, best improvement, and operators subset no. 255 (which is 11111111 in binary representation and corresponds to using all eight operators). As it was not feasible to



(a) Best performance



(b) Average performance

Fig. 2. Parameters tuning

tune all three parameters simultaneously, the process was split into two phases. First, all 255 possible subsets out the eight operators described in Sect. 2.4 were tested. In this phase, the remaining two parameters were set to RVND and BI. Second, all four combinations of the two remaining parameters were tested simultaneously, while the already selected subset of operators was fixed.

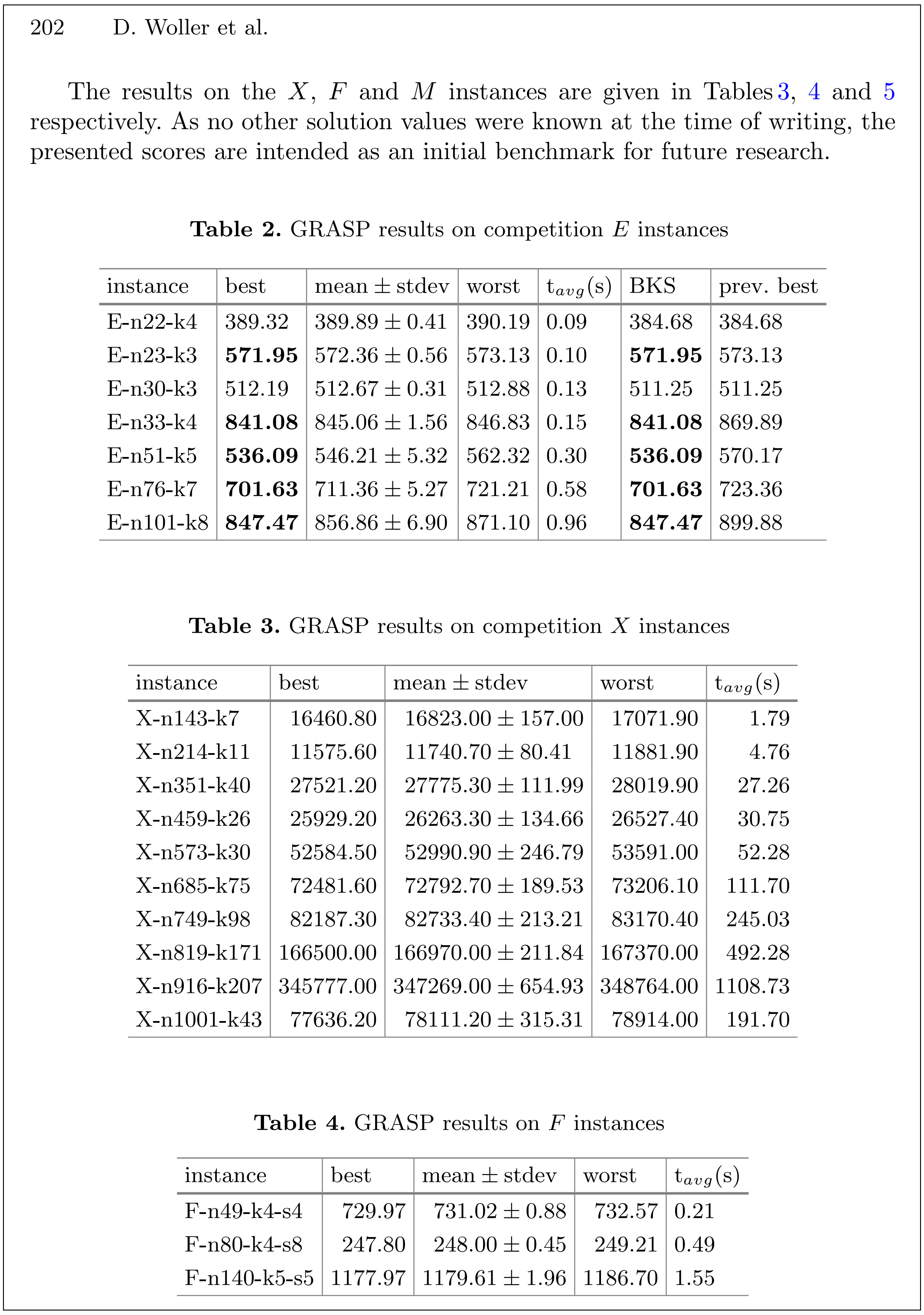
In the first phase, the method rvnd_BI_ls:195 performed best, as it yielded the lowest best score most frequently. The operators used in this setup are 2-opt, 1-point, 2-point, 3-point. Interestingly, no or-opt operator is included. The results of the second phase are presented in Fig. 2 and in Table 1. Fig. 2a displays the best performance of the individual setups, Fig. 2b the average performance and Table 1 provides counts of achieving the lowest (=best) score for each setup. The results are plotted relative to the best score achieved by the setup rvnd_BI_ls:195, which serves as a reference. It turns out that rvnd_BI_ls:195 is also best in terms of achieving the lowest best score (9 times out of 17 instances) in the second phase. However, vnd_BI_ls:195 is slightly better in terms of achieving the lowest average score. The first improvement descent strategy is generally rather unsuccessful. Based on these results, the setup rvnd_BI_ls:195 is selected as the final method. As can be seen in Fig. 2, the differences among individual setups are minor. When averaged across all instances, the worst setup vnd_FI_ls:195 is worse by 1% in terms of the best score and only by 0.5% in terms of the average score than the reference setup.

Table 1. Parameters tuning

GRASP setup	Lowest avg - cnt	lowest best - cnt
rvnd_BI_ls:195	4	9
rvnd_BI_ls:255	3	7
rvnd_FI_ls:195	2	6
vnd_FI_ls:195	3	2
vnd_BI_ls:195	5	2

3.3 Final Results

This section documents the performance of the tuned GRASP metaheuristic on all the available instances from the dataset [13]. The results on the *E* instances are presented in Table 2. The authors of [14] provided fitness values of the best-known solutions for these instances in [12]. These values are shown in the right-most column. The implemented GRASP metaheuristic found better solutions for 5 out of the 7 instances - the improved values are displayed in bold font in Table 2. The current best-known solution scores are given in the column marked as BKS. The best score obtained by the GRASP metaheuristic is at most by 1.2% worse than the BKS (instance E-n22-k4). On the other hand, the GRASP metaheuristic in some cases improved the previous best score by as much as 6% (instances E-n51-k5 and E-n101-k8).



The results on the X, F and M instances are given in Tables 3, 4 and 5 respectively. As no other solution values were known at the time of writing, the presented scores are intended as an initial benchmark for future research.

Table 2. GRASP results on competition E instances

instance	best	mean ± stdev	worst	t_{avg}(s)	BKS	prev. best
E-n22-k4	389.32	389.89 ± 0.41	390.19	0.09	384.68	384.68
E-n23-k3	**571.95**	572.36 ± 0.56	573.13	0.10	**571.95**	573.13
E-n30-k3	512.19	512.67 ± 0.31	512.88	0.13	511.25	511.25
E-n33-k4	**841.08**	845.06 ± 1.56	846.83	0.15	**841.08**	869.89
E-n51-k5	**536.09**	546.21 ± 5.32	562.32	0.30	**536.09**	570.17
E-n76-k7	**701.63**	711.36 ± 5.27	721.21	0.58	**701.63**	723.36
E-n101-k8	**847.47**	856.86 ± 6.90	871.10	0.96	**847.47**	899.88

Table 3. GRASP results on competition X instances

instance	best	mean ± stdev	worst	t_{avg}(s)
X-n143-k7	16460.80	16823.00 ± 157.00	17071.90	1.79
X-n214-k11	11575.60	11740.70 ± 80.41	11881.90	4.76
X-n351-k40	27521.20	27775.30 ± 111.99	28019.90	27.26
X-n459-k26	25929.20	26263.30 ± 134.66	26527.40	30.75
X-n573-k30	52584.50	52990.90 ± 246.79	53591.00	52.28
X-n685-k75	72481.60	72792.70 ± 189.53	73206.10	111.70
X-n749-k98	82187.30	82733.40 ± 213.21	83170.40	245.03
X-n819-k171	166500.00	166970.00 ± 211.84	167370.00	492.28
X-n916-k207	345777.00	347269.00 ± 654.93	348764.00	1108.73
X-n1001-k43	77636.20	78111.20 ± 315.31	78914.00	191.70

Table 4. GRASP results on F instances

instance	best	mean ± stdev	worst	t_{avg}(s)
F-n49-k4-s4	729.97	731.02 ± 0.88	732.57	0.21
F-n80-k4-s8	247.80	248.00 ± 0.45	249.21	0.49
F-n140-k5-s5	1177.97	1179.61 ± 1.96	1186.70	1.55

Table 5. GRASP results on M instances

instance	best	mean ± stdev	worst	t_{avg}(s)
M-n110-k10-s9	829.00	829.29 ± 0.41	830.11	1.04
M-n126-k7-s5	1066.00	1068.05 ± 1.40	1070.10	1.36
M-n163-k12-s12	1068.60	1081.42 ± 8.24	1106.36	2.31
M-n212-k16-s12	1359.55	1377.25 ± 9.82	1395.23	4.45

4 Conclusion

This paper addresses the recently formulated Electric Vehicle Routing Problem and presents the GRASP metaheuristic for finding high-quality solutions in a reasonable time. The performance is tested on the recently proposed dataset [13] of CGVRP instances, which is also the first one publicly available. For the instances with best-known solution values, the GRASP metaheuristic proves to be competitive, as it improves the best-known solution for 5 out of 7 instances. The rest of the instances with no previous solution values is solved as well, and the results are presented for future reference. The main strength of the implemented GRASP metaheuristic lies in efficient local search, which is sped up by using constant-time cost update functions. The key component when applying the GRASP to the EVRP is a newly proposed robust repair procedure called Separate Sequential Fixing (SSF).

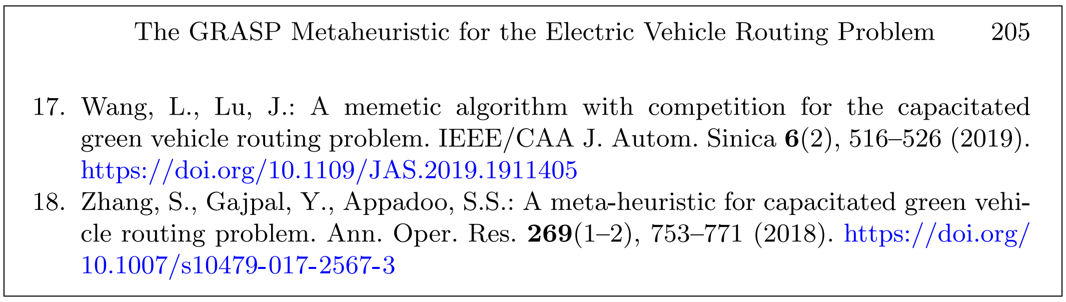
Concerning future work, extracting problem-specific information will be tested. For example, all of the implemented local search operators do not consider the currently unused AFSs, as they are inspired by methods for TSP-like problems and explore only some permutation-based neighborhood of the current solution. Also, the initial NN construction is only distance-based, and meeting the battery and load constraints is left entirely for the repair procedure. Adopting informed methods for initial construction, such as those discussed in [11], might prove beneficial, especially for the larger instances where the quality of the initial solution is crucial. Besides that, it is important to compare the metaheuristic with other methods addressing similar problem formulations, such as [18] or [17]. These were tested on a dataset that is not publicly available and was not obtained at the time of writing.

Acknowledgements. This work has been supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 688117. The work of David Woller and Viktor Kozák has also been supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS18/206/OHK3/3T/37.

References

1. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. J. Oper. Res. Soc. **20**(3), 309–318 (1969). https://doi.org/10.1057/jors.1969.75

2. Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Math. Program. **20**(1), 255–282 (1981). https://doi.org/10.1007/BF01589353
3. Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R.: Variable neighborhood descent. In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Handbook of Heuristics, pp. 341–367. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_9
4. Englert, M., Röglin, H., Vöcking, B.: Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. Algorithmica **68**(1), 190–264 (2013). https://doi.org/10.1007/s00453-013-9801-4
5. Erdelic, T., Carić, T., Lalla-Ruiz, E.: A survey on the electric vehicle routing problem: variants and solution approaches. J. Adv. Transp. **2019**, 48 (2019). https://doi.org/10.1155/2019/5075671
6. Feo, T.A., Resende, M.G.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**(2), 67–71 (1989). https://doi.org/10.1016/0167-6377(89)90002-3
7. Festa, P., Resende, M.G.C.: GRASP. In: Martí, R., Pardalos, P., Resende, M. (eds.) Handbook of Heuristics. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_23
8. Fisher, M.L.: Optimal solution of vehicle routing problems using minimum K-trees. Oper. Res. **42**(4), 626–642 (1994). https://doi.org/10.1287/opre.42.4.626. https://www.jstor.org/stable/171617
9. Goncalves, F., Cardoso, S., Relvas, S.: Optimization of distribution network using electric vehicles: A VRP problem. Technical report. University of Lisbon (2011)
10. Gutin, G., Yeo, A., Zverovich, A.: Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP. Discrete Appl. Math. **117**(1–3), 81–86 (2002). https://doi.org/10.1016/S0166-218X(01)00195-0
11. Kozák, V., Woller, D., Kulich, M.: Initial solution constructors for capacitated green vehicle routing problem. In: Modelling and Simulation for Autonomous Systems (MESAS) **2020** (2020)
12. Mavrovouniotis, M.: CEC-12 Competition on Electric Vehicle Routing Problem (2020). https://mavrovouniotis.github.io/EVRPcompetition2020/. Accessed 23 Nov 2020
13. Mavrovouniotis, M., Menelaou, C., Timotheou, S., Ellinas, G.: A benchmark test suite for the electric capacitated vehicle routing problem. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2020). https://doi.org/10.1109/CEC48606.2020.9185753
14. Mavrovouniotis, M., Menelaou, C., Timotheou, S., Panayiotou, C., Ellinas, G., Polycarpou, M.: Benchmark Set for the IEEE WCCI-2020 Competition on Evolutionary Computation for the Electric Vehicle Routing Problem. Technical report, KIOS Research and Innovation Center of Excellence, Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus (2020). https://mavrovouniotis.github.io/EVRPcompetition2020/TR-EVRP-Competition.pdf
15. Normasari, N.M.E., Yu, V.F., Bachtiyar, C.: Sukoyo: A simulated annealing heuristic for the capacitated green vehicle routing problem. Mathematical Problems in Engineering **2019** (2019). https://doi.org/10.1155/2019/2358258
16. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. **257**, 845–858 (2016). https://doi.org/10.1016/j.ejor.2016.08.012



17. Wang, L., Lu, J.: A memetic algorithm with competition for the capacitated green vehicle routing problem. IEEE/CAA J. Autom. Sinica **6**(2), 516–526 (2019). https://doi.org/10.1109/JAS.2019.1911405
18. Zhang, S., Gajpal, Y., Appadoo, S.S.: A meta-heuristic for capacitated green vehicle routing problem. Ann. Oper. Res. **269**(1–2), 753–771 (2018). https://doi.org/10.1007/s10479-017-2567-3

Chapter 5

The ALNS metaheuristic for the transmission maintenance scheduling

The ALNS metaheuristic for the transmission maintenance scheduling [c3] is the third core publication of this thesis. This work addresses the competition problem of the ROADEF Challenge 2020 [28], which is a prestigious international competition focused on novel combinatorial optimization problems with industrial applications. Our team (David Woller and Jakub Rada) finished tied for 2nd place in the junior category (31 teams) and 8th in the overall ranking (74 teams). The preliminary results and the semifinal method were also presented in [r9] during the competition, which lasted 18 months.

[c3] **Woller, D.**, Rada, J., Kulich, M., "The ALNS metaheuristic for the transmission maintenance scheduling", *Journal of Heuristics*, vol. 29, no. 2-3, pp. 349–382, 2023. DOI: `10.1007/s10732-023-09514-x`, **70% contribution, IF 2.7 (Q2 in Computer Science, Theory & Methods), citations: 1 in Web of Science, 1 in Scopus, 1 in Google Scholar.**

The 2020 problem was proposed by the French high voltage transmission network operator RTE. The goal was to schedule the interventions needed in the network for regular maintenance. The problem was complex, considering about 10 real-world properties and 6 types of hard constraints, most of which were time-variable. These include mutual exclusivity of interventions, different types of limited resources, or non-deterministic risk factors. The largest instances required scheduling more than 700 interventions over a year-long time horizon, with one-day granularity, nine types of resources, and more than 800 exclusivity constraints.

We proposed a metaheuristic algorithm based on the ALNS metaheuristic [90], which we expanded by a local search phase. The main contribution is the design of a large bank of problem-specific construction and destruction heuristics and several local search operators. The heuristics represent the crucial component of the ALNS. A total number of 11 destroy heuristics and 41 repair heuristics were designed based on various problem properties, including those created by hybridization mechanisms. The final method was configured and partially designed automatically, using the irace package [91].

The experimental results presented in [c3] provide a detailed evaluation of the algorithm's performance, which is based on the Best-Known Solutions (BKSs) from the competition. Even on the most challenging data set, the mean gap of the proposed method is within 2% from the BKS. We also carried out a statistical test that compared all methods submitted to the final phase. Although the competition used a custom ranking metric, it turns out that the final ranking would be very similar according to the paired t-test we used. Furthermore, the contribution of individual components and the adaptive behavior of ALNS are analyzed in depth.

The competitors who advanced to the final phase were invited to present their work in the Journal of Heuristics. Thus, some of the related works were published simultaneously with [c3]. The definition of the problem and the generation of instances are described in [92]. The winning team proposed a hybrid metaheuristic algorithm that combines MILP and ILS [41]. The MILP solver was used for finding a feasible initial solution and a custom ILS metaheuristic algorithm, based on several basic local search neighorhoods and a single perturbation operator, was used for futher improving the solution. The 2^{nd} team designed an algorithm based purely on the ILS metaheuristic [42], and the 3^{rd} one proposed a hybrid algorithm combining GRASP and MILP [49]. This time, the GRASP was used for quick finding of a good-quality initial solution, and the MILP solver then ensured solution feasibility and further improved its quality. Some other competitors published their methods as well. Two of them proposed an approach based on problem decomposition and solving MILP relaxations: 7^{th} [93] and 11^{th} [94], while one designed a hybrid algorithm combining VNS and CMA-ES [95]. In summary, metaheuristic algorithms based around efficiently implemented local search tend to outperform more complex methods with stronger theoretical foundations in the competition, presumably due to their considerably better scalability. Combining them with exact MILP solvers then ensures solution feasibility. The exact solvers may also be very beneficial for improving near-optimal solutions or solving smaller subproblems.

Journal of Heuristics (2023) 29:349–382
https://doi.org/10.1007/s10732-023-09514-x

The ALNS metaheuristic for the transmission maintenance scheduling

David Woller[1,2] · Jakub Rada[1] · Miroslav Kulich[1]

Received: 31 March 2022 / Revised: 10 January 2023 / Accepted: 10 May 2023 /
Published online: 27 May 2023

Abstract
ROADEF Challenge is an established international competition addressing challenging industrial problems of combinatorial optimization. It is organized by the French Operations Research and Decision Support Society (ROADEF) every 2 years since 1999. The most recent ROADEF challenge 2020 was co-organized by the French electricity transmission network operator, the RTE company. The competition problem addressed a novel variant of the transmission maintenance scheduling problem, distinctive in that it has multiple time-dependent properties, constraints, and a risk-based aggregate objective function. Therefore, the problem is more complex than the previous formulations, and the existing methods are not directly applicable. This paper presents a metaheuristic algorithm based on the adaptive large neighborhood search. The algorithm's performance is based on a large bank of newly proposed problem-specific destroy and repair heuristics, an efficient local search engine, and a penalization mechanism for avoiding invalid solutions. The algorithm is compared with the best-known solutions from all competition phases and other methods submitted to the final phase. The result shows that the method yields consistent performance in all available datasets. The proposed algorithm finished 6th in the semifinal phase of the competition and 8–9th in the final phase. Finally, the effect of individual components and the algorithm's behaviour are analyzed in detail.

✉ David Woller
wolledav@cvut.cz

Jakub Rada
radajak5@fel.cvut.cz

Miroslav Kulich
kulich@cvut.cz

1 Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Jugoslávských Partyzánů 1580/3, Praha 6 160 00, Czech Republic

2 Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo Náměstí 13, Praha 2 121 35, Czech Republic

Keywords Adaptive large neighborhood search · Metaheuristics · Combinatorial optimization · Transmission maintenance scheduling · ROADEF challenge 2020

1 Introduction

The power transmission and distribution industry provide a large number of combinatorial optimization problems, some of which can be solved by classical graph theory algorithms (Than Kyi et al. 2019; Borůvka 1926), while others require the design of specialized algorithms. This paper addresses a novel variant of the Transmission Maintenance Scheduling (TMS) problem. The goal of TMS is to schedule the maintenance of a high-voltage electricity network, which requires the disconnecting of individual power lines. These disconnections are called interventions and correspond to the tasks to be scheduled. The addressed variant is more complex than various existing variants of TMS (Froger et al. 2016), as it has multiple time-dependent properties, constraints, and a nonlinear objective function. The objective function to be minimized expresses the risk-induced cost of a schedule, where the financial risk evaluation is based on historical data. The addressed TMS variant is proposed by the operator of the French power network (the largest in Europe), the RTE company.

As the TMS is a challenging problem with direct industrial application, it was announced as a competition problem in the ROADEF Challenge 2020. ROADEF Challenge is an established international competition held every two years since 1999. The competition's goal is to identify a previously unsolved real-world industrial problem and present it to the operations research community. The duration of the 2020 challenge was 16 months, and 74 teams participated. Previous ROADEF challenges addressed various problems, such as the Glass Sheets Cutting Optimization problem (proposed by Saint-Gobain Glass France, 2018), the Liquid Oxygen Inventory Routing problem (Air Liquide, 2016), the Rolling Stock Unit Management at Railway Sites problem (Société Nationale des Chemins de Fer, 2014), the Machine Reassignment problem (Google, 2012), or the Large-scale Production Management problem (Électricité de France, 2010). The successful techniques come from all areas of combinatorial optimization. For example, the methods from the final phase in 2014 included custom graph algorithms, local search, metaheuristics (Tabu Search, Simulated Annealing, Large Neighborhood Search), as well as Constraint Programming and Integer Programming techniques (Double Column Generation, Benders Decomposition) (Artigues et al. 2018).

In this paper, a metaheuristic algorithm based on the Adaptive Large Neighborhood Search (ALNS) metaheuristic (Pisinger and Ropke 2010) is proposed for the TMS. The basic principle of the algorithm is to repeatedly partially destroy and repair the current solution, which allows the exploration of various solution neighbourhoods. In these neighbourhoods, a local search is applied to refine the current solution and reach a local optimum. A heuristic approach was selected, as the competition instances contain up to 1000 interventions and are likely to be computationally intractable for exact methods. The addressed TMS variant is rich in properties and constraints, and thus, methods for other variants could not be easily reapplied. The ALNS metaheuristic in particular was selected because a large number of destroy and repair heuristics can be designed, each mimicking a different partial property of the problem.

The initial version of the proposed algorithm was described in Woller and Kulich (2021), with experimental results on the qualification and semifinal dataset. The qualification version consisted of the standard ALNS metaheuristic adapted for the TMS problem, extended by a local search phase. An augmented objective function was formulated to handle both soft and hard constraint violations, and a bank of problem-specific destroy and repair heuristics was newly proposed, together with a set of local search operators.

This paper describes the algorithm submitted to the final phase of the challenge. Several extensions are newly introduced in this paper, together with an exhaustive experimental evaluation on all four competition datasets. The following contributions are newly proposed in this paper, as they were not implemented in the qualification method.

- The proposed repair heuristics are systematically hybridized. Hybridization adds randomness to otherwise deterministic intervention selection and start time selection and enables to further diversify the search process.
- The evaluation of some repair heuristics is sped by estimates based on precomputing intervention static properties, such as average resource demand, cost, and length.
- An automated algorithm design and tuning setup is presented. In the latter stages of the competition, both the structure of the algorithm and the parameters were determined by Iterated Racing for Automatic Algorithm Configuration (López-Ibáñez et al. 2016).

The presented method yields consistent performance on all 4 competition datasets. The method finished 6th in the semifinal phase of the competition (1st in the Junior category) and 8–9th in the final phase (2nd–3rd in the Junior category).

The rest of the paper is structured as follows. Section 2 details the related works. Section 3 provides a formal description of the Transmission Maintenance Scheduling problem. The method submitted to the final competition phase is described in Sect. 4. The experimental results are presented and discussed in Sects. 5, and 6 gives the conclusions.

2 Related works

The ROADEF Challenge 2020 addresses a variant of the Transmission Maintenance Scheduling (TMS) problem. The goal of TMS is to schedule interventions in a transmission network to perform the necessary maintenance. The existing literature distinguishes between long-term TMS and short-term TMS (Shahidehpour and Marwali 2000). In the case of long-term TMS, the time horizon is typically a year, the granularity is days or even weeks, and the maintenance to be scheduled is preventive. The competition problem falls into this category. As for short-term TMS, the time horizon ranges in days or weeks, the granularity of the schedule can be an hour or less, and maintenance is typically preplanned and reactively rescheduled depending on the current state of the network. An example of a short-term TMS variant can be found in Lv et al. (2015).

A closely related problem to TMS is the Generator Maintenance Scheduling problem (GMS). The goal of the GMS is to schedule the shutdown of power generation units rather than disconnecting power lines, but otherwise, the GMS has objectives and constraints similar to the TMS (Froger et al. 2016). These two problems are often addressed simultaneously (Wang et al. 2016).

A wide range of TMS problems was proposed in the literature, but the formulation used in the ROADEF Challenge 2020 competition problem does not seem to fit any variant previously addressed. In existing TMS variants, the structure of the network is often modelled by a transportation model (Abirami et al. 2014) or a DC power flow model (Da Silva et al. 2000). The competition problem does not employ an explicit network model. The TMS objective function can be reliability-based (Schlünz and Van Vuuren 2013), cost-based (El-Sharkh 2014) or a combination of both objectives (Moro and Ramos 1999). The competition problem uses the last variant, as its objective function is a nonlinear aggregation of risk-induced cost values. There can be numerous constraints related to different attributes of the TMS. Interventions can have restricted time windows and be subject to precedence or concurrency constraints. Maintenance can be limited by the availability of workforce or resources. To ensure incident-free network operation, there may also be constraints on the maximum capacity of the transmission lines or the minimum customer demand (Froger et al. 2016). From these, the competition problem considers time windows, mutual exclusivity of interventions, and limited workforce. TMS problems often need to reflect uncertainty, which can be caused by unpredictable fluctuations in supply and demand or extreme weather conditions. This uncertainty can be modelled by the loss of load probability (Reihani et al. 2012) or the expected energy not served (Lu et al. 2012). The competition problem does not utilize an explicit model. Instead, uncertainty is implicitly incorporated into the input data, which contain risk-based costs for each possible start time of every intervention and all relevant crisis scenarios. Finally, most constraints and problem properties are time-dependent, which is not common in existing formulations.

Due to the intractability of the TMS, there has been a cumulative effort of researchers to develop both specialized exact methods and heuristic methods. TMS variants are often formulated as a Mixed Integer Programming problem and addressed with a commercial solver, although this approach is reported to be feasible only for small instances (Mollahassani-Pour et al. 2014). Its scalability can be improved using the Benders decomposition (Geetha and Swarup 2009), as many TMS variants possess the necessary block structure. Some custom problem-specific exact methods were also proposed, such as a dynamic programming approach (Huang 1997) or the branch-and-cut technique (Pandžić et al. 2012). Various metaheuristic algorithms were successfully applied to approximately solve large instances. According to Froger et al. (2016), the most commonly adapted metaheuristics are population-based, such as Genetic Algorithm (Volkanovski et al. 2008) or Particle Swarm optimization (Suresh and Kumarappan 2013). Metaheuristics based on local search are also applied, for example, Simulated Annealing (Saraiva et al. 2011) or Tabu Search (Burke and Smith 2000). Some authors combine exact and heuristic methods in a hybrid solver, e.g. Genetic Algorithm and Mixed Integer Programming (Feng et al. 2009) or Non-Dominated

Sorting Genetic Algorithm III with a custom Dual Simplex method (Salinas San Martin et al. 2022).

The Adaptive Large Neighborhood Search (ALNS) metaheuristic was first applied to the Pickup and Delivery Problem with Time Windows (Ropke and Pisinger 2006) and it is an extension of the Large Neighborhood Search (LNS) metaheuristic (Shaw 1998). LNS introduced the concept of large neighbourhoods, defined by pairs of destroy and repair heuristics, as a counterpart to small neighbourhoods typically used in local search operators. The ALNS then added a learning mechanism that adjusts the use of individual heuristics according to their previous performance, thus adapting the method to the currently solved instance. The main strength of the LNS is providing a robust diversification mechanism that prevents premature convergence, whereas the ALNS is suitable for managing large banks of available heuristics or addressing highly heterogeneous datasets. The ALNS was previously applied to a related problem focused on Maintenance Scheduling for Offshore Oil and Gas Platforms (Khalid et al. 2021). Regarding scheduling problems in general, the existing literature documents many successful applications, such as Service Technician Routing and Scheduling (Kovacs et al. 2012), Multiple Agile Satellites Scheduling (He et al. 2018), Electric Vehicle Scheduling (Wen et al. 2016) or Distributed Reentrant Permutation Flow Shop Scheduling (Rifai et al. 2016). These problems have multiple properties that can be exploited by problem-specific destroy and repair heuristics, which is true for the addressed TMS problem as well. The hybridization mechanisms used in this paper to further expand the portfolio of heuristics and adding a local search phase to the standard ALNS are inspired by Smith and Imeson (2017), which proposed a state of the art ALNS-based algorithm for the Generalized Traveling Salesman Problem. Finally, the proposed method obtains valid solutions using a static penalization of constraint violations with tunable weights, frequently used in Evolutionary Optimization (Back et al. 1997). More robust alternatives to this approach are dynamic penalization (Joines et al. 1994) or adaptive penalization (Ben Hamida and Schoenauer 2000). To the best of our knowledge, the ALNS is applied to TMS for the first time in this paper, respectively, in the preceding conference paper (Woller and Kulich 2021).

3 Problem definition

This section formally defines the TMS problem addressed in the ROADEF Challenge 2020. It is structured as follows. The necessary notation is defined in Sect. 3.1. The problem constraints are described in Sect. 3.2 and the objective function is provided in Sect. 3.3. More details about the problem can be found in Ruiz et al. (2020).

3.1 Notations

This section defines the notation necessary for the formal definition of the problem and the exact description of the proposed method. The notation is adapted from Ruiz et al. (2020) and slightly expanded. As mentioned in Sect. 2, the TMS variant addressed employs an explicit model of the transmission network. Therefore, the formulation is more similar to a highly constrained scheduling problem.

Planning horizon is finite and discrete. Let $T \in \mathbb{N}$ denote the number of time steps in an instance. The discrete-time horizon is defined as $H = \{1, .., T\}$. The value of T determines the granularity of the instance; for example, $T = 53$ corresponds to weeks and $T = 365$ to days.

Interventions correspond to the disconnections of individual power lines from the network and represent the tasks to be scheduled. Let I be the set of all interventions and $i \in I$ a single intervention. Then $\Delta_{i,t} \in \mathbb{N}$ is the duration of an intervention i, given that it started at time t.

Exclusions describe the mutual exclusivity of some interventions. Let Exc be the set of all exclusions. An exclusion is a triplet $(i_1, i_2, t) \in Exc$, where i_1, i_2 are two interventions that cannot be scheduled simultaneously at time $t \in H$.

Resources, such as workforce or equipment, are needed to carry out an intervention. Let C be the set of all resources. The demand for the resource $c \in C$ at time $t \in H$ by an intervention $i \in I$, which started at time t', is given by $r^{c,t}_{i,t'} \in \mathbb{R}$. Each resource $c \in C$ has a lower usage bound $l^c_t \in \mathbb{R}$ and an upper usage bound $u^t_c \in \mathbb{R}$ for each time $t \in H$.

Scenarios are external factors, such as seasonal weather conditions, that can disrupt power delivery and cause financial losses to the network operator. Each time $t \in H$ is assigned a set of possible scenarios S_t.

Risk expresses the possible financial loss due to the scheduling of an intervention $i \in I$ to begin at time $t' \in H$. The risk value at time $t \in H$ is then expressed as $risk^{s,t}_{i,t'} \in \mathbb{R}$, where $s \in S_t$ is an imminent scenario. The risk values are based on historical data collected by the network operator.

Solution of the TMS problem is a list x of pairs $(i, t') \in I \times H$, where t' is the scheduled start time of an intervention i. Finally, let I_t be the set of interventions ongoing at time $t \in H$ and let I_x be the set of interventions scheduled in a solution x. An example of a valid solution is shown in Fig. 1.

3.2 Constraints

Any solution x must meet the following constraints to represent a valid schedule.

Non-preemptive scheduling: an intervention must proceed without interruption once it has started. If an intervention $i \in I$ starts at time $t \in H$, it must end at time $t + \Delta_{i,t}$.

Everything scheduled on time: all interventions must end within the planning horizon. It must hold that $t + \Delta_{i,t} \leq T + 1, \forall i \in I$.

Resource constraints: the total workload of each resource $c \in C$ at time $t \in H$ must be within its lower and upper bounds. The total workload of resource c at time t is given by

$$r^{c,t} = \sum_{i \in I_t} r^{c,t}_{i,t'}.$$

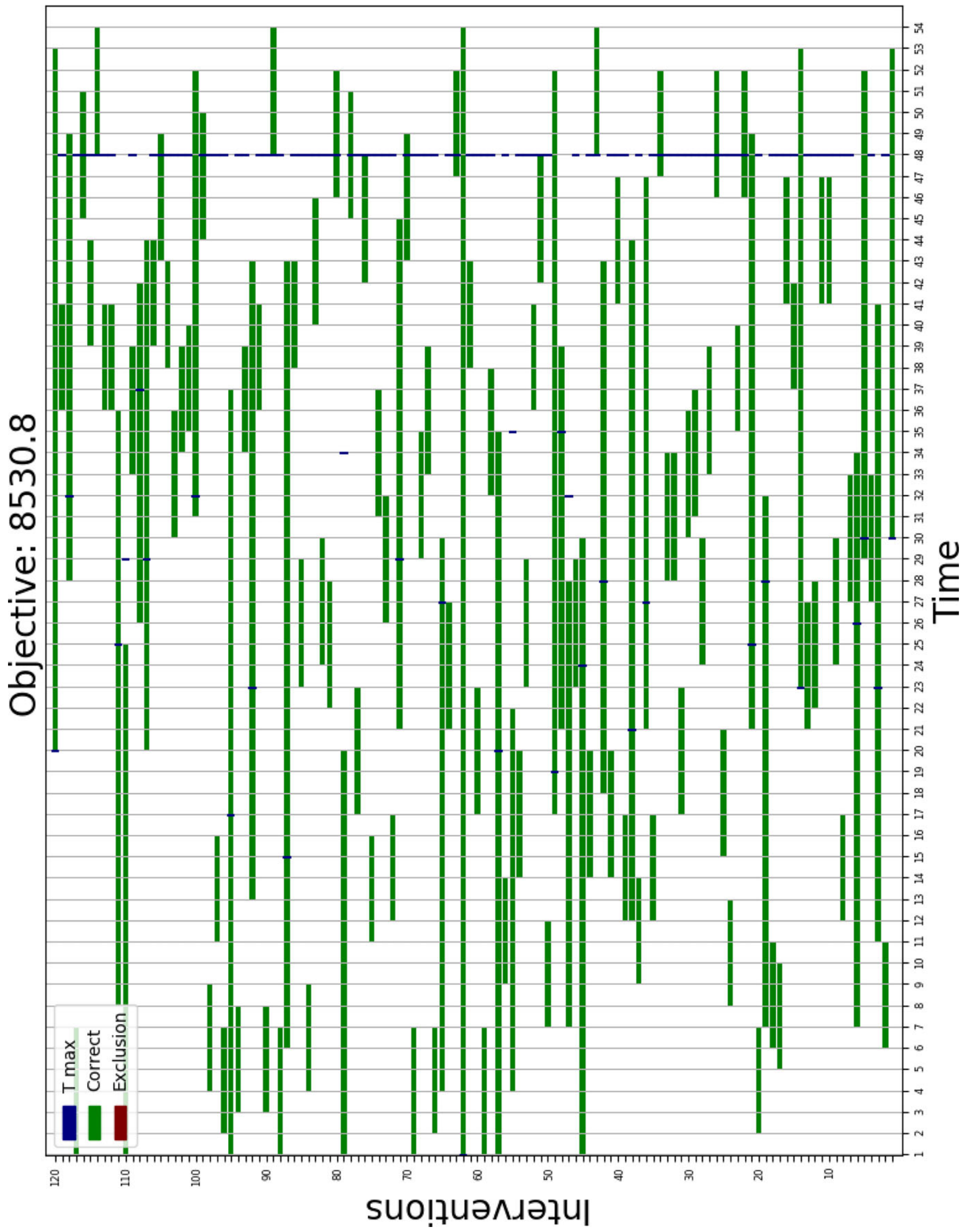


Fig. 1 Solved instance C01

It must hold that $l_t^c \leq r^{c,t} \leq u_t^c, \forall c \in C, t \in H$.

Exclusivity constraints: all the exclusions $(i_1, i_2, t) \in Exc$ has to be respected at all times. It must hold that $i_1 \in I_t \implies i_2 \notin I_t, \forall (i_1, i_2, t) \in Exc$.

3.3 Objective

The objective value obj of a schedule is defined as a weighted aggregation of two values: mean cost obj_1 and expected excess obj_2.

Mean cost evaluates the total planning risk averaged for all scenarios and time steps. The cumulative planning risk at $t \in H$ for a scenario $s \in S_t$ is defined as

$$risk^{s,t} = \sum_{i \in I_t} risk_{i,t'}^{s,t},$$

where t' is the start time of the intervention i. The mean cumulative planning risk at $t \in H$ is then defined as

$$\overline{risk^t} = \frac{1}{|S_t|} \sum_{s \in S_t} risk^{s,t}.$$

Finally, the mean cost is defined as

$$obj_1 = \frac{1}{T} \sum_{t \in H} \overline{risk^t}.$$

Expected excess is intended to suppress extreme cost variability in different scenarios. The expected excess at time $t \in H$ is defined as

$$Excess_\tau(t) = max(0, Q_\tau^t - \overline{risk^t}).$$

Here, Q_τ^t is the τ quantile of cumulative planning risks of all scenarios $s \in S_t$ at time $t \in H$, defined as

$$Q_\tau^t = Q_\tau(\{risk^{s,t}\}_{s \in S_t}),$$

. The value of τ is part of a TMS problem instance. The expected excess of the whole schedule is then defined as

$$obj_2 = \frac{1}{T} \sum_{t \in H} Excess_\tau(t).$$

Final objective obj is given by

$$obj = \alpha \times obj_1 + (1 - \alpha) \times obj_2,$$

where $\alpha \in [0, 1]$ is a provided scaling factor.

4 ALNS metaheuristic for the TMS problem

This section describes the algorithm proposed for the final phase of the challenge. The high-level ALNS metaheuristic is described in Sect. 4.1. Section 4.2 introduces the augmented objective function used to penalize invalid solutions. The key element of the ALNS is a bank of destroy and repair heuristics, designed specifically for the TMS problem. These are described in Sect. 4.3. Hybridization, which systematically adds randomness to the proposed repair heuristics, is introduced in Sect. 4.4. Section 4.5 then presents an approach to reducing the time requirements of some repair heuristics. Finally, the local search is described in Sect. 4.6.

4.1 Overall solution approach

The Adaptive Large Neighborhood Search metaheuristic, first introduced in Ropke and Pisinger (2006), is based on repeated partial destruction and reconstruction of a current solution. For this purpose, a large bank of heuristics for destroying and repairing was proposed specifically for the TMS problem. The usage of individual heuristics is adaptive, meaning that their selection weights in each iteration are adjusted on the basis of their previous performance. In addition, a local search phase was added to the standard ALNS, together with newly proposed local search operators. Local search enables efficient refinement of a current solution and reaching local optima.

Algorithm 1 ALNS metaheuristic

```
1: $i \leftarrow 0,\ x^* \leftarrow \emptyset$
2: while !stop() do
3:   if $i = 0$ then
4:     $x' \leftarrow$ construction()
5:     $\rho^-,\ \rho^+ \leftarrow (1, ..., 1)$
6:   $x \leftarrow x'$
7:   $N_r \leftarrow \mathcal{U}(0, \mathtt{DEPTH} * \mathtt{size}(x))$
8:   select d, r according to $\rho^-$ and $\rho^+$
9:   $x \leftarrow \mathtt{r}(\mathtt{d}(x, N_r), N_r)$
10:  $x \leftarrow$ local_search($x$)
11:  if $c(x) < c(x')$ then
12:    $i \leftarrow 0,\ x' \leftarrow x$
13:  else
14:    $i \leftarrow i + 1$
15:  if $c(x) < c(x^*)$ then
16:    $x^* \leftarrow x$
   update $\rho^-$(d) and $\rho^+$(r)
17:  if $i = i_{max}$ then
18:    $i \leftarrow 0$
19: return $x'$
```

The resulting extended variant of the ALNS is described in Algorithm 1. Here, the variables x, x', x^* represent the current, current best and overall best solution, respectively. The counter i controls the restarting of the main ALNS loop and resets after $i_{max} = \mathtt{ITERS\_MAX}$ not improving iterations. Each restart begins with creating a new solution x' and resetting the selection weights ρ^-, ρ^+ of destroy and repair heuristics, respectively (lines 3–5). Each iteration performs the following process. The rate of destruction N_r of a solution x is randomly sampled uniformly from the interval $(0, \texttt{DEPTH} * \texttt{size}(x))$ (line 7). Then, the destroy heuristic $\texttt{d}$ and the repair heuristic $\texttt{r}$ are selected according to the selection weights ρ^-, ρ^+, using a roulette wheel selection mechanism (line 8). After that, N_r interventions are removed from x using $\texttt{d}$ and reinserted using $\texttt{r}$ and subsequently x is subjected to a local search (lines 9–10). Finally, the acceptance and restart criteria are checked and the selection weights are updated (lines 11–18). The main ALNS loop is controlled by an arbitrary stop condition (line 2), which was given by a fixed computation time according to the rules of the challenge. Alternatively, a maximum number of fitness evaluations or the number of iterations could be used.

The mechanism of updating the selection weights of the individual heuristics is intended to reflect their previous performance and is adapted from Pisinger and Ropke (2010). The weight $\rho^-(\texttt{d})$ of a destroy heuristic $\texttt{d}$ is updated using the following formula:

$$\rho^-(\texttt{d}) = \lambda \rho^-(\texttt{d}) + (1 - \lambda)\omega,$$

where

$$\omega = \begin{cases} \Omega_1, & \text{if new overall best solution } x^* \text{ was found,} \\ \Omega_2, & \text{if new current best solution } x' \text{ was found,} \\ \Omega_3, & \text{otherwise.} \end{cases}$$

It must hold that $\Omega_1 \geq \Omega_2 \geq \Omega_3 \geq 0$. Then $\lambda \in [0, 1]$ is a decay parameter. Regarding the remaining solver parameters, $\texttt{ITERS\_MAX} \in \mathbb{N}$ and $\texttt{DEPTH} \in [0, 1]$. The values of $\texttt{DEPTH}$, $\texttt{ITERS\_MAX}$, λ, Ω_1, Ω_2, and Ω_3 are all fixed solver parameters. Their tuning is described in Sect. 5.3.

4.2 Augmented objective function

The addressed variant of the TMS problem considers four kinds of constraints, described in Sect. 3.2. The scheduling problem is non-preemptive and limited by a finite planning horizon, both of which requirements are easy to satisfy and incorporate into the implementation. However, the resource and exclusivity constraints are variable in time and thus significantly harder to meet. No polynomial-time constructive or repair procedure is known. Therefore, the proposed algorithm also considers invalid solutions during the search process and treats the latter two constraints as soft constraints. The invalidity rate is penalized and combined with the actual objective function of the TMS problem obj in the so-called augmented objective function obj_{aug}. This mechanism is inspired by Michel and Hentenryck (2018).

Individual measures of violating the resource and disjunctive constraints are defined as follows. The excessive usage of the resource c at time t is given by $r^{c,t}_{over} = \max(r^{c,t} - u^c_t, 0)$ and the insufficient usage by $r^{c,t}_{under} = \max(l^c_t - r^{c,t}, 0)$.

Then, the total resource usage of all resources beyond their bounds can be defined as

$$r_{under} = \sum_{c \in C} \sum_{t \in H} r_{under}^{c,t},$$
$$r_{over} = \sum_{c \in C} \sum_{t \in H} r_{over}^{c,t}.$$

As for the exclusivity constraints, the penalty can be defined as

$$e_{pen} = \sum_{(i_1, i_2, t) \in Exc} \texttt{both\_scheduled}(i_1, i_2, t),$$

where $\texttt{both\_scheduled}(i_1, i_2, t) = 1$ if $i_1, i_2 \in I_t$, and 0 otherwise. Finally, the augmented objective function obj_{aug} can be defined as a weighted aggregation of obj and the introduced penalties:

$$obj_{aug} = obj + \beta_1 * r_{under} + \beta_2 * r_{over} + \gamma * e_{pen}.$$

The values of $\beta_1, \beta_2, \gamma \in [10,000, 100,000]$ are tunable parameters. Their ranges are empirically set so that all valid solutions have values of obj_{aug} lower than those of the invalid ones.

4.3 Destroy and repair heuristics

The basic principle of ALNS is to repeatedly ruin and recreate part of a current solution x. In each iteration, N_r interventions are removed by applying a randomly selected destroy heuristic (d) N_r times (Algorithm 1, line 9). Then, these interventions are scheduled again, using a repair heuristic r. The triplet (d, r, N_r) defines the "large neighborhood" in the ALNS metaheuristic. Various destroy heuristics designed specifically for the TMS problem are introduced in Sect. 4.3.1 and repair heuristics in Sect. 4.3.2. Section 4.4 describes two hybridization mechanisms that allow a further generalization of the proposed repair heuristics. Finally, an approach to reduce the computational complexity of repair heuristics is presented in Sect. 4.5.

4.3.1 Destroy heuristics

All destroy heuristics operate as follows. Let x be a current solution and I_x be the set of interventions currently scheduled in x. A destroy heuristic selects a single intervention $i \in I_x$ and removes the pair (i, t) from x. Intervention i can be randomly selected according to some of its individual properties (length, number of exclusions) or according to its influence on some properties of the current schedule (decrease in cost, decrease in resource demand). The resulting partial solution is marked as $x \backslash i$. Regarding complexity, most of the destroy heuristics are $\mathcal{O}(|I|)$. The proposed destroy heuristics are formally defined in Table 1.

Table 1 Destroy heuristics

Heuristic	Intervention selection rule
Random (RND)	$i = \underset{i \in I_x}{\text{rand}()}$
Cheapest (CH)	$i = \arg\min_{i \in I_x}(obj_{aug}(x) - obj_{aug}(x \backslash i))$
Most expensive (ME)	$i = \arg\max_{i \in I_x}(obj_{aug}(x) - obj_{aug}(x \backslash i))$
Lowest resource demand (LRD)	$i = \arg\min_{i \in I_x}(r_{total}(x) - r_{total}(x \backslash i))$
Highest resource demand (HRD)	$i = \arg\max_{i \in I_x}(r_{total}(x) - r_{total}(x \backslash i))$
Shortest (SH)	$i = \arg\min_{i \in I_x} \Delta_{i,t'}$
Longest (LN)	$i = \arg\max_{i \in I_x} \Delta_{i,t'}$
Least exclusions (LEX)	$i = \arg\min_{i \in I_x} \|Exc_i\|$
Most exclusions (MEX)	$i = \arg\max_{i \in I_x} \|Exc_i\|$
Least used (LU)	$i = \arg\min_{i \in I_x} removed_i$
Most used (MU)	$i = \arg\max_{i \in I_x} removed_i$

Here, t' is the start time currently scheduled for $i \in I_x$, $Exc_i \subset Exc$ is a set of exclusions involving i, $removed_i$ is a counter incremented at each removal of i, and r_{total} is the total resource usage of a current solution, defined as

$$r_{total} = \sum_{c \in C} \sum_{t \in H} r^{c,t}.$$

For example, the HRD destroy heuristic removes the intervention i, which is the most resource-intensive in a current solution x.

4.3.2 Repair heuristics

All repair heuristics perform the following operation. A single unscheduled intervention $i \in I \backslash I_x$ is selected and its new start time t' is determined. The pair (i, t') is then added to a partial solution x, which is marked as $x \cup (i, t')$. The complexity of most repair heuristics is $\mathcal{O}(|I| \times |H|)$ due to the need to determine the start time $t' \in H$. As most of the properties of the intervention are time-dependent, the start time t' influences the duration of the intervention, the total cost and the demand for resources of $x \cup (i, t')$. Therefore, four different start time selection mechanisms are proposed in Table 2.

Individual repair heuristics are formally defined in Table 3. Most heuristics employ the cheapest start time $t'_{i,cheap}$, including those based on a time-independent property (RND, LEX, MEX, LU, MU). Only the LRD2, SH2 and LN2 heuristics employ an alternative start time selection mechanism consistent with their intervention selection

Table 2 Start time selection mechanisms

Mechanism	Start time selection rule
Cheapest	$t'_{i,cheap} = \arg\min_{t \in H}(obj_{aug}(x \cup (i,t)) - obj_{aug}(x))$
Lowest resource demand	$t'_{i,lrd} = \arg\min_{t \in H}(r_{total}(x \cup (i,t)) - r_{total}(x))$
Shortest	$t'_{i,short} = \arg\min_{t \in H} \Delta_{i,t}$
Longest	$t'_{i,long} = \arg\max_{t \in H} \Delta_{i,t}$

Table 3 Repair heuristics

Heuristic	Intervention selection rule
Random (RND)	$i = \underset{i \in I \setminus I_x}{\text{rand}()}$
Cheapest (CH)	$i = \arg\min_{i \in I \setminus I_x} (obj_{aug}(x \cup (i, t'_{i,cheap})) - obj_{aug}(x))$
Most expensive (ME)	$i = \arg\max_{i \in I \setminus I_x} (obj_{aug}(x \cup (i, t'_{i,cheap})) - obj_{aug}(x))$
Lowest res. demand 1 (LRD1)	$i = \arg\min_{i \in I \setminus I_x} (r_{total}(x \cup (i, t'_{i,cheap})) - r_{total}(x))$
Lowest res. demand 2 (LRD2)	$i = \arg\min_{i \in I \setminus I_x} (r_{total}(x \cup (i, t'_{i,lrd})) - r_{total}(x))$
Highest resource demand (HRD)	$i = \arg\max_{i \in I \setminus I_x} (r_{total}(x \cup (i, t'_{i,cheap})) - r_{total}(x))$
Shortest 1 (SH1)	$i = \arg\min_{i \in I \setminus I_x} \Delta_{i,t'_{i,cheap}}$
Shortest 2 (SH2)	$i = \arg\min_{i \in I \setminus I_x} \Delta_{i,t'_{i,short}}$
Longest 1 (LN1)	$i = \arg\max_{i \in I \setminus I_x} \Delta_{i,t'_{i,cheap}}$
Longest 2 (LN2)	$i = \arg\max_{i \in I \setminus I_x} \Delta_{i,t'_{i,long}}$
Least exclusions (LEX)	$i = \arg\min_{i \in I \setminus I_x} \lvert Exc_i \rvert$
Most exclusions (MEX)	$i = \arg\max_{i \in I \setminus I_x} \lvert Exc_i \rvert$
Least used (LU)	$i = \arg\min_{i \in I \setminus I_x} removed_i$
Most used (MU)	$i = \arg\max_{i \in I \setminus I_x} removed_i$

rule. For example, the LRD2 repair heuristic schedules the intervention i to $t'_{i,lrd}$, which causes the lowest increase in total resource demand, regardless of increase in obj_{aug}.

All repair heuristics can also be used as an initial solution construction procedure by simply calling the heuristic $|I|$ times, starting with an empty solution x. This is used in each restart of the ALNS metaheuristic (line 4, Algorithm 1). The repair heuristic selected to serve as a construction procedure is the longest 1 (LN1) heuristic. The tuning process that results in this choice is described in Sect. 5.3.

4.4 Hybridization of repair heuristics

The repair heuristics introduced so far are all deterministic, with the only exception of the trivial RND repair heuristic. The same holds for the start time selection mechanisms. Therefore, applying the same repair heuristic to the same partial solution will, in most cases, produce the same complete solution, which may cause stagnation of the ALNS search process. To prevent this, the selection of the intervention and the selection of the start time is further randomized through the following mechanisms, adopted from Smith and Imeson (2017).

4.4.1 Randomization of start time selection

When selecting an intervention $i \in I \backslash I_x$ to schedule, most repair heuristics consider the insertion costs at the cheapest start time $t'_{i,cheap}$, defined in Table 2. The start time selection mechanism can be easily randomized by adding noise to the insertion cost using the following formula:

$$t'_{i,cheap} = \arg\min_{t \in H}((1 + rand)(obj'_{aug})),$$

where $obj'_{aug} = obj_{aug}(x \cup (i,t)) - obj_{aug}(x)$ is the insertion cost of scheduling an intervention i at time t in a partial solution x, $rand$ is a number generated uniformly randomly from $[0, \eta]$ each time the insertion cost is evaluated, and η is a solver parameter. If $\eta = 0$, the described start time selection mechanism becomes deterministic. Otherwise, $\eta = H$, where $H \in (0, 0.5]$ is a tunable solver parameter. The three remaining start time selection mechanisms could be randomized analogically. However, each is used only in a single repair heuristic; thus, their randomization was omitted.

4.4.2 Randomization of intervention selection

The selection of interventions in some repair heuristics is randomized similarly to the start time. Instead of always selecting an intervention $i \in I \backslash I_x$ that minimizes or maximizes a certain property (e.g., minimizes resource demand in the case of the LRD1 heuristic), the following mechanism is used. All interventions from $I \backslash I_x$ are sorted according to the property currently considered (e.g., lowest resource demand to the highest resource demand for LRD1). Then, an index $k \in \{1, 2..., l\}$ is randomly selected according to the unnormalized probability mass function $[\mu^0, \mu^1, ...\mu^{l-1}]$, where $l = |I \backslash I_x|$ and μ is a solver parameter. After that, the intervention with the $k - th$ lowest value of the currently considered property is selected.

The value of μ determines the behaviour of the repair heuristic, as it controls whether an intervention with the smallest or largest possible property value is most likely to be selected. If $\mu = 0$, the selection process is deterministic, and the intervention with the lowest property value is always selected. When $\mu \in (0, 1)$, the selection strongly favours the few interventions with the lowest property values. The $\mu = 1$ corresponds

Table 4 Hybridized repair heuristics

	$\eta = 0$	$\eta = H$
$\mu = 0$	CH, LRD1, SH1, LEX, LU, ME, HRD, LM1, MEX, MU	H1-RND, H1-CH, H1-LRD1, H1-SH1, H1-LEX, H1-LU, H1-ME, H1-HRD, H1-LM1, H1-MEX, H1-MU
$\mu = M_1$	H2-CH, H2-LRD1, H2-SH1, H2-LEX, H2-LU	H3-CH, H3-LRD1, H3-SH1, H3-LEX, H3-LU
$\mu = M_2$	H2-ME, H2-HRD, H2-LN1, H2-MEX, H2-MU	H3-ME, H3-HRD, H3-LN1, H3-MEX, H3-MU

to a uniform random selection. Finally, $\mu > 1$ favours interventions with the highest property values.

In the solver, two separate values of μ are used to parameterize some of the repair heuristics: $M_1 \in (0, 1)$ and $M_2 > 1$ for the selection of interventions with some minimal, respectively, maximal properties. The value M_1 is used in the heuristics intended to assign a high selection probability to interventions with low property values (e.g., LRD1), while M_2 is used in the heuristics that assign a high selection probability to interventions with high property values (e.g., HRD).

4.4.3 Introduced hybrid heuristics

Two previously described mechanisms were combined to create a large bank of randomized repair heuristics. In addition to the deterministic heuristics described in Sect. 4.3.2, the following randomized hybrid variants were added.

The first set of hybrid heuristics uses only the randomized start time selection mechanism, determined by the parameter $\eta = H$. The intervention selection mechanism remains deterministic, therefore $\mu = 0$. These heuristics are marked with the H1 prefix. The second set uses only the randomization of the intervention selection, determined by the parameter M_1 or M_2. These heuristics are marked with the H2 prefix. The third set, marked with the H3 prefix, uses both randomization mechanisms at the same time. Note that M_1 is used to hybridize the minimizing heuristics (CH, LRD1, SH1, LEX, LU) and M_2 the maximizing (ME, HRD, LM1, MEX, MU). The tuning of the actual values of the parameters H, M_1, and M_2 is described in Sect. 5.3. All introduced heuristics, including the original deterministic variants, are classified in Table 4.

4.5 Repair heuristics speed up

Some repair heuristics select an intervention to schedule based on some time-independent properties of the intervention, such as the number of exclusions (LEX, MEX, and their variants), their previous usage (LU, MU, and their variants), or completely randomly (RND). Most heuristics, however, evaluate some intervention property at all possible intervention start times and select the best start time possible. This was introduced in Sect. 4.3.2 as start time selection mechanisms. As the best possible start time must be determined for each unscheduled intervention in $i \in I \backslash I_x$, these heuristics have a higher time complexity than those based on time-independent properties.

The properties considered (length, cost, or resource demand) tend to be relatively similar across all possible start times: a very long intervention will probably be more expensive than a very short one, regardless of their start times. Using this assumption, the set of currently unscheduled interventions $I \backslash I_x$ considered by a repair heuristic is reduced before applying the heuristic. For this purpose, the average resource demand, cost, and length are precomputed for all interventions. Then, the repair heuristics consider only a candidate subset of fixed size $I_c \subset I \backslash I_x$, where $|I_c| = l_c$. The interventions in I_c are preselected according to their average values of the property currently considered (e.g., I_c for the CH heuristic contains only l_c interventions from $I \backslash I_x$ with the lowest average cost, while I_c for the HRD heuristic contains only l_c interventions with the highest resource demand).

The value of l_c is defined as

$$l_c = \min(batch\_size * |I|, |I \backslash I_x|),$$

where $batch\_size \in (0, 1]$. The value of $batch\_size$ is parametrized separately for each property: $batch\_size$ = `LENGTH_BATCH` is used for those heuristics that use the duration of the intervention, $batch\_size$ = `COST_BATCH` for those that use cost, and $batch\_size$ = `RD_BATCH` for those that use resource demand.

4.6 Local search

The purpose of the local search is to reach a local optima with respect to multiple neighborhoods, after the current solution x has been partially destroyed and recreated (line 10 of Algorithm 1). It enables further refinement of a current solution, which could not be achieved solely by the main ALNS ruin-and-recreate operation. The local search is controlled by the Randomized Variable Neighborhood Descent heuristic (RVND, Duarte et al. 2018), which is described in Algorithm 2.

Algorithm 2 RVND heuristic

```
    Input: solution $x$
1:  $improved \leftarrow$ true
2:  while $improved$ do
3:      shuffle($N$)
4:      for $k \leftarrow 1$ to $k_{max}$ do
5:          $improved \leftarrow$ false
6:          $x' \leftarrow$ best_improve($N_k(x)$)
7:          if $obj_{aug}(x') < obj_{aug}(x)$ then
8:              $x \leftarrow x'$
9:              $improved \leftarrow$ true
10:             break
    return $x$
```

The RVND repeatedly performs the following process until no improvement is achieved in any of the neighborhoods N. First, the k neighborhoods in N are randomly shuffled (line 3). Then, the neighborhoods in N are sequentially searched for the most improving move (lines 4–6). If a better solution x' is found, the process is restarted (lines 7–10). Otherwise, a locally optimal solution is reached, and the local search terminates. Three local search operators, corresponding to different neighborhoods in N, are specifically proposed for the TMS problem.

1-shift (1SH) reschedules a single intervention i' to a new start time t'. The pair with the best improvement (i', t') is selected according to the following formula:

$$i', t' = \arg \max_{i \in I_c, t \in H} (obj_{aug}(x) - obj_{aug}(x \backslash i \cup (i, t))).$$

The improved solution x' is then obtained as $x' = x \backslash i' \cup (i', t')$. In each iteration of the RVND, the 1SH operator considers a randomly sampled subset $I_c \subset I_x$. The size of I_c is given by $|I_c| = \texttt{ONE\_SHIFT\_LIMIT} * |I_x|$, where $\texttt{ONE\_SHIFT\_LIMIT} \in [0, 1]$ is a tunable solver parameter introduced to control the intensity of the operator.

Random 2-shift (2SH-R) reschedules two randomly selected interventions $i_1, i_2 \in I_x$. The most improving start times $t'_1, t'_2 \in H$ are selected as follows:

$$t'_1, t'_2 = \arg \max_{t_1, t_2 \in H} (obj_{aug}(x) - obj_{aug}(x')).$$

The improved solution is then $x' = x \backslash \{i_1, i_2\} \cup \{(i_1, t'_1), (i_2, t'_2)\})$. The 2SH-R operator is applied `2_SHIFT_LIMIT` times in each RVND iteration, where `2_SHIFT_LIMIT` $\in \mathbb{N}$ is a tunable solver parameter.

Exclusion 2-shift (2SH-E) is a variant of the 2SH-R operator, which reschedules two randomly selected interventions $i_1, i_2 \in Exc$. The 2SH-E operator is applied only if the exclusion penalty e_{pen} of a current solution x is nonzero, and thus some exclusivity constraints are violated. Again, the parameter `2_SHIFT_LIMIT` controls the intensity of this operator.

5 Results and discussion

This section describes the testing setup in Sect. 5.1, the competition datasets provided in Sect. 5.2, and the tuning process in Sect. 5.3. The performance of the proposed ALNS metaheuristic on all competition datasets is documented in Sect. 5, together with a summary of the competition results and a comparison with the qualification method. Finally, Sect. 5.5 provides an in-depth analysis of the behavior of ALNS and evaluates the benefit of its individual components.

5.1 Testing setup

The algorithm is implemented in C++. All results are obtained on a Linux computer with an Intel Core i7-7700 3.60 GHz processor. According to the competition rules, the instances are solved in a short 15-minute run and a long 90-min run. In this paper,

Table 5 Datasets

Dataset A id: $\|I\|$, $\|C\|$, T, $\|Exc\|$	Dataset B id: $\|I\|$, $\|C\|$, T, $\|Exc\|$	Dataset C id: $\|I\|$, $\|C\|$, T, $\|Exc\|$	Dataset X id: $\|I\|$, $\|C\|$, T, $\|Exc\|$
A01: 181, 9, 90, 48	B01: 100, 9, 53, 19	C01: 120, 9, 53, 36	X01: 120, 9, 53, 30
A02: 89, 9, 90, 25	B02: 100, 9, 53, 14	C02: 120, 9, 53, 22	X02: 706, 9, 53, 792
A03: 91, 10, 90, 8	B03: 706, 9, 53, 764	C03: 706, 9, 53, 787	X03: 280, 9, 53, 111
A04: 706, 9, 365, 846	B04: 706, 9, 53, 764	C04: 706, 9, 53, 771	X04: 426, 9, 25, 327
A05: 180, 9, 182, 53	B05: 706, 9, 53, 846	C05: 706, 9, 53, 846	X05: 467, 9, 220, 390
A06: 180, 10, 182, 53	B06: 100, 9, 53, 14	C06: 280, 9, 53, 102	X06: 528, 9, 300, 472
A07: 36, 9, 17, 2	B07: 250, 9, 53, 124	C07: 120, 9, 42, 26	X07: 209, 9, 300, 51
A08: 18, 9, 17, 2	B08: 119, 9, 42, 18	C08: 426, 9, 25, 211	X08: 209, 9, 300, 32
A09: 18, 10, 17, 0	B09: 120, 9, 42, 30	C09: 110, 9, 53, 19	X09: 548, 9, 30, 539
A10: 108, 9, 53, 30	B10: 398, 9, 25, 249	C10: 522, 9, 102, 480	X10: 460, 9, 35, 338
A11: 54, 9, 53, 4	B11: 100, 9, 53, 21	C11: 89, 9, 102, 28	X11: 521, 9, 131, 462
A12: 54, 10, 53, 0	B12: 495, 9, 102, 394	C12: 298, 9, 191, 124	X12: 522, 9, 131, 479
A13: 179, 9, 90, 83	B13: 99, 9, 102, 2	C13: 505, 9, 230, 356	X13: 336, 9, 212, 163
A14: 108, 10, 53, 13	B14: 297, 9, 191, 142	C14: 465, 9, 220, 414	X14: 613, 9, 180, 611
A15: 108, 10, 53, 13	B15: 495, 9, 250, 425	C15: 528, 9, 300, 409	X15: 613, 9, 180, 601

30 short runs and 5 long runs are carried out for each instance. The random number generator is seeded with consecutive integers, starting with 1 – {1, 2, ..., 30} for short runs and {1, 2, ..., 5} for long runs. The Best Known Scores (BKS) are taken from the best solutions found in the competition in 90-minute runs. The BKSs were obtained by various methods from other competitors.

5.2 Datasets

Four datasets, each consisting of 15 instances, were provided throughout the competition. Each competition phase was evaluated on a different dataset: sprint phase on A, qualification on A, semifinal on B, and final on C and X. Dataset X was not available before submitting the final program. The following basic properties of each instance are listed in Table 5: number of interventions $|I|$, number of resources $|C|$, planning horizon T, and number of exclusions $|Exc|$. All datasets are publicly available at ROADEF (2020).

5.3 Parameters tuning

The algorithm has numerous parameters, which were tuned specifically for a particular dataset during the competition. Similarly, the usage of individual destroy and repair heuristics, local search operators, and initial construction was also parametrized and tuned. For this purpose, the iterated racing procedure, implemented within the irace package (López-Ibáñez et al. 2016), was used. The results presented in this paper are generated using a common configuration, which was generated after the competition using all four competition datasets. Although tuning for a particular dataset or even an instance would provide better results, this approach is less prone to overfitting and allows a more objective assessment of the method's performance. The final configuration is provided in Table 6.

5.4 Competition results

This section presents detailed results obtained by the final proposed method on all four competition datasets. Performance in the short and long runs is evaluated separately. For each instance, the basic statistical measures (min score, mean score, and standard deviation) are shown. The results also contain gap values calculated relative to the BKS. The gap is given in percents (%) and calculated as $gap = 100 \times \frac{score - BKS}{BKS}$, where $score$ is either the best value (min gap) or the average value in all runs (mean gap). As the method does not guarantee the finding of a valid solution, a success rate measure is provided for datasets C and X, where the method did not succeed every time. The success rate is given as $100 \times \frac{\text{no. of successful runs}}{\text{total no. of runs}}$ (%). The final method is compared with the qualification method described in Woller and Kulich (2021) on datasets A and B, for which the qualification method was tuned, and with the results of the winner of the competition, team S34, on datasets C and X. Finally, a summary of the final results of the competition is presented, which contains 13 different methods. When different methods are compared, hypothesis testing is performed. For this purpose, the paired t-test is employed. Hypothesis H1 states that the average of gaps of the final ALNS method is better (thus, lower) than that of a reference method, and the confidence level CL (%) of H1 being true is provided.

Tables 7 and 8 show results on the qualification dataset A. ALNS reached BKS (or found a solution within the 0.01% gap) in the majority of instances both in the short and long runs. On average, the mean gap was 0.19% in short runs and 0.10% in long runs. Therefore, the ALNS can be considered highly efficient on this dataset. Both tables also show the results of the method submitted in the qualification phase, which was described in Woller and Kulich (2021). Here, the symbol $\uparrow$ indicates that the qualification method performed worse than the final method, whereas the symbol $\downarrow$ indicates the opposite and no symbol means achieving an identical result by both methods. It can be seen that the average values of the min and mean gaps obtained by the final method are better than those obtained by the qualification method, although the difference is close to 0.1%. The confidence level of the final method being better than the qualification method is 92.71% (15-minute run), respectively 97.76% (90-minute run).

Tables 9 and 10 show results on the semifinal dataset B. The ALNS generally remains within the 0.76% gap in short runs and 0.59% in long runs. Again, the results of the qualification method are provided. Although the qualification method found better solutions in some cases, the average mean gap of the final method is lower by 1.14% in the short runs and by 0.73% in the long runs, with confidence levels of 89.12%, respectively 86.03%. Although the average mean gap was reduced by half

Table 6 Final configuration

Parameters	Values
ALNS	DEPTH $= 0.134$, ITERS_MAX $= 881$, $\lambda = 0.791$, $\Omega_1 = 89.552$, $\Omega_2 = 26.452$, $\Omega_3 = 0.464$
obj_{aug}	$\beta_{lower} = 63614$, $\beta_{upper} = 35265$, $\gamma = 25881$
Destroy heur	RND, MEX, LRD, SH, MEX, LEX, LU
Repair heur	RND, CH, ME, LRD1, LRD2, LN1, LN2, MEX, MU, H1-ME, H1-RND, H1-MEX, H2-LRD1, H2-HRD, H2-LN1, H2-SH1, H2-MEX, H2-LEX, H2-LU, H3-CH, H3-MEX,H3-SH1,H3-MEX, H3-LEX, H3-MU, H3-LU
Construction	LN1
Hybridization	$H = 0.117$, $M_1 = 0.938$, $M_2 = 190.7$
rep. speed-up	LENGTH_BATCH $= 0.224$, COST_BATCH $= 0.585$, RD_BATCH $= 0.528$
LS operators	1SH, 2SH-E
LS params	ONE_SHIFT_LIMIT $= 0.433$, TWO_SHIFT_LIMIT $= 14$

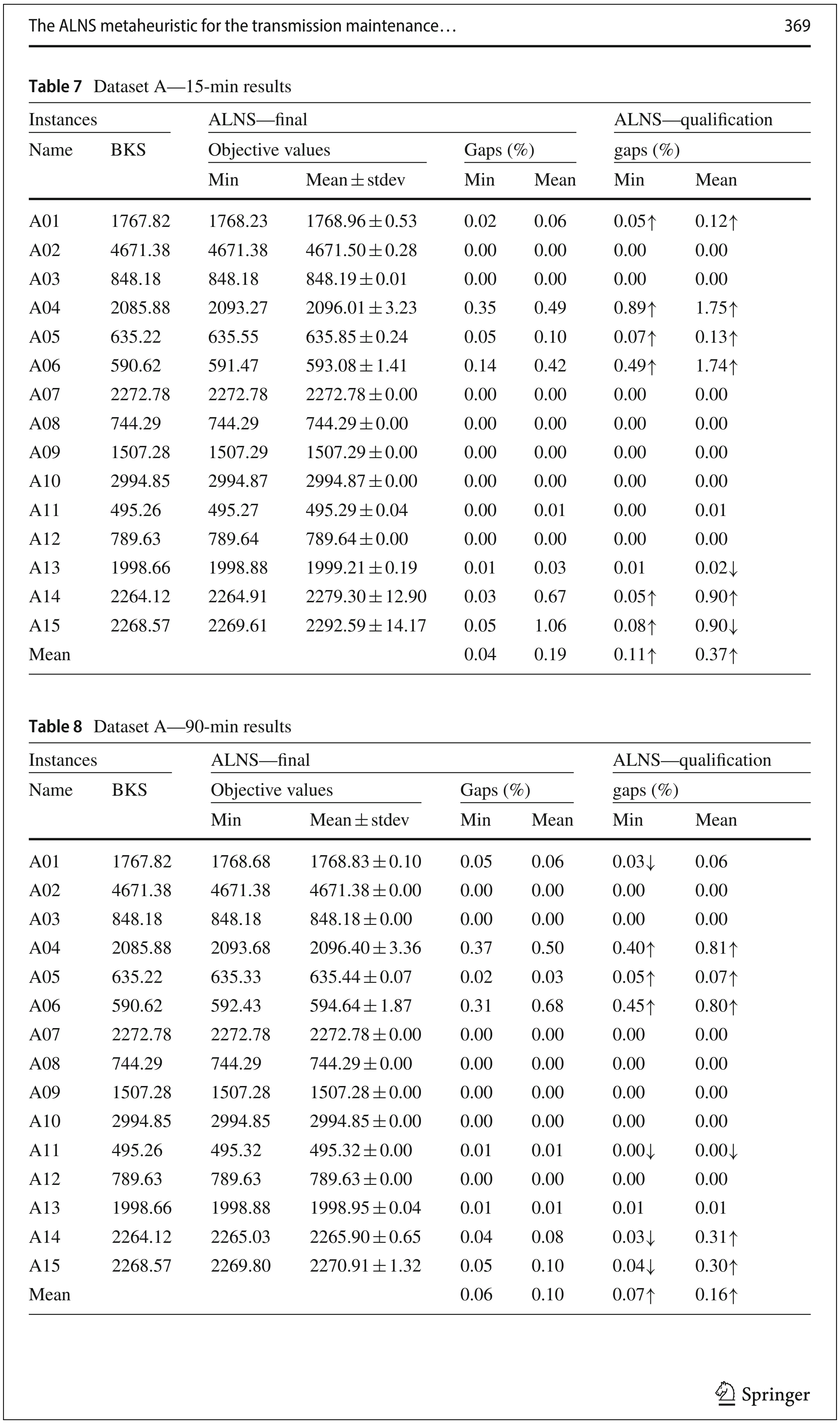


Table 7 Dataset A—15-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
A01	1767.82	1768.23	1768.96 ± 0.53	0.02	0.06	0.05↑	0.12↑
A02	4671.38	4671.38	4671.50 ± 0.28	0.00	0.00	0.00	0.00
A03	848.18	848.18	848.19 ± 0.01	0.00	0.00	0.00	0.00
A04	2085.88	2093.27	2096.01 ± 3.23	0.35	0.49	0.89↑	1.75↑
A05	635.22	635.55	635.85 ± 0.24	0.05	0.10	0.07↑	0.13↑
A06	590.62	591.47	593.08 ± 1.41	0.14	0.42	0.49↑	1.74↑
A07	2272.78	2272.78	2272.78 ± 0.00	0.00	0.00	0.00	0.00
A08	744.29	744.29	744.29 ± 0.00	0.00	0.00	0.00	0.00
A09	1507.28	1507.29	1507.29 ± 0.00	0.00	0.00	0.00	0.00
A10	2994.85	2994.87	2994.87 ± 0.00	0.00	0.00	0.00	0.00
A11	495.26	495.27	495.29 ± 0.04	0.00	0.01	0.00	0.01
A12	789.63	789.64	789.64 ± 0.00	0.00	0.00	0.00	0.00
A13	1998.66	1998.88	1999.21 ± 0.19	0.01	0.03	0.01	0.02↓
A14	2264.12	2264.91	2279.30 ± 12.90	0.03	0.67	0.05↑	0.90↑
A15	2268.57	2269.61	2292.59 ± 14.17	0.05	1.06	0.08↑	0.90↓
Mean				0.04	0.19	0.11↑	0.37↑

Table 8 Dataset A—90-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
A01	1767.82	1768.68	1768.83 ± 0.10	0.05	0.06	0.03↓	0.06
A02	4671.38	4671.38	4671.38 ± 0.00	0.00	0.00	0.00	0.00
A03	848.18	848.18	848.18 ± 0.00	0.00	0.00	0.00	0.00
A04	2085.88	2093.68	2096.40 ± 3.36	0.37	0.50	0.40↑	0.81↑
A05	635.22	635.33	635.44 ± 0.07	0.02	0.03	0.05↑	0.07↑
A06	590.62	592.43	594.64 ± 1.87	0.31	0.68	0.45↑	0.80↑
A07	2272.78	2272.78	2272.78 ± 0.00	0.00	0.00	0.00	0.00
A08	744.29	744.29	744.29 ± 0.00	0.00	0.00	0.00	0.00
A09	1507.28	1507.28	1507.28 ± 0.00	0.00	0.00	0.00	0.00
A10	2994.85	2994.85	2994.85 ± 0.00	0.00	0.00	0.00	0.00
A11	495.26	495.32	495.32 ± 0.00	0.01	0.01	0.00↓	0.00↓
A12	789.63	789.63	789.63 ± 0.00	0.00	0.00	0.00	0.00
A13	1998.66	1998.88	1998.95 ± 0.04	0.01	0.01	0.01	0.01
A14	2264.12	2265.03	2265.90 ± 0.65	0.04	0.08	0.03↓	0.31↑
A15	2268.57	2269.80	2270.91 ± 1.32	0.05	0.10	0.04↓	0.30↑
Mean				0.06	0.10	0.07↑	0.16↑

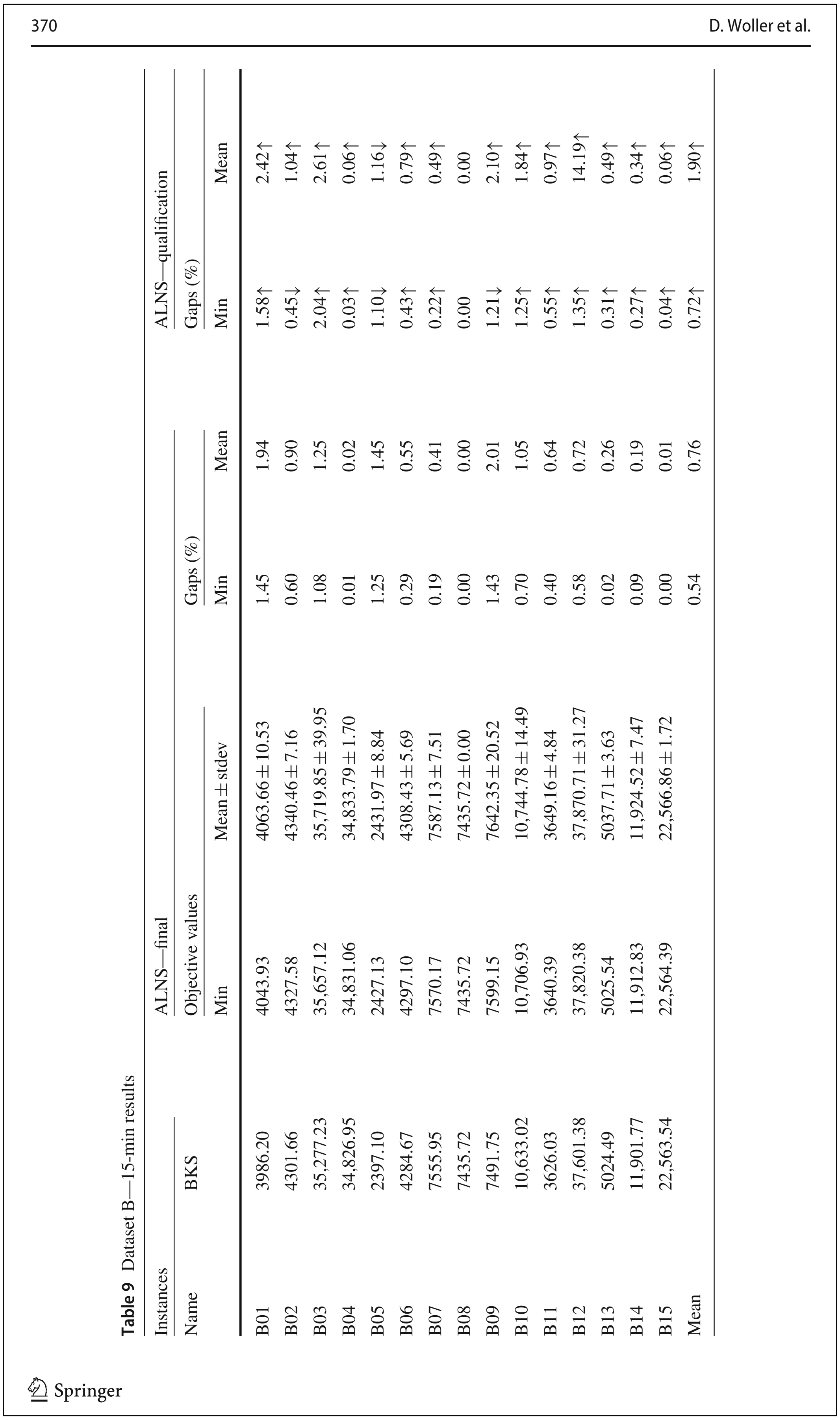


Table 9 Dataset B—15-min results

Instances		ALNS—final				ALNS—qualification	
		Objective values		Gaps (%)		Gaps (%)	
Name	BKS	Min	Mean ± stdev	Min	Mean	Min	Mean
B01	3986.20	4043.93	4063.66 ± 10.53	1.45	1.94	1.58↑	2.42↑
B02	4301.66	4327.58	4340.46 ± 7.16	0.60	0.90	0.45↓	1.04↑
B03	35,277.23	35,657.12	35,719.85 ± 39.95	1.08	1.25	2.04↑	2.61↑
B04	34,826.95	34,831.06	34,833.79 ± 1.70	0.01	0.02	0.03↑	0.06↑
B05	2397.10	2427.13	2431.97 ± 8.84	1.25	1.45	1.10↓	1.16↓
B06	4284.67	4297.10	4308.43 ± 5.69	0.29	0.55	0.43↑	0.79↑
B07	7555.95	7570.17	7587.13 ± 7.51	0.19	0.41	0.22↑	0.49↑
B08	7435.72	7435.72	7435.72 ± 0.00	0.00	0.00	0.00	0.00
B09	7491.75	7599.15	7642.35 ± 20.52	1.43	2.01	1.21↓	2.10↑
B10	10,633.02	10,706.93	10,744.78 ± 14.49	0.70	1.05	1.25↑	1.84↑
B11	3626.03	3640.39	3649.16 ± 4.84	0.40	0.64	0.55↑	0.97↑
B12	37,601.38	37,820.38	37,870.71 ± 31.27	0.58	0.72	1.35↑	14.19↑
B13	5024.49	5025.54	5037.71 ± 3.63	0.02	0.26	0.31↑	0.49↑
B14	11,901.77	11,912.83	11,924.52 ± 7.47	0.09	0.19	0.27↑	0.34↑
B15	22,563.54	22,564.39	22,566.86 ± 1.72	0.00	0.01	0.04↑	0.06↑
Mean				0.54	0.76	0.72↑	1.90↑

Table 10 Dataset B—90-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		Gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
B01	3986.20	4044.04	4047.57 ± 4.07	1.45	1.54	1.13↓	1.71↑
B02	4301.66	4321.71	4333.18 ± 8.43	0.47	0.73	0.24↓	0.42↓
B03	35,277.23	35,585.40	35,628.56 ± 26.32	0.87	1.00	1.60↑	1.86↑
B04	34,826.95	34,830.40	34,831.86 ± 0.91	0.01	0.01	0.01	0.02↑
B05	2397.10	2420.33	2425.48 ± 3.01	0.97	1.18	1.03↑	1.08↓
B06	4284.67	4294.41	4300.43 ± 5.13	0.23	0.37	0.03↓	0.31↓
B07	7555.95	7569.85	7577.99 ± 7.90	0.18	0.29	0.02↓	0.34↑
B08	7435.72	7435.72	7435.72 ± 0.00	0.00	0.00	0.00	0.00
B09	7491.75	7610.53	7616.32 ± 6.35	1.59	1.66	0.95↓	1.44↓
B10	10,633.02	10,720.15	10,725.87 ± 4.76	0.82	0.87	1.24↑	1.40↑
B11	3626.03	3639.19	3642.11 ± 2.62	0.36	0.44	0.32↓	0.63↑
B12	37,601.38	37,694.28	37,739.04 ± 26.19	0.25	0.37	0.79↑	10.25↑
B13	5024.49	5033.33	5034.53 ± 1.13	0.18	0.20	0.07↓	0.21↑
B14	11,901.77	11,911.26	11,916.43 ± 6.07	0.08	0.12	0.11↑	0.18↑
B15	22,563.54	22,563.83	22,564.43 ± 0.58	0.00	0.00	0.01↑	0.02↑
Mean				0.50	0.59	0.50	1.32↑

Table 11 Dataset C—15-min results

Instances		ALNS - final					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
C01	8515.90	8524.87	8565.86 ± 17.04	0.11	0.59	100	0.00
C02	3539.80	3561.53	3574.81 ± 6.32	0.61	0.99	100	0.06
C03	33,512.26	34,013.56	34,133.22 ± 60.39	1.50	1.85	100	0.00
C04	37,586.31	37,591.07	37,599.35 ± 4.99	0.01	0.03	100	0.00
C05	3166.18	3183.77	3188.11 ± 2.21	0.56	0.69	100	0.03
C06	8394.48	8485.49	8516.27 ± 15.83	1.08	1.45	100	0.02
C07	6083.04	6116.58	6146.59 ± 20.12	0.55	1.04	100	0.04
C08	11,155.64	11,329.41	11,374.24 ± 22.16	1.56	1.96	100	0.06
C09	5585.65	5619.92	5647.14 ± 14.25	0.61	1.10	100	0.28
C10	43,341.84	44,165.64	44,424.34 ± 123.76	1.90	2.50	100	0.00
C11	5749.96	5757.73	5776.58 ± 7.48	0.14	0.46	100	0.00
C12	12,718.79	12,778.96	12,804.79 ± 11.50	0.47	0.68	100	0.02
C13	42,484.56	43,504.11	43,576.51 ± 47.53	2.40	2.57	97	0.01
C14	26,457.11	26,925.26	26,951.30 ± 21.51	1.77	1.87	43	0.04
C15	39,757.55	40,345.14	40,435.46 ± 46.11	1.48	1.71	70	0.01
Mean				0.98	1.30	94	0.04

Table 12 Dataset C—90-min results

Instances		ALNS—final					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
C01	8515.90	8528.53	8540.50 ± 8.05	0.15	0.29	100	0.00
C02	3539.80	3558.16	3566.61 ± 6.70	0.52	0.76	100	0.00
C03	33,512.26	33,864.63	33,897.94 ± 42.40	1.05	1.15	100	0.00
C04	37,586.31	37,589.96	37,592.46 ± 2.92	0.01	0.02	100	0.00
C05	3166.18	3182.76	3186.49 ± 3.25	0.52	0.64	100	0.00
C06	8394.48	8483.00	8496.60 ± 12.22	1.05	1.22	100	0.00
C07	6083.04	6112.08	6125.20 ± 7.65	0.48	0.69	100	0.00
C08	11,155.64	11,319.23	11,328.73 ± 5.50	1.47	1.55	100	0.00
C09	5585.65	5618.38	5628.46 ± 12.94	0.59	0.77	100	0.15
C10	43,341.84	43,879.90	43,964.73 ± 97.83	1.24	1.44	100	0.00
C11	5749.96	5755.15	5762.74 ± 5.37	0.09	0.22	100	0.00
C12	12,718.79	12,753.57	12,763.37 ± 9.85	0.27	0.35	100	0.00
C13	42,484.56	43,167.98	43,195.75 ± 26.98	1.61	1.67	100	0.00
C14	26,457.11	26,751.71	26,775.25 ± 20.22	1.11	1.20	100	0.00
C15	39,757.55	40,098.38	40,154.43 ± 50.32	0.86	1.00	100	0.00
Mean				0.73	0.86	100	0.01

when using the final method, the statistical significance decreased compared to dataset A.

Tables 11 and 12 show the results obtained for the final public dataset C. Here, the ALNS occasionally dropped below 100% success rate in short runs (instances C13, C14, and C15), but maintained 100% in long runs. The average minimal and mean gaps are still within 1% of the BKS in long runs and within 1.3% in short runs. Therefore, the quality of the solution can still be considered very good. Both tables also contain minimal gaps obtained by the winning team S34, which produced the majority of the BKSs in the long runs.

Finally, Tables 13 and 14 show the results obtained for the final dataset X. This dataset proves to be more demanding than the first three datasets, especially in terms of constraint satisfaction. The average success rate is 91% in the short runs and 95% in the long runs. Regarding solution quality, the average gaps are close to 2% in long runs and above 2% in short runs. Therefore, X instances seem rather challenging for the ALNS, although decent solutions can still be obtained within given time limits. Again, the winning team S34 found most of the BKSs.

The results of the final phase of the competition, evaluated on datasets C and X, are summarized in Table 15. Some implementations were not shared. Thus, the results are adopted from the competition website (ROADEF 2020). In total, 13 of 74 registered teams advanced to the final phase. For each team, the mean gap, confidence level CL and success rate across both datasets are presented, together with the competition score and the final ranking. The competition score is defined in ROADEF (2020). A team could obtain from 0 to 10 points per instance, 300 in total. A team loses a point for each competitor that achieves strictly better combined objective value in the short and long run on a given instance.

The confidence level CL indicates the probability that the overall mean gap of the proposed ALNS method is better than that of the reference method. Each instance was solved by each method only once in the final phase. Multiple methods, including ALNS, did not reach a 100% success rate on some instances. The results of the proposed ALNS, as submitted to the final phase, are shown under team ID J49 and ranked 8th-9th according to the competition scoring. Due to numerical issues in the submitted implementation, the final version produced a large number of invalid solutions on the hidden dataset X. The results shown under the ID J49* were generated after the competition using the parameter setup from this paper with the discovered issues fixed. CL values are calculated using the J49* results to provide a more accurate statistical comparison.

The CL values are consistent with the final competition ranking. Some of the methods are publicly available at ROADEF (2020), although not yet described in a publication. The winning team S34 combined Mixed Integer Programming (MIP) solver and the custom Local Search, where the MIP was used to find a feasible solution and the Local Search to refine it. The 3rd team S56 combined a custom GRASP metaheuristic with a MIP solver but used GRASP for initial construction and MIP for refining. The 4th team S19 applied purely heuristic A* Local Search, whereas the 5th J73 and 7th S58 focused on MIP relaxation and eventual problem decomposition. No details are available about the remaining submissions.

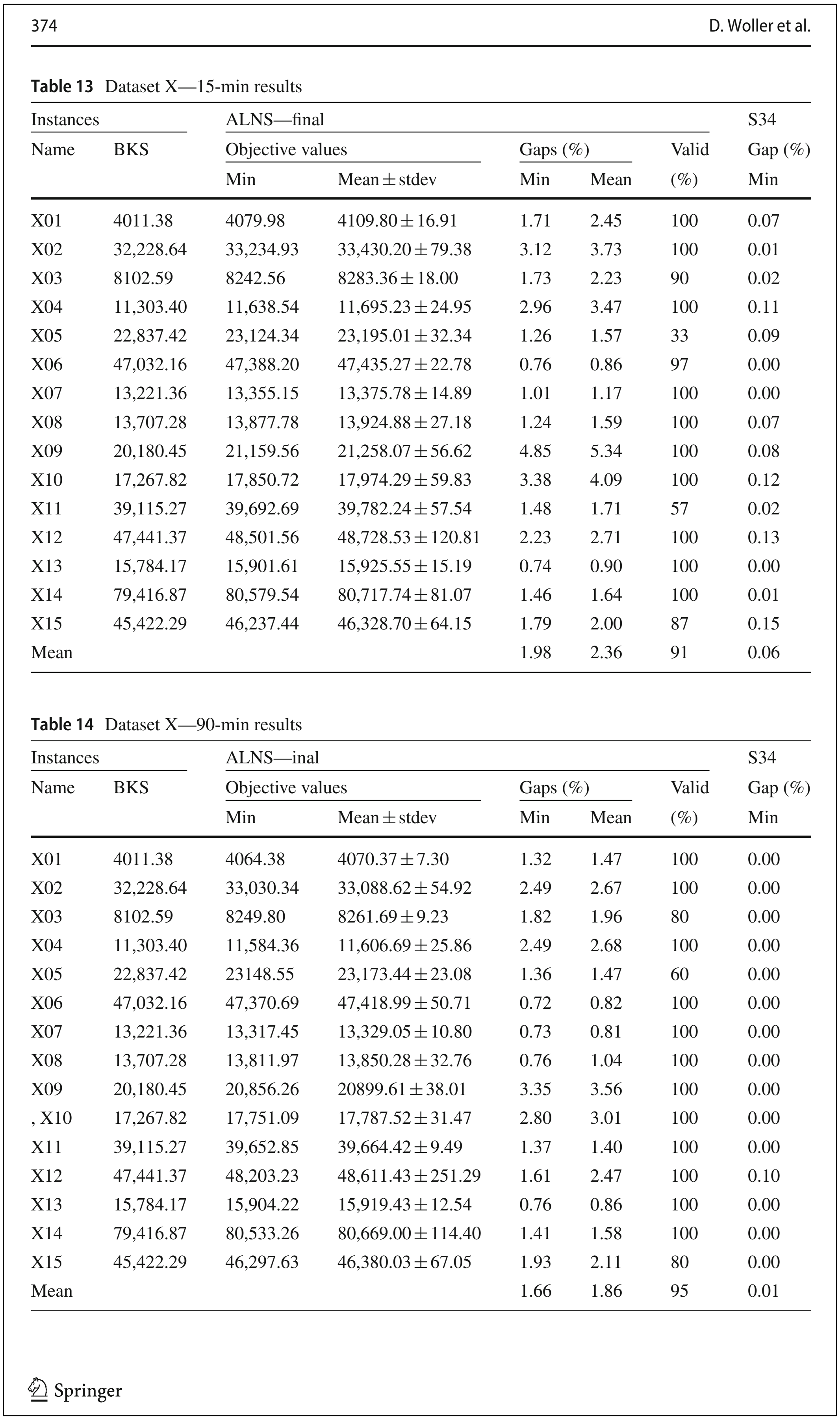


Table 13 Dataset X—15-min results

Instances		ALNS—final					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
X01	4011.38	4079.98	4109.80 ± 16.91	1.71	2.45	100	0.07
X02	32,228.64	33,234.93	33,430.20 ± 79.38	3.12	3.73	100	0.01
X03	8102.59	8242.56	8283.36 ± 18.00	1.73	2.23	90	0.02
X04	11,303.40	11,638.54	11,695.23 ± 24.95	2.96	3.47	100	0.11
X05	22,837.42	23,124.34	23,195.01 ± 32.34	1.26	1.57	33	0.09
X06	47,032.16	47,388.20	47,435.27 ± 22.78	0.76	0.86	97	0.00
X07	13,221.36	13,355.15	13,375.78 ± 14.89	1.01	1.17	100	0.00
X08	13,707.28	13,877.78	13,924.88 ± 27.18	1.24	1.59	100	0.07
X09	20,180.45	21,159.56	21,258.07 ± 56.62	4.85	5.34	100	0.08
X10	17,267.82	17,850.72	17,974.29 ± 59.83	3.38	4.09	100	0.12
X11	39,115.27	39,692.69	39,782.24 ± 57.54	1.48	1.71	57	0.02
X12	47,441.37	48,501.56	48,728.53 ± 120.81	2.23	2.71	100	0.13
X13	15,784.17	15,901.61	15,925.55 ± 15.19	0.74	0.90	100	0.00
X14	79,416.87	80,579.54	80,717.74 ± 81.07	1.46	1.64	100	0.01
X15	45,422.29	46,237.44	46,328.70 ± 64.15	1.79	2.00	87	0.15
Mean				1.98	2.36	91	0.06

Table 14 Dataset X—90-min results

Instances		ALNS—inal					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
X01	4011.38	4064.38	4070.37 ± 7.30	1.32	1.47	100	0.00
X02	32,228.64	33,030.34	33,088.62 ± 54.92	2.49	2.67	100	0.00
X03	8102.59	8249.80	8261.69 ± 9.23	1.82	1.96	80	0.00
X04	11,303.40	11,584.36	11,606.69 ± 25.86	2.49	2.68	100	0.00
X05	22,837.42	23148.55	23,173.44 ± 23.08	1.36	1.47	60	0.00
X06	47,032.16	47,370.69	47,418.99 ± 50.71	0.72	0.82	100	0.00
X07	13,221.36	13,317.45	13,329.05 ± 10.80	0.73	0.81	100	0.00
X08	13,707.28	13,811.97	13,850.28 ± 32.76	0.76	1.04	100	0.00
X09	20,180.45	20,856.26	20899.61 ± 38.01	3.35	3.56	100	0.00
, X10	17,267.82	17,751.09	17,787.52 ± 31.47	2.80	3.01	100	0.00
X11	39,115.27	39,652.85	39,664.42 ± 9.49	1.37	1.40	100	0.00
X12	47,441.37	48,203.23	48,611.43 ± 251.29	1.61	2.47	100	0.10
X13	15,784.17	15,904.22	15,919.43 ± 12.54	0.76	0.86	100	0.00
X14	79,416.87	80,533.26	80,669.00 ± 114.40	1.41	1.58	100	0.00
X15	45,422.29	46,297.63	46,380.03 ± 67.05	1.93	2.11	80	0.00
Mean				1.66	1.86	95	0.01

Table 15 Final results of the competition

Team	15-min results			90-min results			Final results	
	Mean ± stdev Gap (%)	CL (%)	Valid (%)	Mean ± stdev (%)	CL (%)	Valid (%)	Score (−)	Rank (−)
S34	0.05 ± 0.06	0	100	0.01 ± 0.03	0.00	100	291	1
S66	0.11 ± 0.13	0	90	0.07 ± 0.08	0.00	100	236	2
S56	0.14 ± 0.16	0	93	0.07 ± 0.08	0.00	93	225	3
S19	0.35 ± 0.28	0	100	0.25 ± 0.26	0.00	97	187	4
J73	0.95 ± 1.81	6.75	100	0.69 ± 0.93	0.64	100	139	5
S68	0.54 ± 0.55	0.01	90	0.54 ± 0.55	0.02	87	139	5
S58	0.80 ± 1.96	6.20	93	0.27 ± 0.25	0.00	83	138	7
J49	0.80 ± 0.49	45.97	67	0.66 ± 0.42	35.04	77	77	8
J49*	1.48 ± 1.07	–	100	1.20 ± 0.82	–	100	–	–
J43	1.34 ± 1.13	79.01	70	1.45 ± 1.49	82.53	97	77	8
J24	3.23 ± 2.06	100	100	2.71 ± 1.94	100.00	100	49	10
J3	3.09 ± 2.08	100	97	1.93 ± 1.52	99.96	100	48	11
S14	2.46 ± 1.95	99.48	83	2.14 ± 1.73	99.79	97	40	12
S28	2.54 ± 2.13	99.53	53	1.99 ± 1.66	99.68	57	13	13

5.5 ALNS analysis

The behaviour of the ALNS is further analyzed in a series of experiments. The goal of this section is to provide insight into the algorithm operation and assess the benefit of individual components.

The first experiment, presented in Table 16, documents the importance of local search, which is not a standard part of ALNS. The following two setups were tested on dataset X: standard ALNS without local search and isolated local search without the ALNS. The results show that the standard ALNS reaches a mean gap of 2.70%, but with a success rate of only 32% overall and 0% for 7 instances out of 15. The isolated local search achieves a slightly better mean gap of 2.23%, but its success rate on all instances is only 12% and it finds a valid solution only for 5 instances. Thus, both setups perform quite poorly in terms of finding valid solutions. Presumably, the standard ALNS provides a robust diversification mechanism but fails to reach valid local optima without the intensification element of the local search.

The second experiment investigates the benefit brought by individual components used in the final tuned setup on dataset X. The efficiency of a component is estimated as the percentage ratio between the number of improving calls and the total calls of the component. A call is considered improving if it leads to a new overall best solution after destroying, repairing, and local search. The efficiency of destroy heuristics on dataset X is summarized in Fig. 2a. One data point corresponds to a single run on a single instance, and each boxplot summarizes data across all instances and runs for a single heuristic. Individual heuristics are sorted from highest to lowest mean efficiency. The mean efficiency of all heuristics is around 10%, with the random destroy heuristic

Table 16 Dataset X—15-min results, separating components

Instances name	standard ALNS			Isolated local search		
	Gaps (%)		valid	Gaps (%)		Valid
	Min	Mean	(%)	Min	Mean	(%)
X01	2.06	2.61	100	3.02	3.58	20
X02	7.21	8.38	30	–	–	0
X03-X05	–	–	0	–	–	0
X06	1.22	1.41	47	–	–	0
X07	1.20	1.56	100	1.81	1.96	100
X08	1.41	1.73	97	1.51	1.70	30
X09	–	–	0	–	–	0
X10	5.22	5.79	17	–	–	0
X11-X12	–	–	0	–	–	0
X13	1.18	1.32	63	1.39	1.48	30
X14	2.12	2.57	27	2.28	2.42	7
X15	–	–	0	–	–	0
Mean	2.70	3.17	32	2.00	2.23	12

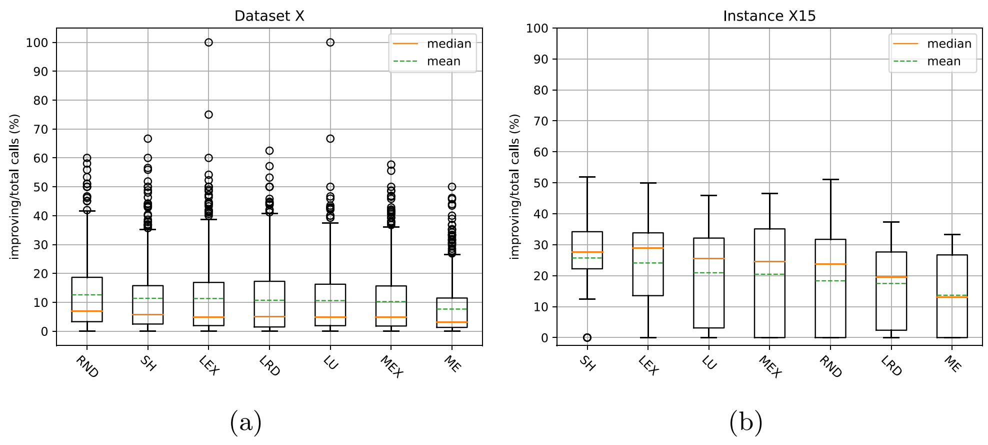


Fig. 2 Destroy heuristics efficiency

being the most successful. Each boxplot contains a large number of outliers, often reaching 50% efficiency. This is also documented in Fig. 2b, which displays the same statistics for instance X15. Here, both the heuristic ranking and their efficiency differ significantly from the averaged data in Fig. 2a. The same analysis was performed for the repair heuristics (Fig. 3) and the local search operators (Fig. 4a). The repair heuristics perform similarly to the destroy heuristics: their mean efficiency is also close to 10% with lots of outliers. In conclusion, all the heuristics not disabled by the automated tuning process have very similar mean efficiency. However, their performance greatly varies from instance to instance, as documented by the large number of outliers. Thanks

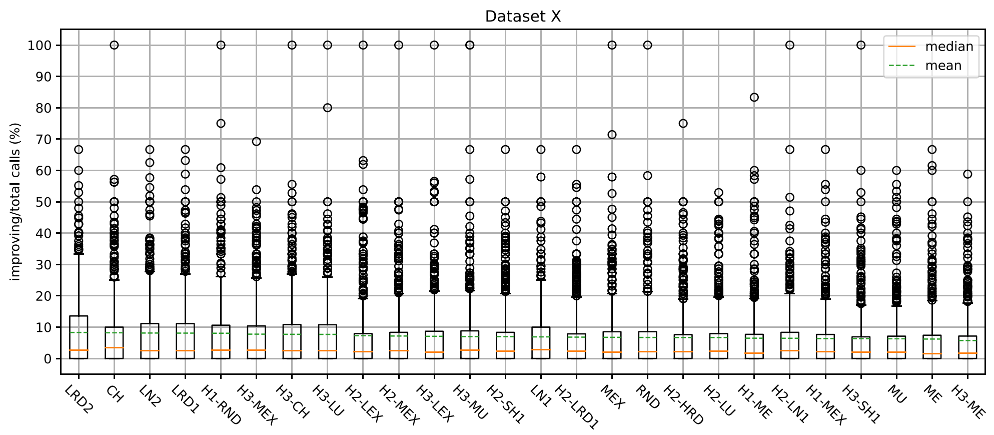
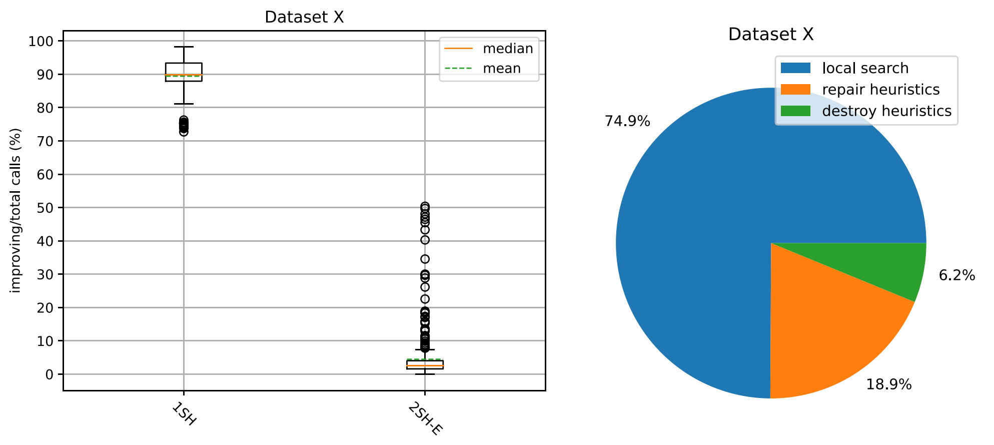


Fig. 3 Repair heuristics eficiency



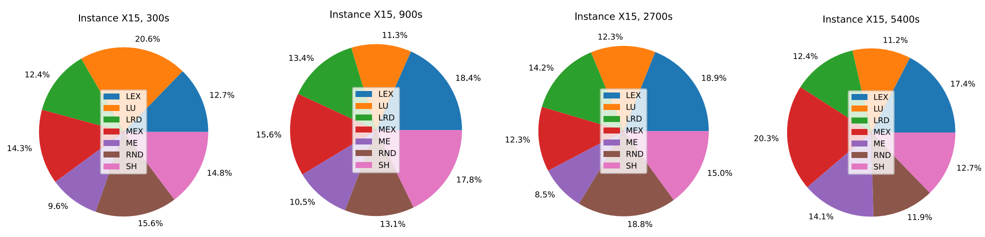
(a) Local search operators efficiency (b) Time spent by components

Fig. 4 Local search

to the ALNS weight-adjusting process, unsuccessful heuristics on a given instance are called less frequently and can be kept in the tuned configuration.

As for local search operators, efficiency was measured by counting improving calls within the RVND heuristic, not the high-level ALNS. The data show that the mean efficiency of the very simple 1-shift operator is close to 90%, whereas the Exclusion 2-shift's is below 5%, although with a large number of outliers.

The third experiment focuses on the CPU time requirements of individual components and is summarized in a single pie chart in Fig. 4b. It shows that 74.9% of the computational budget is consumed by the local search, 18.9% by repairing and only 6.2% by destroying. The ratio between destroying and repairing is given by the higher complexity of repair heuristics, which need to select both an intervention to schedule and its start time, whereas destroy heuristics select only an intervention to remove. The time spent by the local search is controlled by parameters `ONE_SHIFT_LIMIT`



(a) 5 min (b) 15-minute (c) 45 min (d) 90-minute

Fig. 5 Selection weights of destroy heuristics in time

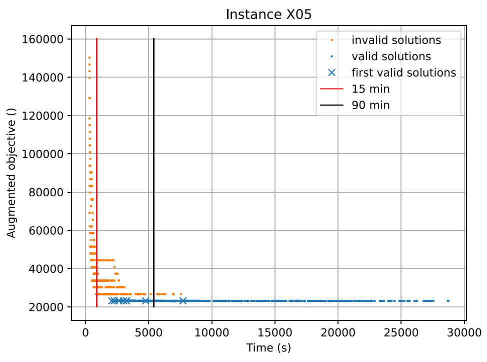
Table 17 Very long runs

Instances		ALNS - final				
Name	BKS	Objective values		Gaps (%)		Valid
		Min	Mean ± stdev	Min	Mean	(%)
X03	8102.59	8205.44	8211.85 ± 4.83	1.27	1.35	100
X05	22,837.42	23,064.99	23,082.82 ± 15.92	1.00	1.07	100
X15	45,422.29	45,916.02	45,992.48 ± 61.12	1.09	1.26	100
Average				1.12	1.23	100

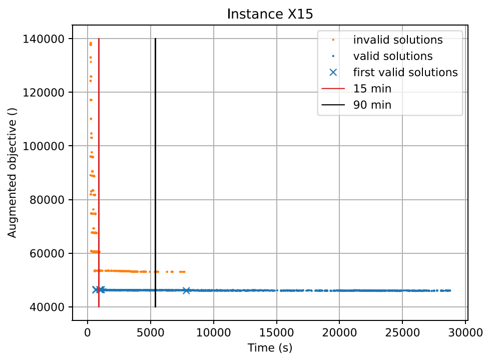
and `TWO_SHIFT_LIMIT`, which were subject to tuning. Thus, the presented ratio is presumably close to an optimal setup.

The fourth experiment attempts to provide insight into the ALNS learning mechanism, which adjusts the selection weights of individual heuristics based on their previous performance. The normalized selection weights of destroy heuristics in different time steps, averaged across multiple runs on a single instance X15, are visualized by pie charts in Fig. 5. It can be seen that the selection weights do not change dramatically and generally oscillate between $10 - 20\%$. The selection weights of individual heuristics do not copy their relative efficiency, shown in Fig. 2b, as their benefit changes over time. The intensity of the learning mechanism is controlled by the parameter $\lambda \in [0, 1]$, where 0 corresponds to no adjusting and 1 to no memory. It is tuned to $\lambda = 0.791$; thus, the weight adjusting is rather aggressive and implements short-term memory.

The final fifth experiment addresses those instances for which a valid solution was not always found even in the long 90-minute runs: X03, X05, and X15. Each of these instances was solved 8 times with a random seed, given a time budget of 8 h. The results of this experiment are provided in Table 17. The solver achieved a 100% success rate in this experiment, thus indicating that it is capable of consistently finding valid solutions for all competition datasets, although not always within the competition-defined time budget. This is illustrated in Fig. 6, which shows the convergence of the solver for instances X05 and X15 over multiple runs. The values of the augmented objective after each improving ALNS iteration are plotted against time. Invalid solutions, valid solutions, and first valid solutions in a given run are distinguished. In both instances, the solver is shown to sometimes work with an invalid solution even after the 90-



(a)



(b)

Fig. 6 ALNS convergence

minute time limit. The augmented objective function does not converge smoothly when improving invalid solutions, as it contains a penalty for unmet exclusivity constraints, which changes in discrete steps.

6 Conclusions

An adaptation of the ALNS metaheuristic for a variant of the transmission maintenance scheduling (TMS) problem assigned within the ROADEF Challenge 2020 is described in this paper. Although ALNS is not a commonly applied metaheuristic in TMS problems, the proposed method performed well in the competition of 74 teams, as it finished 6th in the semifinal phase (1st in the Junior category) and 8-9th in the final phase (2nd–3rd in the Junior category). In terms of solution quality, the method consistently finds solutions within the 2% gap on the most difficult competition dataset X and within 1% of the remaining datasets A, B, and C. The main contribution of the approach lies in proposing a large number of destroy and repair heuristics that exploit individual properties and constraints of the problem. These heuristics could easily be reapplied to other scheduling problems with a similar set of properties.

The paper also provides an experimental analysis of the proposed approach. Although the method is tuned, only 10% of the ALNS iterations result in an improved solution. The proposed method employs a penalization mechanism to satisfy multiple hard constraints of the problem. The method is shown to produce valid solutions with a high success rate (above 89% in 90-min runs). The approach could be improved by incorporating a complete procedure to ensure that a valid solution is returned every time, either custom or based on an exact MIP solver. The standard ALNS metaheuristic is extended by a local search, which is shown to be essential for its performance but also to be responsible for consuming most of the computational budget. The convergence rate of the proposed method could be further improved by speeding up the local search, for example, by restricting its operators only to neighborhoods consisting of valid solutions.

Acknowledgements The work on this paper was supported by the Czech Science Foundation grant 23 - 05104S. The work of David Woller was also supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/122/OHK3/2T/13. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

Abirami, M., Ganesan, S., Subramanian, S., Anandhakumar, R.: Source and transmission line maintenance outage scheduling in a power system using teaching learning based optimization algorithm. Appl. Soft Comput. **21**, 72–83 (2014). https://doi.org/10.1016/j.asoc.2014.03.015

Artigues, C., Bourreau, E., Jost, V., Kedad-Sidhoum, S., Ramond, F.: Trains do not vanish: the ROADEF/EURO challenge 2014. Ann. Oper. Res. **271**(2), 1091–1105 (2018). https://doi.org/10.1007/s10479-018-3066-x

Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Computation, 1st edn. IOP Publishing Ltd. (1997)

Ben Hamida, S., Schoenauer, M.: An adaptive algorithm for constrained optimization problems. In: International Conference on Parallel Problem Solving from Nature, pp. 529–538 (2000). Springer

Borůvka, O.: O jistém problému minimálním. Práce Moravské pírodovědecké společnosti II **I**(3), 37–58 (1926)

Burke, E.K., Smith, A.J.: Hybrid evolutionary techniques for the maintenance scheduling problem. IEEE Trans. Power Syst. **15**(1), 122–128 (2000). https://doi.org/10.1109/59.852110

Da Silva, E.L., Schilling, M.T., Rafael, M.C.: Generation maintenance scheduling considering transmission constraints. IEEE Trans. Power Syst. **15**(2), 838–843 (2000). https://doi.org/10.1109/59.867182

Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R.: In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Variable Neighborhood Descent, pp. 341–367. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_9

El-Sharkh, M.Y.: Clonal selection algorithm for power generators maintenance scheduling. Int. J. Electr. Power Energy Syst. **57**, 73–78 (2014). https://doi.org/10.1016/j.ijepes.2013.11.051

Feng, C., Wang, X., Li, F.: Optimal maintenance scheduling of power producers considering unexpected unit failure. IET Gener. Transm. Distrib. **3**(5), 460–471 (2009). https://doi.org/10.1049/iet-gtd.2008.0427

Froger, A., Gendreau, M., Mendoza, J.E., Pinson, É., Rousseau, L.M.: Maintenance scheduling in the electricity industry: a literature review. Eur. J. Oper. Res. **251**(3), 695–706 (2016). https://doi.org/10.1016/j.ejor.2015.08.045

Geetha, T., Swarup, K.S.: Coordinated preventive maintenance scheduling of GENCO and TRANSCO in restructured power systems. Int. J. Electr. Power Energy Syst. **31**(10), 626–638 (2009). https://doi.org/10.1016/j.ijepes.2009.06.006

He, L., Liu, X., Laporte, G., Chen, Y., Chen, Y.: An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. Comput. Oper. Res. **100**, 12–25 (2018). https://doi.org/10.1016/j.cor.2018.06.020

Huang, S.J.: Generator maintenance scheduling: a fuzzy system approach with genetic enhancement. Electr. Power Syst. Res. **41**(3), 233–239 (1997). https://doi.org/10.1016/s0378-7796(96)01194-7

Joines, J.A., Houck, C.R., et al.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Ga's. In: International Conference on Evolutionary Computation, pp. 579–584 (1994)

Khalid, W., Soleymani, I., Mortensen, N.H., Sigsgaard, K.V.: Ai-based maintenance scheduling for offshore oil and gas platforms. In: 2021 Annual Reliability and Maintainability Symposium (RAMS), pp. 1–6 (2021). https://doi.org/10.1109/RAMS48097.2021.9605794

Kovacs, A.A., Parragh, S.N., Doerner, K.F., Hartl, R.F.: Adaptive large neighborhood search for service technician routing and scheduling problems. J. Sched. **15**(5), 579–600 (2012). https://doi.org/10.1007/s10951-011-0246-9

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002

Lu, C., Wang, J., Sun, P.: Short-term transmission maintenance scheduling based on the benders decomposition. In: APPEEC 2012 (2012). https://doi.org/10.1109/APPEEC.2012.6307696

Lv, C., Wang, J., You, S., Zhang, Z.: Short-term transmission maintenance scheduling based on the Benders decomposition. Int. Trans. Electric. Energy Syst. **25**(4), 697–712 (2015). https://doi.org/10.1002/etep.1867

Michel, L., Hentenryck, P.V.: In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Constraint-Based Local Search, pp. 223–260. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_7

Mollahassani-Pour, M., Abdollahi, A., Rashidinejad, M.: Application of a novel cost reduction index to preventive maintenance scheduling. Int. J. Electr. Power Energy Syst. **56**, 235–240 (2014). https://doi.org/10.1016/j.ijepes.2013.11.026

Moro, L.M., Ramos, A.: Goal programming approach to maintenance scheduling of generating units in large scale power systems. IEEE Trans. Power Syst. **14**(3), 1021–1028 (1999). https://doi.org/10.1109/59.780915

Pandžić, H., Conejo, A.J., Kuzle, I., Caro, E.: Yearly maintenance scheduling of transmission lines within a market environment. IEEE Trans. Power Syst. **27**(1), 407–415 (2012). https://doi.org/10.1109/TPWRS.2011.2159743

Pisinger, D., Ropke, S.: Large neighborhood search. In: Handbook of Metaeuristics, pp. 399–419. Springer, Cham (2010). https://doi.org/10.1007/978-1-4419-1665-5_13

Reihani, E., Sarikhani, A., Davodi, M., Davodi, M.: Reliability based generator maintenance scheduling using hybrid evolutionary approach. Int. J. Electr. Power Energy Syst. **42**(1), 434–439 (2012). https://doi.org/10.1016/j.ijepes.2012.04.018

Rifai, A.P., Nguyen, H.-T., Dawal, S.Z.M.: Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl. Soft Comput. **40**, 42–57 (2016). https://doi.org/10.1016/j.asoc.2015.11.034

ROADEF: Challenge ROADEF/EURO 2020: Maintenance planning Problem (2020). https://www.roadef.org/challenge/2020/en/

Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006). https://doi.org/10.1287/trsc.1050.0135

Ruiz, M., Tournebise, P., Panciatici, P.: ROADEF Challenge RTE: Grid Operation-Based Outage Maintenance Planning. Technical report, RTE (2020)

Salinas San Martin, L., Yang, J., Liu, Y.: Hybrid NSGA III/dual simplex approach to generation and transmission maintenance scheduling. Int. J. Electric. Power Energy Syst. **135**, 10 (2022). https://doi.org/10.1016/j.ijepes.2021.107498

Saraiva, J.T., Pereira, M.L., Mendes, V.T., Sousa, J.C.: A simulated annealing based approach to solve the generator maintenance scheduling problem. Electr. Power Syst. Res. **81**(7), 1283–1291 (2011). https://doi.org/10.1016/j.epsr.2011.01.013

Schlünz, E.B., Van Vuuren, J.H.: An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. Int. J. Electr. Power Energy Syst. **53**(1), 166–174 (2013). https://doi.org/10.1016/j.ijepes.2013.04.010

Shahidehpour, M., Marwali, M.: Maintenance Scheduling in Restructured Power Systems. Springer, New York (2000). https://doi.org/10.1007/978-1-4615-4473-9

Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: International Conference on Principles and Practice of Constraint Programming, pp. 417–431. Springer (1998)

Smith, S.L., Imeson, F.: GLNS: an effective large neighborhood search heuristic for the generalized traveling salesman problem. Comput. Oper. Res. (2017). https://doi.org/10.1016/j.cor.2017.05.010

Suresh, K., Kumarappan, N.: Hybrid improved binary particle swarm optimization approach for generation maintenance scheduling problem. Swarm Evol. Comput. **9**, 69–89 (2013). https://doi.org/10.1016/j.swevo.2012.11.003

Than Kyi, M., Maw, S.S., Naing, L.L.: Mathematical estimation for maximum flow in electricity distribution network by Ford–Fulkerson iteration algorithm. Int. J. Sci. Res. **9**(8), 9229 (2019). https://doi.org/10.29322/ijsrp.9.08.2019.p9229

Volkanovski, A., Mavko, B., Boševski, T., Čauševski, A., Čepin, M.: Genetic algorithm optimisation of the maintenance scheduling of generating units in a power system. Reliab. Eng. Syst. Saf. **93**(6), 779–789 (2008). https://doi.org/10.1016/j.ress.2007.03.027

Wang, Y., Zhong, H., Xia, Q., Kirschen, D.S., Kang, C.: An approach for integrated generation and transmission maintenance scheduling considering N-1 contingencies. IEEE Trans. Power Syst. **31**(3), 2225–2233 (2016). https://doi.org/10.1109/TPWRS.2015.2453115

Wen, M., Linde, E., Ropke, S., Mirchandani, P., Larsen, A.: An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. Comput. Oper. Res. **76**, 73–83 (2016). https://doi.org/10.1016/j.cor.2016.06.013

Woller, D., Kulich, M.: The ALNS metaheuristic for the maintenance scheduling problem. In: Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2021, pp. 156–164 (2021). https://doi.org/10.5220/0010552101560164

Chapter 6

Metaheuristic solver for problems with permutative representation

The fourth core publication is called Metaheuristic solver for problems with permutative representation [c4]. The initial concept was developed in the diploma thesis [s12] and further expanded in the bachelor thesis [s13], both of which were supervised by the author. The developed solver was also presented at the 23rd Conference of the International Federation of Operational Research Societies (IFORS 2023) [96].

[c4] **Woller, D.**, Hrazdíra, J., Kulich, M., "Metaheuristic Solver for Problems with Permutative Representation", in *Intelligent Computing & Optimization*, Springer International Publishing, 2023, pp. 42–54, ISBN: 978-3-031-19958-5. DOI: 10.1007/978-3-031-19958-5_5, **50% contribution, citations: 0 in Web of Science, 0 in Scopus, 1 in Google Scholar.**

This stream of work chronologically follows the core publications already presented and is directly motivated by them. Instead of developing a specialized solver for a single novel combinatorial optimization problem, we propose a simple formalism that can be used to formulate a large class of problems with permutative representation. That is, their solution can be fully encoded as an ordered sequence of nodes from a predefined set. The nodes may appear repeatedly in the solution sequence as long as the frequency of their occurrence stays within the predefined bounds. The cost function may be arbitrary, and all constraints need to be described in the form of penalty functions.

We introduce a generic metaheuristic solver capable of solving any problem defined in the proposed formalism. The solver offers several local search-based metaheuristics and a bank of 11 local search operators, 3 construction procedures, and 6 perturbation operators. It is designed to be highly modular - given a specific problem to solve, the solver can be automatically configured, both in terms of numerical parameters and selection of individual components. Individual penalty functions are aggregated and treated by a static penalization mechanism.

The experimental results present a comparison with the commercial Gurobi Optimizer [23], which is a state of the art generic IP solver. In a fixed-time experiment, the proposed generic metaheuristic solver outperformed the Gurobi Optimizer both in terms of solution quality and scalability. The experiment was carried out for several textbook optimization problems, specifically the Capacitated Vehicle Routing Problem (CVRP), Non-Permutation Flowshop Scheduling Problem (NPFS), and Quadratic Assignment Problem (QAP), all of which are $\mathcal{NP}$-hard, but have only a few constraints. The solver was deployed to problems richer in constraints in the follow-up work [s13], [c5]. In [c5], it was deployed to several newly proposed variants of the TSP and HCP and solution feasibility was guaranteed by adding a custom exact construction procedure. In [s13], it was used to solve the EVRP and TMS competition problems addressed in previous core publications and compared with problem-specific metaheuristic algorithms. The constraint satisfaction rate was improved by adding the Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) to the generic solver.

Metaheuristic solver for problems with permutative representation

David Woller [1,2,(✉)], Jan Hrazdíra[1], and Miroslav Kulich [1]

[1] Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University in Prague
Jugoslávských partyzánů 1580/3, Praha 6, 160 00, Czech Republic
[2] Department of Cybernetics, Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13, Praha 2, 121 35, Czech Republic
`wolledav@cvut.cz`

Abstract. Today, a large proportion of combinatorial optimization problems can be efficiently formulated as a mixed-integer program and solved with an exact solver. However, exact solvers do not scale well and thus custom metaheuristic algorithms are being designed to provide better scalability at the cost of no optimality guarantees and time-consuming development. This paper proposes a novel formalism for a large class of problems with permutative representation, together with a metaheuristic solver addressing these problems. This approach combines the advantages of both exact and metaheuristic solvers: straightforward problem formulation, scalability, low design time, and ability to find high quality solutions. Three different problems are formulated in the proposed formalism and solved with the proposed solver. The solver is benchmarked against the Gurobi Optimizer and significantly outperforms it in experiments with a fixed computational budget.

Keywords: Metaheuristics, Iterated Local Search, Variable Neighborhood Search, Permutative Representation

1 Introduction

Despite its relatively short history, combinatorial optimization is now a mature field with robust theoretical foundations and a broad portfolio of powerful methods. New and challenging applications are constantly emerging in a wide range of diverse fields, such as artificial intelligence, machine learning, supply chain management, or financial engineering. However, the underlying optimization problems are often intractable. Therefore, selecting the most suitable approach typically requires a trade-off between some of several criteria: optimality or approximation guarantee, design time, runtime, scalability, and versatility.

When the solution optimality is required, the available design time is limited, and the instances are of moderate size, the most suitable choice is to use Integer Programming (IP). Today, multiple solvers are available, such as Gurobi, CPLEX, or Xpress, that provide highly efficient implementations of state of the art exact algorithms. The only user requirement is to formalize the problem and create a sensible model, which makes these solvers very attractive and widely used in practice. However, an explicit IP formulation of a high-dimensional problem may result in an immensely large model, which can be difficult to work with because of memory limitations. Furthermore, the computational complexity of an exact solver is inherently exponential when solving an $\mathcal{NP}$-hard problem, making it intractable. For these reasons, scalability remains a major limitation of problem-nonspecific IP solvers.

Computationally challenging applications are often tackled by metaheuristics, high-level algorithmic frameworks that can be adapted to problem-specific algorithms. Metaheuristic algorithms typically do not provide any guarantees about the quality of the solution, but are frequently used in applications where no other approach is computationally feasible. Most applications of metaheuristics require considerable design time and the implementations are not versatile. The reason for this is that essential components, such as repair heuristics, destroy heuristics, or local search operators, need to be tailored to a particular problem.

This paper presents a generic metaheuristic solver that is capable of solving a set of problems that share the same solution representation. The main goal is to combine the scalability of custom metaheuristic algorithms with the versatility of modeling paradigms such as IP. The representation of interest encodes a solution as an ordered sequence of nodes which can have arbitrary length and frequency of individual nodes. Any additional constraints and specifics are incorporated into an aggregated fitness function through penalties. This representation allows for the addressing of a broad portfolio of routing, scheduling, or sequencing problems. The proposed solver is benchmarked against the Gurobi optimizer on three $\mathcal{NP}$-hard problems. The main contributions of this paper are:

– proposing a unifying formalism for a wide class of permutative problems,
– proposing a generic solver working with the proposed formalism,
– providing a fair and extensive benchmark against the Gurobi optimizer.

The remainder of this paper is structured as follows. Section 2 discusses related works. Section 3.1 provides the definition of the proposed formalism, as well as the definitions of three classical problems in this formalism. The proposed solver is described in Section 3.2. The experimental results are presented in Section 4 and the conclusions in Section 5.

2 Related Works

First, several metaheuristic solvers capable of addressing a wider portfolio of related problems exist. For example, the LKH3 solver [9], is able to solve a large number of variants of Vehicle Routing Problem, Sequential Ordering Problem, Travelling Repairman Problem, Travelling Salesman Problem, and others by transforming these problems into a standard TSP. Another example is the Unified Hybrid Genetic Search for Multiattribute Vehicle Routing Problems [21], addressing a broad range of VRP problems. Other solvers are designed as black-box solvers for any problem with a solution representable by a fixed-length permutation. These are, for example, the genetic algorithm (GA) combined with Branch & Bound technique proposed in [14], or the Bayesian optimization approach in [5]. However, the solution representation used in this paper is more general as it allows the solution sequence to have arbitrary length and node frequency.

Second, various metaheuristic frameworks exist. The purpose of these frameworks is to provide problem-independent building blocks that can be used to efficiently assemble a specialized solver. The common drawback is that the use of a custom solution representation typically requires the implementation of custom operators. According to a recent survey [17], most widely used frameworks are the Evolutionary Computation Research System [18] and the ParadiseEO framework [6]. Both implement a large number of mostly evolutionary algorithms and support basic representations such as integer vectors, binary vectors, or fixed-length permutations. Some frameworks are more specialized. For example, the MOEA framework [8] focuses on multi-objective evolutionary algorithms, and the JAMES framework [4] specializes in local search metaheuristics.

Third, the proposed solver contains a plethora of alternative heuristics, operators, and parameters that should be chosen appropriately for the problem at hand. Therefore, the automated design and configuration of heuristic algorithms is also a relevant domain. For this purpose, an external heuristic tuning algorithm is often used [19]. Some successful approaches are: the ParamILS tool [2] based on the Iterated Local Search metaheuristic [13], the Sequential Model-Based Algorithm Configuration tool [10], or the Iterated Racing tool [12], which was actually used for configuration of the proposed solver.

3 Methodology

This section details the main contributions. The common formalism used by the proposed solver is introduced in Section 3.1 and its usage is demonstrated on three problems: Capacitated Vehicle Routing Problem (CVRP), Non-Permutation Flowshop Scheduling Problem (NPFS) and Quadratic Assignment Problem (QAP). The proposed metaheuristic solver is described in Section 3.2.

3.1 Problem definitions

We newly propose the following generic problem definition that serves as an interface between the proposed solver and the specific problem to be solved. It defines only a few properties common to a large number of problems with permutative representation, which are known to the solver. Problem-specific properties are intended to be translated into penalty functions, which are then aggregated with the problem fitness function.

Generic problem definition is given as follows. Let A be a set of n unique nodes (Eq. 1), which can be present in a solution vector X of length m (Eq. 5). A node $a_i \in A$ can be present in X multiple times and the number of its occurrences is given by frequency f_i (Eq. 6-7). Here, the logical expression $[\![.]\!]$ evaluates to 1 if true, 0 otherwise. Each element $a_i \in A$ has defined lower and upper bounds l_i, u_i in the vectors L, U (Eq. 2-3), which limit the frequency of a_i (Eq. 8). The goal is to minimize the augmented fitness function $g(X)$ (Eq. 4).

$$\begin{array}{llll}
\text{Given set of nodes} & A = \{a_1, a_2, ..., a_n\}, & A \subset \mathbb{N} & (1) \\
\text{and vectors of bounds} & L = [l_1, l_2, ..., l_n], & L \in \mathbb{N}_0^n & (2) \\
 & U = [u_1, u_2, ..., u_n], & U \in \mathbb{N}_0^n & (3) \\
\text{minimize fitness function} & g(X) : A^m \to \mathbb{R} & & (4) \\
\text{where} & X = [x_1, x_2, ..., x_m], & X \in A^m & (5) \\
 & F = [f_1, f_2, ..., f_n], & F \in \mathbb{N}_0^n & (6) \\
 & \forall a_i \in A : f_i = \sum_{j=1}^{m} [\![x_j = a_i]\!] & & (7) \\
 & \forall a_i \in A : l_i \le f_i \le u_i & & (8)
\end{array}$$

Note that X can have variable length if $L \neq U$. Also, the fitness function $g(X)$ can be completely arbitrary. In the following applications, it is expressed as $g(X) = \hat{g}(X) + p(X)$, where $\hat{g}(X)$ is the actual fitness function of the problem currently solved and $p(X)$ is a penalty function enforcing additional problem-specific properties in X.

Capacitated Vehicle Routing Problem (CVRP) is a classic routing problem. The CVRP's objective is to serve a set of customers with a fleet of k vehicles, while satisfying the demands of customers on cargo and respecting the limited load capacity of the vehicles. An equivalent interpretation used here is that a single vehicle is required to carry out at most k trips, while reloading the cargo at the central depot. The CVRP can be defined in the proposed formalism as follows.

$$D = a_1 \tag{9}$$
$$L = [2, 1, ..., 1] \tag{10}$$
$$U = [k + 1, 1, ..., 1] \tag{11}$$
$$\hat{g}(X) = \sum_{i=1}^{m-1} cost(x_i, x_{i+1}) \tag{12}$$
$$p(X) = M([\![x_1 \neq D]\!] + [\![x_m \neq D]\!]) \tag{13}$$
$$+ M \sum_{i=1}^{m} \max(0, dem(x_i)[\![load(x_i) < dem(x_i)]\!]) \tag{14}$$
$$\text{where } load(x_i) = \begin{cases} 0, & \text{if } i = 1 \\ Q, & \text{if } x_{i-1} = D \\ load(x_{i-1}) - dem(x_{i-1}), & \text{otherwise} \end{cases} \tag{15}$$

The set of nodes A contains $n - 1$ customers to visit and a single depot D, which is fixed at a_1 (Eq. 9). The solution vector X contains individual trips separated by depot visits: $X = [D, x_2, ..., D, ..., x_{m-1}, D]$. The bounds L, U ensure that each customer is visited exactly once and the depot is visited at most $k+1$ times (Eqs.10-11). The CVRP fitness function $\hat{g}(X)$ corresponds to the total distance traveled by the vehicle (Eq. 12), where $cost(x_i, x_{i+1})$ is the distance between the nodes x_i and x_{i+1}. Two additional penalties must be added by the penalty function $p(X)$. The vehicle must start and end at the depot (Eq. 13) and the vehicle must have sufficient cargo load to fully satisfy each visited customer (Eq. 14). Here, $load(x_i)$ is the vehicle cargo load before entering node x_i and $dem(x_i)$ is the demand of node x_i. The vehicle is empty before entering the first node in X and loaded to its maximum load capacity Q in the depot (Eq. 15). Finally, M is a large constant.

Non-Permutation Flowshop Scheduling Problem (NPFS) is a variant of a more common Flowshop Scheduling Problem (FSP). In NPFS, n jobs from A have to be scheduled on k machines. The solution vector $X = [x_1, x_2, ..., x_m] = [\pi_1, \pi_2, ..., \pi_k]$ consists of k permutations π_j of length n, where each of these permutations determines the order of processing of the jobs on the machine $j \in \{1, 2, ..., k\}$. Therefore, $m = nk$. The bounds L, U reflect that each job has to be processed on each machine exactly once (Eqs. 16-17). The order in which individual machines are visited by each job is fixed and is given by the index k of the machine. Unlike in the FSP, the processing order within individual machines can differ in NPFS; therefore, generally $\pi_i \neq \pi_j$. In the following equations, $start(a_i, \pi_j)$ is the current start time of the processing of the job a_i in permutation π_j, $proc(a_i, j)$ is the processing time of the job a_i in machine j, and $f_{a_i}^{\pi_j}$ is the frequency of the node a_i in permutation π_j.

$$L = [k, k, ..., k] \tag{16}$$
$$U = [k, k, ..., k] \tag{17}$$
$$\hat{g}(X) = \max_{a_i \in A}(start(a_i, \pi_k) + proc(a_i, k)) \tag{18}$$
$$p(X) = M \sum_{i=1}^{n} \sum_{j=1}^{k} |1 - f_{a_i}^{\pi_j}| \tag{19}$$
$$+ M \sum_{i=1}^{n} \sum_{j=1}^{k-1} [\![start(a_i, \pi_{j+1}) \geq start(a_i, \pi_j) + proc(a_i, j)]\!] \tag{20}$$

The goal is to minimize the total makespan, i.e., the last job on the k-th machine should finish as early as possible (Eq. 18). Two penalties must be defined in $p(X)$. First, each job has to be scheduled exactly once on each machine (Eq. 19). Second, a job a_i can start to be processed on machine $j+1$ only after it finishes on machine j (Eq. 20).

Quadratic Assignment Problem (QAP) is a generalization of the following facility location problem: n distinct facilities from $\{1, 2, ..., n\}$ must be assigned to n distinct locations from A. Flow $f(i, j)$ is given for all pairs of facilities i, j and distance $d(a_k, a_l)$ is given for all pairs of locations a_k, a_l. The solution vector X contains a single permutation, where x_i determines the location assigned to the facility i. The goal is to minimize the sum of all flows weighted by the corresponding distances according to the assignment X (Eq. 23).

$$L = [1, 1, ..., 1] \tag{21}$$
$$U = [1, 1, ..., 1] \tag{22}$$
$$\hat{g}(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} flow(i, j) dist(x_i, x_j) \tag{23}$$

QAP formulation in the proposed formalism is exceptionally simple and does not require any penalties. By contrast, formulating the problem as MILP is complicated by the non-linearity of the problem, and a linearization technique needs to be employed.

3.2 Proposed solver

The proposed metaheuristic solver addresses the generic problem defined in Section 3.1. The user is required to specify the set of nodes A, vectors L, B containing their bounds and the augmented fitness function $g(X) = \hat{g}(X) + p(X)$. The solver minimizes $g(X)$ over all possible sequences $X \in A^m$ and guarantees that the condition on bounds defined in Eq. 8 is satisfied. All remaining

Double Bridge(X, k) generates k distinct random indices between 1 and m. The solution X is then divided into $k+1$ segments determined by these indices, and all of these segments are reversed.
Random Double Bridge(X, k) operates similarly to Double Bridge. However, each of the $k+1$ segments of X is reversed with a probability of 0.5.
Reinsert(X, k) selects uniformly randomly k distinct nodes from A. All occurrences of these nodes are removed from X and reinserted in randomly selected positions.
Random Swap(X, k) randomly selects two nodes in X and swaps their positions. This operation is performed k times.
Random Move(X, k) randomly selects a node in X and moves it to a random location. This operation is performed k times.
Random Move All(X, k) randomly selects a node a from A. All occurrences of a are randomly moved in X up to a maximal distance k from their original locations. This operation is performed k times.

Table 1. Perturbations

problem-specific constraints are hidden from the solver and must be enforced by the penalty function $p(X)$.

The solver contains four sets of alternative components: metaheuristics (MH), local search heuristics (LS), perturbations (P) and construction procedures (C). When solving a specific problem, one component from each of these categories must be set. The solver also contains a list of local search operators (O), at least one of which must always be used. The best configuration for a given problem is automatically identified using the Iterated Racing tool (irace) [12]. The remainder of this section provides a brief description of individual components, some of which were adapted from the literature and are described only briefly.

Metaheuristics are top-level procedures that control the entire optimization process. Two metaheuristics are currently implemented in the proposed solver: Iterated Local Search **ILS**(C, LS, P, O, k) [13] and Variable Neighborhood Search **VNS**(C, LS, P, O, k_{min}, k_{max}) [15]. Both metaheuristics construct an initial solution and then alternate local search and perturbation, until the timeout is reached. The parameter k is a numerical parameter of the perturbation, which is fixed in the ILS and variable in the VNS.

Local search heuristics control the application of individual local search operators in O to a current solution X, with the aim of reaching a new local optimum. Five variants of the Variable Neighborhood Descent (VND) heuristic adapted from [7] are implemented in the solver: Basic **BVND**(X, O), Pipe **PVND**(X, O), Cyclic **CVND**(X, O), Random **RVND**(X, O) and Random Pipe **RPVND**(X, O). All VND variants terminate and return a possibly improved solution X, when none of the operators in O succeeds in improving X. The only difference is the rule for iterating through O.

Perturbations are intended to randomly modify a current solution X in order to escape a local optimum. Six perturbations are implemented in the proposed solver and described in Table 1. All perturbations take a current solution X and a numerical parameter k as input and return a modified solution regardless of its fitness. Generally, the value of k determines the intensity of a perturbation.

Construction procedures are used to create an initial solution. Three constructions described in Table 2 are implemented in the proposed solver. All constructions return a solution X valid w.r.t. Eq. 8.

Local search operators define different neighborhoods of a solution X. When applying a local search operator, the entire neighborhood is searched, and the most improving modification of X w.r.t. $g(X)$ is applied. All operators perform improving moves only and are not allowed to violate the constraint given by Eq. 8. Some operators take additional parameters which are fixed in the solver. Thus, multiple variants of some operators are instantiated. The ranges of these parameters were set empirically and are given in the operator description below. The 11 operators described in Table 3 are implemented in the solver.

For some of these operators, the corresponding neighborhood is a subspace of another operator's neighborhood (e.g. Exchange and Reverse Exchange, Centered Exchange and Two-opt). The reason for this is that smaller neighborhoods can be searched faster and still lead to new local optima. On the other hand, a larger neighborhood may be beneficial when the search stagnates.

4 Results

The proposed solver described in Section 3.2 was implemented in C++. Both the implementation and the solution files are publicly available at [22]. The solver was benchmarked against the exact Gurobi Optimizer. The solver was parallelized using the OpenMP library, and its configuration for each problem was tuned by the irace package with a budget of 4000 experiments per problem. Both solvers were allowed to use eight threads. All experiments were carried out on dedicated machines with Ubuntu 18.04 OS, Intel Core i7-7700 CPU, 32 GB RAM memory, and 34 GB swap memory.

For each problem, a data set of 10 commonly used test instances of variable size and difficulty was selected. A different dataset of 20 training instances was used for automatic configuration of the proposed solver. The tuned configurations for each problem are shown in Table 4. Both solvers were given the same computing time, which was set empirically according to the size of the instance. Each instance was solved once by the Gurobi and fifty times by the proposed solver, as it is stochastic.

The following data are presented for each instance: the best known solution from the literature (BKS), the computing time budget (*time*), the lower bound and the gap value of the solution fitness found by the Gurobi (*LB*, *gap*), the

Greedy() repeatedly applies the local search operator Insert(X). The construction terminates, when $\forall a_i \in A : f_i = l_i$.
Random() repeatedly inserts nodes into random locations in X, until $\forall a_i \in A : f_i = l_i$.
Random-replicate() first generates a random permutation of all nodes in A. A copy of this permutation is repeatedly appended to itself, until $\forall a_i \in A : f_i \geq l_i$. If $f_i = u_i$, a_i is removed from the appended copy.

Table 2. Construction procedures

Insert(X) attempts to insert a node $a_i \in A$ to a position $j \in \{1, 2, ..., m\}$. The most improving combination of i, j w.r.t. $g(X)$ is used. As the operator is used in greedy construction, it internally extends the penalty function as $p(X) = p(X) + M^2 \sum_{i=1}^{n} \max(l_i - f_i, 0)$.
Remove(X) attempts to remove a node $x_i \in X$.
Two-opt(X) attempts to reverse a substring of X given by indices $i, j \in \{1, 2, ..., m\}$.
Exchange Nodes(X) attempts to exchange all occurrences of nodes $a_i, a_j \in A$ in X.
Exchange First Nodes(X) attempts to exchange first k occurrences of nodes $a_i, a_j \in A$ in X, where $k \in \{1, 2, ..., \max(f_i, f_j)\}$.
Exchange(X, p, q) attempts to exchange substrings of fixed lengths p, q in X. Variants: $(p, q) \in \{(1,1), (1,2), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$.
Reverse Exchange(X, p, q) operates similarly to the Exchange operator, but it also attempts to reverse one or both substrings before exchanging.
Centered Exchange(X, p) reverts the substring of length $2p + 1$, starting at position $i \in \{0, m - p + 1\}$. Variants: $p \in \{1, 2, 3, 4, 5\}$.
Move(X, p) attempts to move a substring of X with fixed length p from a position $i \in \{1, 2, ..., m - p + 1\}$ to a position $j \in \{1, 2, ..., m\}$. Variants: $p \in \{1, 2, 3, 4, 5\}$.
Reverse Move(X, p) operates similarly to the Move operator, but it also attempts to reverse the substring. Variants: $p \in \{2, 3, 4, 5\}$.
Move all(X, p) attempts to move all occurrences of a node $a_i \in A$ in X. All occurrences are shifted by the same distance distance $p \in \{-p, ..., 0, ..., p\}$ from their original positions. Variants: $p \in \{1, 2, 3, 4, 10\}$.

Table 3. Local search operators

same values obtained in ten times the computing budget (LB_{10t}, gap_{10t}), the best gap, the mean gap, and the standard deviation achieved by the proposed solver ($gap^*, \overline{gap}$, $stdev$). The values of gap are calculated as $gap(\%) = 100(\frac{\hat{g}(X)}{BKS} - 1)$. BKSs that are known to be optimal are written in bold.

Problem	Configuration
CVRP	MH: VNS($k_{min} = 3$, $k_{max} = 8$); C: Random; LS: RVND; P: Random Double Bridge; O: Insert, Remove, Two-opt, Exchange Nodes, Exchange($(p, q) \in \{(1,1), (1,2), (2,3), (2,4), (3,3), (3,4)\}$), Reverse Exchange($(p, q) \in \{(1,2)\}$), Centered Exchange($p \in \{1, 2, 3, 4\}$), Move($p \in \{5\}$), Reverse Move($p \in \{2, 3, 5\}$), Move All($p \in \{2, 3, 4\}$)
NPFS	MH: ILS($k = 1$); C: Random-replicate; LS: BVND; P: Random Swap; O: Exchange Nodes, Exchange($(p, q) \in \{(1,1), (2,2), (2,4), (3,4), (4,4)\}$). Reverse Exchange($(p, q) \in \{(2,3)\}$), Centered Exchange($p \in \{2, 4\}$), Move($p \in \{1, 2, 5\}$), Reverse Move($p \in \{4\}$), Move All($p \in \{1, 3, 4\}$)
QAP	MH: ILS($k = 6$); C: Random; LS: PVND; P: Random Move; O: Two-opt, Exchange Nodes; Exchange($(p, q) \in \{(1,1), (2,2), (2,3), (2,4), (3,4)\}$), Reverse Exchange($(p, q) \in \{(2,2), (2,4), (4,4)\}$), Centered Exchange($p \in \{1, 2, 3, 5\}$), Move($p \in \{3\}$), Reverse Move($p \in \{3\}$), Move All($p \in \{1, 2, 3\}$).

Table 4. Tuned configurations

CVRP results are presented in Table 5. The problem was solved in Gurobi using the Miller-Tucker-Zemlin ILP formulation [16]. Testing instances were selected from CVRPLIB [11]. The proposed solver was able to find the optimum for the smallest three instances and for most of the instances, both gap^* and $\overline{gap}$ are within 5%. The proposed solver found a valid solution every time. On the other hand, Gurobi generated a valid solution only for the three smallest instances and with significantly larger gap. For the remaining seven instances, Gurobi returned only a lower bound, which did not improve much even in the longer runs with ten times the computing time.

NPFS results are presented in Table 6. The ILP formulation was adapted from [1] and the testing instances were selected from the dataset [20]. The BKS

			Gurobi				Proposed solver		
Instance	$time[s]$	BKS	LB	LB_{10t}	gap	gap_{10t}	gap^*	$\overline{gap}$	$stdev$
P-n050-k10	600	**696**	567	579	2.73%	1.01%	**0.00%**	0.18%	1.07
A-n65-k09	600	**1174**	822	831	29.05%	2.47%	**0.00%**	0.22%	2.03
P-n076-k05	600	**627**	567	571	13.88%	6.86%	**0.00%**	0.16%	1.50
X-n148-k46	1800	**43448**	23072	23705	-	-	0.12%	0.51%	89.94
X-n204-k19	1800	**19565**	14105	14118	-	-	0.57%	1.19%	60.48
X-n251-k28	1800	**38684**	20779	20779	-	-	1.04%	1.47%	95.73
X-n351-k40	3600	25896	14061	14061	-	-	3.84%	5.02%	174.04
X-n561-k42	3600	42717	25226	25768	-	-	2.58%	3.47%	200.30
X-n749-k98	3600	77269	30465	30805	-	-	4.93%	6.10%	388.35
X-n1001-k43	3600	72355	29862	30419	-	-	7.62%	8.90%	395.64

Table 5. CVRP results

values shown were generated for the Permutation Flowshop Problem (PFS), which requires the same job order on each machine. These are valid for the NPFS as well, but may be suboptimal. The proposed solver found a valid solution every time, and both gap^* and $\overline{gap}$ are within 7% for most instances. BKS was found twice, and in one case, a solution of better quality than PFS BKS was found. Gurobi found a valid solution for eight instances. For the remaining two, Gurobi failed to build the problem model due to excessive memory requirements. The gap and gap_{10t} values found by the Gurobi do not differ significantly from each other and are several times higher than those of the proposed solver.

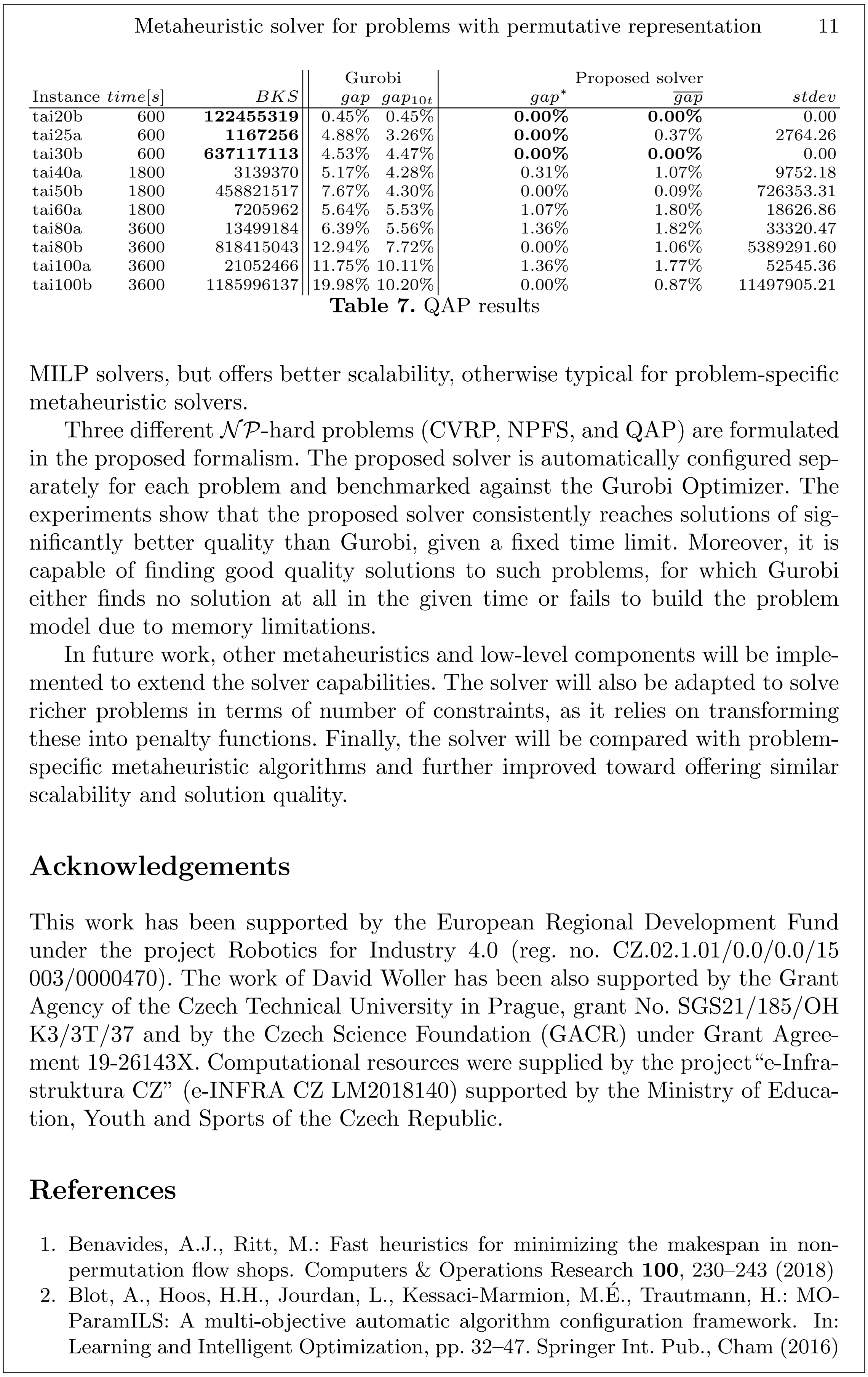
			Gurobi				Proposed solver		
Instance	$time[s]$	BKS	LB	LB_{10t}	gap	gap_{10t}	gap^*	$\overline{gap}$	$stdev$
VFR20_5_1	600	1192	810	814	6.29%	3.61%	0.00%	0.16%	1.15
VFR20_15_1	600	1936	1409	1437	7.95%	4.86%	-0.41%	0.12%	3.83
VFR40_5_1	600	2396	1452	1460	6.59%	6.01%	0.00%	0.00%	0.00
VFR100_20_1	1800	6121	3482	3518	28.41%	27.46%	4.20%	5.28%	29.11
VFR100_20_2	1800	6119	3590	3622	29.14%	27.46%	5.77%	6.81%	25.53
VFR200_20_1	1800	11181	3684	5806	22.81%	22.73%	4.52%	5.52%	62.15
VFR400_40_1	3600	23085	2482	11997	22.57%	22.57%	6.14%	7.40%	165.19
VFR400_60_1	3600	25214	-	-	-	-	5.93%	6.97%	144.22
VFR600_40_1	3600	33337	2555	2555	18.86%	18.86%	4.80%	5.88%	146.23
VFR600_60_1	3600	35450	-	-	-	-	5.83%	6.89%	172.28

Table 6. NPFS results

QAP results are presented in Table 7. The Xia Yuan linearization [23] was used in the MILP formulation and the testing instances were selected from QAPLIB [3]. For this problem, both Gurobi and the proposed solver found a valid solution every time. For this reason and because of space limitations, the values of LB and LB_{10t} are omitted. The proposed solver found the BKS for six instances. Gurobi performed significantly worse than the proposed solver in both standard and long runs, although the values of gap_{10t} improved noticeably compared to gap.

5 Conclusions

In this paper, a unifying formalism for combinatorial optimization problems with permutative representation is proposed, together with a metaheuristic solver capable of addressing this class of problems. The goal is to provide a versatile solver that requires the user only to model the problem, similarly to various



			Gurobi		Proposed solver		
Instance	$time[s]$	BKS	gap	gap_{10t}	gap^*	$\overline{gap}$	$stdev$
tai20b	600	**122455319**	0.45%	0.45%	**0.00%**	**0.00%**	0.00
tai25a	600	**1167256**	4.88%	3.26%	**0.00%**	0.37%	2764.26
tai30b	600	**637117113**	4.53%	4.47%	**0.00%**	**0.00%**	0.00
tai40a	1800	3139370	5.17%	4.28%	0.31%	1.07%	9752.18
tai50b	1800	458821517	7.67%	4.30%	0.00%	0.09%	726353.31
tai60a	1800	7205962	5.64%	5.53%	1.07%	1.80%	18626.86
tai80a	3600	13499184	6.39%	5.56%	1.36%	1.82%	33320.47
tai80b	3600	818415043	12.94%	7.72%	0.00%	1.06%	5389291.60
tai100a	3600	21052466	11.75%	10.11%	1.36%	1.77%	52545.36
tai100b	3600	1185996137	19.98%	10.20%	0.00%	0.87%	11497905.21

Table 7. QAP results

MILP solvers, but offers better scalability, otherwise typical for problem-specific metaheuristic solvers.

Three different $\mathcal{NP}$-hard problems (CVRP, NPFS, and QAP) are formulated in the proposed formalism. The proposed solver is automatically configured separately for each problem and benchmarked against the Gurobi Optimizer. The experiments show that the proposed solver consistently reaches solutions of significantly better quality than Gurobi, given a fixed time limit. Moreover, it is capable of finding good quality solutions to such problems, for which Gurobi either finds no solution at all in the given time or fails to build the problem model due to memory limitations.

In future work, other metaheuristics and low-level components will be implemented to extend the solver capabilities. The solver will also be adapted to solve richer problems in terms of number of constraints, as it relies on transforming these into penalty functions. Finally, the solver will be compared with problem-specific metaheuristic algorithms and further improved toward offering similar scalability and solution quality.

Acknowledgements

This work has been supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000470). The work of David Woller has been also supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS21/185/OH K3/3T/37 and by the Czech Science Foundation (GACR) under Grant Agreement 19-26143X. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Benavides, A.J., Ritt, M.: Fast heuristics for minimizing the makespan in non-permutation flow shops. Computers & Operations Research **100**, 230–243 (2018)
2. Blot, A., Hoos, H.H., Jourdan, L., Kessaci-Marmion, M.É., Trautmann, H.: MO-ParamILS: A multi-objective automatic algorithm configuration framework. In: Learning and Intelligent Optimization, pp. 32–47. Springer Int. Pub., Cham (2016)

3. Burkard, R., Çela, E., Karisch, S., Rendl, F.: QAPLIB (2012). URL https://coral.ise.lehigh.edu/data-sets/qaplib/
4. De Beukelaer, H., Davenport, G.F., De Meyer, G., Fack, V.: JAMES: An object-oriented Java framework for discrete optimization using local search metaheuristics. Software - Practice and Experience **47**(6), 921–938 (2017)
5. Deshwal, A., Belakaria, S., Doppa, J.R., Kim, D.H.: Bayesian optimization over permutation spaces. Proceedings of the AAAI Conf **36**(6), 6515–6523 (2022)
6. Dreo, J., Liefooghe, A., Verel, S., Schoenauer, M., Merelo, J.J., Quemy, A., Bouvier, B., Gmys, J.: Paradiseo: From a modular framework for evolutionary computation to the automated design of metaheuristics: 22 years of Paradiseo. In: GECCO 2021 Companion, pp. 1522–1530. Association for Computing Machinery, Inc (2021)
7. Duarte, A., Mladenović, N., Sánchez-Oro, J., Todosijević, R.: Variable neighborhood descent. In: Handbook of Heuristics, vol. 1-2, pp. 341–367. Springer Int. Pub. (2018)
8. Hadka, D., Reed, P.M., Simpson, T.W.: Diagnostic assessment of the borg MOEA for many-objective product family design problems. In: 2012 IEEE Congress on Evolutionary Computation, CEC 2012 (2012)
9. Helsgaun, K.: An Extension of the LKH TSP Solver for Constrained TSP and VRP. Tech. rep., Roskilde University (2017)
10. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Learning and Intelligent Optimization, pp. 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
11. Lima, I.: CVRPLIB (2014). URL http://vrp.galgos.inf.puc-rio.br/
12. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
13. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: Framework and applications. In: Handbook of Metaheuristics, pp. 129–168. Springer Int. Pub., Cham (2019)
14. Mehdi, M.: Parallel Hybrid Optimization Methods for Permutation Based Problems. Ph.D. thesis, Université des Sciences et Technologie de Lille (2011)
15. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research **24**(11), 1097–1100 (1997)
16. Moghadam, B.F., Sadjadi, S.J., Seyedhosseini, S.M.: Comparing mathematical and heuristic methods for robust VRP. IJRRAS **2**(2), 108–116 (2010)
17. Parejo, J.A., Ruiz-Cortés, A., Lozano, S., Fernandez, P.: Metaheuristic optimization frameworks: A survey and benchmarking. Soft Comput. **16**(3), 527–561 (2012)
18. Scott, E.O., Luke, S.: ECJ at 20: Toward a general metaheuristics toolkit. In: GECCO 2019 Companion, pp. 1391–1398. ACM (2019)
19. Stutzle, T., López-Ibáñez, M.: Automated design of metaheuristic algorithms. In: Handbook of Metaheuristics, pp. 541–579. Springer Int. Pub., Cham (2019)
20. Vallada, E., Ruiz, R., Framinan, J.M.: New hard benchmark for flowshop scheduling problems minimising makespan. European Journal of Operational Research **240**(3), 666–677 (2015)
21. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research **234**(3), 658–673 (2014)
22. Woller, D.: Permutator github repository (2022). URL https://github.com/wolledav/permutator
23. Xia, Y., Yuan, Y.X.: A new linearization method for quadratic assignment problems. Optimization Methods and Software **21**(5), 805–818 (2006)

Chapter 7

The Hamiltonian Cycle and Travelling Salesperson problems with traversal-dependent edge deletion

In this chapter, we present the fifth core publication [c5] - The Hamiltonian Cycle and Travelling Salesperson problems with traversal-dependent edge deletion. This research employs the solver developed in the previous core publication [c4] and is further expanded in [c6] and [r11].

[c5] Carmesin, S., **Woller, D.**, Parker, D., Kulich, M., Mansouri, M., "The Hamiltonian Cycle and Travelling Salesperson problems with traversal-dependent edge deletion", *Journal of Computational Science*, vol. 74, p. 102 156, 2023, ISSN: 1877-7503. DOI: `10.1016/j.jocs.2023.102156`, **20% contribution, IF 3.3 (Q2 in Computer Science, Theory & Methods), citations: 0 in Web of Science, 1 in Scopus, 1 in Google Scholar.**

This research is motivated by a practical robotic application [97]. In this application, an autonomous drill rig is tasked with drilling a dense pattern of blast holes in an open-pit mine. Drilling each blast hole generates a pile of excess material, which represents an obstacle to future movement of the drill rig. Based on this motivation, we proposed the concept of self-deleting graphs, where visiting a vertex results in the deletion of a predefined set of edges. Then, we formulated the Travelling Salesperson Problem on Self-Deleting graphs (TSP-SD), the Hamiltonian Cycle Problem on Self-Deleting graphs (HCP-SD) and their relaxed variants.

The contributions in this paper are both theoretical and practical. We present three proofs addressing various properties of paths on self-deleting graphs. Then we propose an exact solver for TSP-SD and HCP-SD. It implements a backward depth-first search in a given self-deleting graph, which builds a valid solution tour from last node to first. The solver is exceptionally efficient when finding a feasible initial solution. We also propose a metaheuristic solver based on the generic metaheuristic solver [c4], which, on the other hand, is efficient in the optimization settings, but does not guarantee solution feasibility or optimality. Finally, by linking both solvers together, we obtain a hybrid approach combining advantages of both the exact and heuristic approaches.

The experimental results document the performance of both solvers in terms of scalability and solution quality. The best-performing setup combines both approaches: the exact solver, configured to terminate when finding the first feasible solution, reliably constructs an initial solution, while the metaheuristic solver is more suitable for finding a locally optimal solution of good quality. Thus, small instances with up to 30 nodes can be solved to optimality, and near-optimal solutions can be found for up to 1000 nodes. Then, we carry out a statistical analysis of the Hamiltonicity bound w.r.t. the average vertex degree of a HCP-SD problem instance and compare it with the standard HCP. Finally, we evaluate the ability of the generic metaheuristic solver [c4] to find a feasible solution on a large dataset of artificial instances without the exact construction.

Journal of Computational Science 74 (2023) 102156

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

The Hamiltonian Cycle and Travelling Salesperson problems with traversal-dependent edge deletion

Sarah Carmesin [a], David Woller [b,c], David Parker [d], Miroslav Kulich [b], Masoumeh Mansouri [a,*]

[a] School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom
[b] Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Jugoslávských partyzánů 1580/3, Prague 6, 160 00, Czech Republic
[c] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí13, Prague 2, 121 35, Czech Republic
[d] Department of Computer Science University of Oxford, Parks Road, Oxford, OX1 3QD, United Kingdom

ARTICLE INFO

Keywords:
Travelling Salesperson problem
Coverage planning
Metaheuristics
Combinatorial optimization

ABSTRACT

Variants of the well-known Hamiltonian Cycle and Travelling Salesperson problems have been studied for decades. Existing formulations assume either a static graph or a temporal graph in which edges are available based on some function of time. In this paper, we introduce a new variant of these problems inspired by applications such as open-pit mining, harvesting and painting, in which some edges become deleted or untraversable depending on the vertices that are visited. We formally define these problems and provide both a theoretical and experimental analysis of them in comparison with the conventional versions. We also propose two solvers, based on an exact backward search and a meta-heuristic solver, and provide an extensive experimental evaluation.

1. Introduction

Finding a closed loop on a graph where every vertex is visited exactly once is a Hamiltonian Cycle Problem (HCP), and its corresponding optimization problem in a weighted graph is a Travelling Salesperson Problem (TSP). Variants of the HCP and the TSP have been studied for decades. However, the wealth of research on this topic does not cover problems where the availability of an edge in a graph depends on the vertices already visited. This specific type of *dynamic* graph is relevant to many real-world applications, such as open-pit mining, harvesting and painting.

For instance, consider the mining inspired example shown in Fig. 1, where the graph depicts a representation of a mining field and each vertex is a place to be drilled by a drilling machine. The problem is to find a route such that each vertex is visited and drilled exactly once, i.e., an instance of a HCP/TSP. However, in this problem, drilling at a vertex creates a pile of rubble, which not only makes traversing that vertex again impossible but also affects the availability of some edges around it. For example, as depicted in Fig. 1(a), when vertex C is drilled, indicated by a red circle, the rubble obstructs three edges, BD, CD and DA, which are all deleted, whereas a different traversal only results in the removal of edge DA, as shown in Fig. 1(b).

To model a graph that changes due to the path of already visited vertices, as exemplified in the scenario above, we introduce a new class of graphs, called *Self-Deleting* (SD). Using this class, we formally define two new problem variants: the Hamiltonian Cycle Problem with Self-Deleting graphs (HCP-SD), and the Travelling Salesperson Problem with Self-Deleting graphs (TSP-SD). We then compare, both theoretically and experimentally, HCP-SD and TSP-SD with the conventional versions. In particular, we identify how a self-deleting graph compares to a standard graph in terms of shortest paths, and determine where HCP and HCP-SD are equivalent. We also statistically analyse, using the graph's average vertex degree, the threshold point near which the most expensive instances of HCP and HCP-SD are located. Finally, we propose two solvers, based on an exact backward search and a meta-heuristic solver. The performance of each is extensively evaluated through experiments with a dataset based on standard TSPLIB instances as well as randomly generated datasets catering for the specificity of these new variants.

The paper is structured as follows. Section 2 gives an overview of related works. In Section 3, we formally define HCP-SD and TSP-SD followed by formal proofs of properties of self-deleting graphs in Section 4. We present exact and heuristic solvers for HCP-SD and TSP-SD in Section 5. A statistical analysis of HCP-SD is given in Section 6. In Section 7, we evaluate the proposed solvers. We give our conclusions in the final section, Section 8.

* Corresponding author.
E-mail addresses: sxc1431@student.bham.ac.uk (S. Carmesin), wolledav@cvut.cz (D. Woller), david.parker@cs.ox.ac.uk (D. Parker), kulich@cvut.cz (M. Kulich), m.mansouri@bham.ac.uk (M. Mansouri).

https://doi.org/10.1016/j.jocs.2023.102156
Received 27 March 2023; Received in revised form 27 September 2023; Accepted 8 October 2023
Available online 17 October 2023

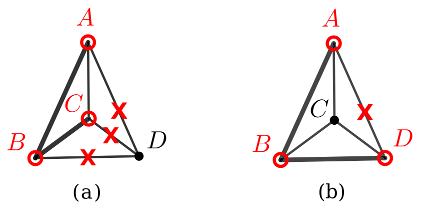


Fig. 1. Representation of a mining example, where due to different traversals, indicated with thicker edges, in (a) and (b) different edges are deleted.

2. Related work

There is a large body of research on the HCP, the TSP and their variants. As mentioned, this paper focuses on a particular type of HCP and TSP where the edges become deleted or untraversable depending on the vertices visited. None of the existing variants of these problems with dynamic graphs has this property. In a TSP on temporal networks, an edge's weight and/or availability changes with respect to some notion of time [1,2], and the unavailable edges can reappear again, as opposed to HCP-SD where the deleted edges are never re-enabled. The other difference is that the weight or availability of an edge in a temporal network changes with time and not due to the way the graph is traversed.

The Covering Canadian Traveller Problem (CCTP) [3] is to find the shortest tour visiting all vertices where the availability of an edge is not known in advance. The traveller only discovers whether an edge is available once reaching one of its end vertices. The availability of an edge is set in advance and not dependent on the traversal.

The Sequential Ordering Problem (SOP), sometimes known as precedence constraint TSP [4], is the problem of finding a minimal cost tour through a graph subject to certain precedence constraints [5]. These constraints are given as a separate acyclic-directed graph. In the SOP, the precedence relation is solely between vertices, however, in our problem we have precedence relations between vertices and edges. Therefore, SOP is a special case of our problem, and we prove this formally in Lemma 4.

The Minimum Latency Problem (MLP) [6,7] is a variant of the TSP where the cost of visiting a node depends on the path that a traveller takes. Given a weighted graph and a path, the latency of a vertex v on that path is defined as the distance travelled on that path until arriving at v for the first time. The goal of the MLP is to find a tour over all vertices such that the total latencies are minimal. Similarly, in our problem the availability of an edge depends on the path taken. However, in a MLP, the graph never changes and the latencies are the result of a simple sum.

On the HCP, some theoretical analysis focuses on investigating conditions, e.g., vertex degree [8,9], under which a graph contains a Hamiltonian cycle. For instance, Pósa [10] and Komlós and Szemerédi [11] proved that there is a sharp threshold for Hamiltonicity in random graphs as the edge density increases. An intuitive approach to finding a Hamilton cycle is to use a depth-first-search (DFS). Rubin [12] introduced some rules to prune the search tree. His rules do not improve the worst-case computation time $O(n!)$, where n is the number of vertices, however statistical analysis has shown that using such criteria improves the average computation time [13,14].

In terms of applications of TSP in automated planning, different variants have been used in coverage route planning [15], e.g., for an autonomous lawnmower [16], or for autonomous drilling of a PCB [17]. Those most relevant to this paper are coverage planning problems whose environments change due to the coverage actions by agents, e.g., robots, that operate within them. The open-pit mining scenario described earlier is an example of such a coverage planning problem for which a specialized solver for the mining case is proposed by [18]. Autonomous harvesting is another instance where heavy vehicles should not pass through the areas already harvested to avoid soil compaction. The harvested areas also limit the mobility of harvesting machines, hence affecting the reachability among the nodes representing areas to be harvested. Ullrich, Hertzberg, and Stiene [19] formulate this application as an optimization problem for which a specialized solver is also proposed. In both cases described above, the authors did not study the theoretical underpinning of the problem, nor provide a general solution that can easily be employed for other instances of problems with traversal-dependent edge deletion.

3. Problem statement

In this section, we formally define self-deleting graphs and introduce the corresponding notions of walks and paths. We then proceed to give a formal definition of the HCP-SD and the TSP-SD problems.

Definition 1. A *self-deleting graph* S is a tuple $S = (G, f)$ where $G = (V, E)$ is a simple, undirected graph and $f : V \to 2^E$. The function f specifies for every vertex $v \in V$ which edges $f(v)$ are deleted from E if the vertex v is processed. We refer to f as the *delete-function*.

If a vertex v is *processed*, we delete edges $f(v)$ from G. For a self-deleting graph S and set $X \subset V$ of vertices, the *residual graph* G_X of S after processing X is defined as:

$$G_X = G \setminus \bigcup_{v \in X} f(v).$$

We call a simple path $p = (v_1, \ldots, v_x)$ in a self-deleting graph *f-conforming* if for every $1 \le i < x$ the edge $e_i = \{v_i, v_{i+1}\}$ is in the residual graph $G_{\{v_1,\ldots,v_i\}}$. An f-conforming simple path p traverses the graph G while processing every vertex on p when it is visited.

In contrast to a path, vertices on a walk can be visited more than once. For a walk on a self-deleting graph, a vertex is processed when it is visited for the last time. Formally, we call a walk $w = (v_1, \ldots, v_x)$ *f-conforming* if for every $1 \le i < x$ the edge $e_i = \{v_i, v_{i+1}\}$ is in the residual graph $G_{\{v_1,\ldots,v_i\}\setminus\{v_{i+1},\ldots,v_x\}}$.

Following standard terminology we call a sequence of vertices $c = (v_1, \ldots, v_x, v_1)$ an f-conforming cycle if $(v_1, \ldots, v_x)$ is an f- conforming path and the edge $\{v_x, v_1\}$ exists in the residual graph G_c. Then, a *Hamiltonian cycle* of self-deleting graph S is an f-conforming cycle that contains all vertices of S exactly once.

Problem 1. Given a self-deleting graph $S = (G, f)$, the *Hamiltonian Cycle Problem on Self-Deleting graphs (HCP-SD)* is to find a Hamiltonian cycle on S.

Problem 2. Given a self-deleting graph $S = (G, f)$, the *weak Hamiltonian Cycle Problem on Self-Deleting graphs (weak HCP-SD)* is to find an (f-conforming) closed walk on S that contains every vertex at least once.

Observation 1. *Every Hamiltonian cycle of S is a Hamiltonian cycle of G.*

This implies that the HCP-SD is at least as hard as finding a Hamiltonian path.

Using a weighted graph as the underlying graph of a self-deleting graph we can define optimization problems on self-deleting graphs as follows.

Problem 3. Given a self-deleting graph $S = (G, f)$, where G is a weighted graph, the *Travelling Salesperson Problem on self-deleting graphs (TSP-SD)* is to find a shortest Hamiltonian cycle on S.

Problem 4. Given a self-deleting graph $S = (G, f)$, the *weak Travelling Salesperson Problem on self-deleting graphs (weak TSP-SD)* is to find a shortest (f-conforming) closed walk on S that contains every vertex at least once.

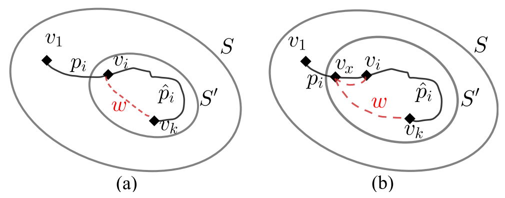


Fig. 2. Illustrations for the proof of Lemma 2: If the path w is shorter than the path $\hat{p}_i$, then p was not a shortest path.

4. Properties of self-deleting graphs

In this section, we provide some formal analysis of self-deleting graphs, in comparison to static graphs. First, we analyse path segments in self-deleting graphs.

Lemma 1. *Let $S = (G, f)$ be a self-deleting graph and $p = (v_1, \dots, v_x)$ be an f-conforming path of S. For every $1 \le i < j \le x$ it holds that the path segment $p_{i,j} = (v_i, \dots, v_j)$ is an f-conforming path of $S_{i,j} = (G', f)$ where G' is the induced subgraph of G on the vertices $(v_i, \dots, v_j)$.*

Proof. Let $p = (v_1, \dots, v_x)$ be an f-conforming path of S and let $1 \le i < j \le x$. By definition, the path segment $p_{i,j} = (v_i, \dots, v_j)$ is an f-conforming path of $S_{i,j} = (G', f)$ if for every $i \le k < j$ it holds that the edge $e_k = \{v_k, v_{k+1}\}$ is in the residual graph $G'_{\{v_i,\dots,v_k\}}$. We now show that, for every $i \le k < j$, the edge $e_k = \{v_k, v_{k+1}\}$ is in the residual graph $G'_{\{v_i,\dots,v_k\}}$.

Assume for a contradiction there is a k with $i \le k < j$ where $e_k \notin G'_{\{v_i,\dots,v_k\}}$. There are two possible reasons for this.

1. $e_k \notin E(G')$: Since e_k is in $E(G)$, G' cannot be an induced subgraph of G and we have a contradiction.
2. e_k gets deleted by some $f(v_y)$, $i \le y \le k$: If $e_k \in \cup_{i \le y \le k} f(v_y)$ then $e_k \in \cup_{1 \le y \le k} f(v_y)$ and therefore $e_k \notin G_{\{v_1,\dots,v_k\}}$. Since $e_k \notin G_{\{p_1,\dots,p_k\}}$ the path p is not f-conforming and we again arrive at a contradiction.

Since both cases yield a contradiction, the lemma holds. □

Let $p = (v_1, \dots, v_x)$ be an f-conforming path from the vertex v_1 to the vertex v_x and let $|p|$ denote the length of the path p. We call p a *shortest f-conforming path* from v_1 to v_x if for every other f-conforming path $\hat{p} = (v_1, \dots, v_x)$ from v_1 to v_x it holds that $|p| \le |\hat{p}|$.

Lemma 2. *Let $p = (v_1, \dots, v_k)$ be a shortest f-conforming path from v_1 to v_k on a self-deleting graph $S = (G, f)$. The following two statements hold:*

1. *For every $1 < i < k$ it holds that the path $p_i = (v_1, \dots, v_i)$ is not necessarily a shortest f-conforming path in S.*
2. *It further holds that the path $\hat{p}_i = (v_i, \dots, v_k)$ is a shortest f-conforming path from v_i to v_k in the self-deleting graph $S' = (G_{\{v_1,\dots,v_i\}}, f)$.*

Proof. Let $p = (v_1, \dots, v_k)$ be a shortest f-conforming path from v_1 to v_k on a self-deleting graph $S = (G, f)$. For any $1 < i < k$ we denote the path segment of p from v_1 to v_i by p_i and the path segment from v_i to v_k by $\hat{p}_i$. Due to Lemma 1, the path segments p_i and $\hat{p}_i$ are f-conforming. We now prove the two statements separately.

1. A shortest f-conforming path from v_1 to v_i could contain a vertex v_j for which $f(v_j)$ deletes an edge needed in the second part $\hat{p}_i$ of the f-conforming path p. So, p_i is not necessarily a shortest f-conforming path from v_1 to v_i.
2. We now prove that the path $\hat{p}_i = (v_i, \dots, v_k)$ is a shortest f-conforming path from v_i to v_k in the self-deleting graph $S' = (G_{\{v_1,\dots,v_i\}}, f)$. For a contradiction assume there is a vertex v_i, with $1 < i < k$, such that there is a f-conforming path w from v_i to v_k in S' that is shorter than $\hat{p}_i$. We consider the following two cases.
   (a) The paths w and p_i do not share a vertex, as depicted in Fig. 2(a). If this is the case, then the path from v_1 to v_k that consists of the path p_i and the path w is shorter than the path p. This is a contradiction.
   (b) The paths w and p_i share a vertex v_x, as depicted in Fig. 2(b). Since by assumption w is f-conforming and $|w| < |\hat{p}_i|$, the walk from v_1 to v_k consisting of p_i and w is shorter than p. We can create an even shorter simple path form v_1 to v_k by omitting the circle that is created by going from v_x to v_i via p and then returning to v_x via w. This is a contradiction to the assumption that p is a shortest path from v_1 to v_k. □

Lemma 2 indicates the inherent difference between static and self-deleting graphs. In static graphs, every segment of a shortest path is a shortest path. This fact is exploited by different algorithms, often based on dynamic programming, for path finding in static graphs, e.g. Dijkstra's algorithm [20]. As a consequence, these types of algorithms cannot easily be applied to self-deleting graphs.

Lemma 3. *Let $S = (G, f)$ be a self-deleting graph, where for every vertex $v \in V(G)$, $f(v)$ deletes only edges that are incident to v, then the Hamiltonian path problem on self-deleting graphs is equivalent to the Hamiltonian path problem on directed graphs.*

Proof. We construct a corresponding directed graph $D = (V, A)$, to a self-deleting graph $S = (G, f)$, where $f(v)$ deletes only edges incident to v, as follows.

$$V(D) = V(G),$$
$$A(D) = \{(v, w) \mid \{v, w\} \in E(G) \land \{v, w\} \notin f(v)\}$$

(Here (v, w) describes the directed arc from v to w, while $\{v, w\}$ describes the undirected edge between v and w.)

Another way to explain this construction is as follows. We make G a directed graph in which each edge is replaced by two arcs in opposite directions. For every vertex v we then delete all outgoing arcs corresponding to an edge in $f(v)$.

We now prove that a path p is f-conforming in S if and only if p is a path in D.

$\Rightarrow$: If the path $p = (v_1, \dots, v_k)$ is f-conforming, it holds by definition that for every $1 \le i < k$ the edge $\{v_i, v_{i+1}\}$ is in the residual graph $G_{\{v_1,\dots,v_i\}}$. Since $\{v_i, v_{i+1}\} \in G_{\{v_1,\dots,v_i\}}$ it holds that $\{v_i, v_{i+1}\} \in E(G)$. Also since $\{v_i, v_{i+1}\} \in G_{\{v_1,\dots,v_i\}}$ it holds that the edge $\{v_i, v_{i+1}\} \notin f(v_x)$ with $1 \le x \le i$, so the edge $\{v_i, v_{i+1}\}$ is in particular not in $f(v_i)$. Therefore the arc (v_i, v_{i+1}) is in $A(D)$.

$\Leftarrow$: Now, let $p = (v_1, \dots, v_k)$ be a simple path in D. So, for every $1 \le i < k$ the arc (v_i, v_{i+1}) is in $A(D)$. For every arc $a = (v, w) \in A(D)$ it holds that the edge $e = \{v, w\}$ is in $E(G)$ and $e \notin f(v)$. Since (v_i, v_{i+1}) is in $A(D)$, it follows that $\{v_i, v_{i+1}\} \notin f(v_i)$. Since no other vertex v_n with $n < i$ is incident to e it follows that $\{v_i, v_{i+1}\} \notin f(v_m)$ for every $m \le i$. So $\{v_i, v_{i+1}\} \in G_{v_1,\dots,v_i}$. Therefore p is f-conforming in S.

Every path through D is an f-conforming path through S and vice-versa. So the Hamiltonian path problem on S is equivalent to the Hamiltonian path problem on D. □

A *sequential ordering problem (SOP)* is defined as a graph $G = (V, E)$ accompanied by a precedence graph P. The precedence graph P is a directed graph defined on the same set of vertices V. It represents the precedence relation between the vertices of G. An edge from v_i to v_j in P implies that v_i must precede v_j in any path through G. The problem is to find a Hamiltonian path in G that does not violate the precedence relation given by P.

Lemma 4. *For every sequential ordering problem SOP there is a corresponding self-deleting graph S_{SOP} such that a path p is a solution to SOP if and only if p is a Hamiltonian path of S_{SOP}.*

Proof. Let a SOP be given by the graph H and the precedence graph P. Let $pre(v) \subseteq V(H)$ be the set of vertices that precede v in P, formally $pre(v) = \{w \mid (w,v) \in A(P)\}$. We construct the corresponding self-deleting graph $S_{SOP} = (G, f)$ as follows.

$$G = H,$$
$$f(v) = \bigcup_{w \in pre(v)} \{e \in E(G) \mid e \text{ incident to } w\}$$

$\Rightarrow$: Let $p = (v_1, \dots, v_k)$ be a path in H that satisfies the precedence relations given in P. So, for every $1 \le i \le j < k$ the vertices v_j and v_{j+1} are not required to precede v_i. Thus, the edges (v_j, v_i) and (v_{j+1}, v_i) are not in P and $v_j, v_{j+1} \notin pre(v_i)$. So by construction of f the edge (v_j, v_{j+1}) does not get deleted by any $f(v_i)$ for $1 \le i \le j$. This implies that the edge (v_j, v_{j+1}) is in the residual graph $G_{\{v_1,\dots,v_j\}}$. Thus, p is f-conforming in S_{SOP}.

$\Leftarrow$: If the path $p = (v_1, \dots, v_k)$ is f-conforming in S_{SOP} it holds per definition that for every $1 \le i < k$ the edge (v_i, v_{i+1}) is in the residual graph $G_{\{v_1,\dots,v_i\}}$. Thus, it holds that the edge (v_i, v_{i+1}) has not been deleted by any vertex v_x with $1 \le x \le i$. It follows that v_i and v_{i+1} are not in $pre(v_x)$ with $1 \le x \le i$. Thus, v_i and v_{i+1} are not required to be visited before v_x with $1 \le x \le i$ and the path p satisfies the precedence conditions in P. It is therefore a valid path in H.

We proved that any valid path in a SOP is f-conforming in the corresponding self-deleting graph and vice-versa. This holds in particular for Hamiltonian paths. □

5. Exact and heuristic solvers

Next, we describe two solvers for the HCP-SD and TSP-SD problems: one that produces an *exact* solution and one which relies on *heuristics*.

5.1. Exact solvers

An intuitive approach to solving the HCP on a self-deleting graph S is to employ a DFS in a forward-search manner: starting with some vertex p_1, we delete all edges in $f(p_1)$ in G, then choose a neighbour p_2 of p_1 as the next vertex on the path and repeat until the path is a Hamiltonian cycle or the current path cannot be extended, in which case we backtrack. This approach can be improved with methods used in algorithms for Hamiltonian cycles in conventional graphs, namely graph/search-tree pruning, as introduced by [12,21]. Their algorithms identify edges that *must* be in a Hamiltonian cycle, e.g., edges incident to a vertex of degree 2, and employ these required edges to improve the average runtime of a forward DFS. However, even with these pruning rules, the algorithm fails to detect paths that cannot be extended to a Hamiltonian cycle early. This is due to the fact that the edge deletion is traversal dependent.

Since failures occurring at a late stage are often due to the choices at an earlier stage of the search, we propose a *backward* search algorithm, shown in Algorithm 1. This takes advantage of the late failures to greatly reduce the size of the search tree. Instead of exploring the path from a start vertex and deleting edges subsequently, Algorithm 1 starts by deleting all edges that would get deleted at some point. It then explores the graph in a backward fashion, adding edges according to visited vertices as follows. During this backward exploration of the graph, edges are added, so searching for required edges, as is done in conventional forward DFS for Hamiltonian cycles, is not possible.

The first call of $backwardSearch$ receives a single start vertex as the path and the self-deleting graph. During the repeated calls of $backwardSearch$, the path grows backwards, so the first call will be with $path = (v_1)$, the next with $path = (v_n, v_1)$, then $path = (v_{n-1}, v_n, v_1)$ and so forth. During each call of $backwardSearch$ the residual graph R with respect to $path$ is calculated (line 1). In line 2 follows a goal check where it is first verified whether the path has the correct length and if so, whether the missing edge between both end vertices exists (line 3). If the initial check fails, the algorithm calculates the set SV of vertices that are candidates for the second vertex in the Hamiltonian path in line 9. If all the candidates are already on $path$ the path cannot be extended to a Hamiltonian cycle. We check this condition in line 10. In line 13 the set N of neighbours of the first vertex of the current $path$ in R is calculated. For every neighbour, $backwardSearch$ is called with an extended path until one of them returns a Hamiltonian cycle.

Algorithm 1 Backward search algorithm for finding a Hamiltonian cycle in a self-deleting graph

```
Input: Current path, the self-deleting graph
Output: Hamiltonian cycle of $S$ or failure
Function: backwardSearch ($path$, $S = (G, f)$)
 1: $R \leftarrow G \setminus \{e \in f(v) \mid v \notin (path \setminus path.last)\}$
 2: if $|path| = |V(G)|$ then
 3:     if $(path.last, path.first) \in E(R)$ then
 4:         return $[path.last] + path$
 5:     else
 6:         return $failure$
 7:     end if
 8: else
 9:     $SV \leftarrow \{v \in V(G) | (path.last, v) \in E(G) \wedge (path.last, v) \notin f(path.last)\}$
10:     if $SV \setminus path = \emptyset$ then
11:         return $failure$
12:     else
13:         $N \leftarrow \{v \mid (path.first, v) \in E(R) \wedge v \notin path\}$
14:         for $v \in N$ do
15:             $result \leftarrow backwardSearch([v] + path, S)$
16:             if $result \neq failure$ then
17:                 return $result$
18:             end if
19:         end for
20:         return $failure$
21:     end if
22: end if
```

Lemma 5. *Let $S = (G, f)$ be a self-deleting graph. If there exists at least one Hamiltonian cycle in S, then the backward search finds a Hamilton cycle.*

Proof. We prove by contradiction: Assume there exists a Hamiltonian self-deleting graph $S = (G, f)$, where the algorithm returns $failure$. Let $n = |V(G)|$ and $P = (p_1, \dots, p_{n+1})$ with $p_1 = p_{n+1}$ a Hamilton cycle of S. We analyse certain function calls to prove the lemma.

If $backwardSearch((p_2, \dots, p_{n+1}), S)$ is called, line 1 calculates the residual graph $R = G \setminus f(p_1)$. Since $|(p_2, \dots, p_{n+1})| = n$, line 3 triggers. The algorithm then checks whether the edge (p_1, p_2) exists in R. If so, P is returned, which is a contradiction since we assumed $failure$ is returned. However, if there is no edge (p_1, p_2) in R then p is not a Hamiltonian cycle, contradicting the assumption.

Therefore $backwardSearch((p_2, \dots, p_{n+1}), S)$ is never called. So there is a largest number $2 \le x \le n$ for which $backwardSearch((p_x, \dots, p_{n+1}), S)$ is never called. We analyse the call $backwardSearch((p_{x+1}, \dots, p_{n+1}), S)$.

In line 1 the residual graph $R = G_{V(G) \setminus \{p_{x+1}, \dots, p_n\}}$ is calculated. Since $|(p_{x+1}, \dots, p_{n+1})| < |V(G)|$, the algorithm continues in line 8. In line 9 the set SV of candidates for the second vertex on the Hamiltonian cycle starting in p_1 is calculated. Since P is a Hamiltonian cycle the set contains at least p_2. And since the current path is $p_{x+1}, \dots, p_n$ with $x \ge 2$, p_2 is not in path and the if-condition in line 10 fails.

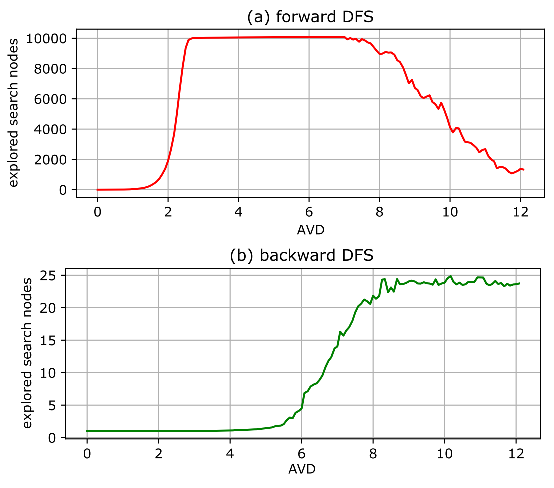


Fig. 3. Explored nodes in the forward-DFS and backward-DFS on the dataset random24-100.

We continue in line 13. Here, the list of neighbours N of current first vertex in R that are not already on the path is calculated. We now consider two cases:

(a) $p_x \notin N$: N contains all neighbours of p_{x+1} in R. So if $p_x \notin R$ then there is no edge between p_x and p_{x+1} in the residual graph after processing $p_1, \ldots, p_x$. Thus, p is no Hamiltonian cycle, a contradiction. So (b) must hold.

(b) $p_x \in N$: The only reason for not calling $backwardSearch((p_x, \ldots, p_{n+1}), S)$ is that another call like $backwardSearch((y, p_{x+1}, \ldots, p_{n+1}), S)$ with $y \in N$ does not return failure. Thus the algorithm finds another Hamilton cycle, this again is a contradiction.

Since we always arrive at a contradiction, the assumption does not hold. Thus, if S is Hamiltonian the algorithm finds a Hamiltonian cycle. $\square$

In order to investigate the behaviour of both exact algorithms, we first need to define the Average Vertex Degree (AVD) for a self-deleting graph. AVD is a metric commonly used in the analysis of static graphs for the HCP. Let k be the number of times an edge e appears in the delete function f. The probability that the edge will be deleted after processing any l vertices from V in arbitrary order is given by $p(e,l) = 1 - \prod_{i=0}^{k-1} \frac{(n-l)-i}{n-i}$. Then, the expected "static" AVD of S after processing any l vertices can be determined as $\delta(l) = (n-1) - \frac{2}{n} \sum_{e \in E} p(e,l)$. From here, we can define the AVD of S as $\frac{1}{n} \sum_{l=1}^{n} \delta(l)$.

A dataset random24-100 of 14 400 random self-deleting graphs with 24 vertices was generated in order to compare both exact algorithms. The delete function f was sampled uniformly randomly with overlapping of $f(v)$ for two distinct v allowed. I terms of AVD, the dataset uniformly covers the interval from 0 to 12. In an experimental comparison between backward and forward search, both solving the same dataset random24-100 and capped at 10 000 expanded search nodes, the backward search performs much better. It was able to solve all instances and on average was able to identify a Hamiltonian instance after 27.9 explored nodes and a non-Hamiltonian instance after 1.6 explored nodes. The forward search failed to find a solution within the limit for most instances. The diagrams in Fig. 3 show the average explored nodes by which either algorithm was able to decide the instance or the limit was reached.

Fig. 4(a) shows the percentage of infeasible instances decided by the backward search at various search depths while using the same random24-100 dataset. Infeasible instances with AVD less than 3 are detected instantly at depth 1. The hardest instances to detect are located between AVDs 6 and 7. Above 7, the dataset does not contain any infeasible instances. Finally, more than 80% of infeasible instances are detected at depth 10, less than half of $|V|$.

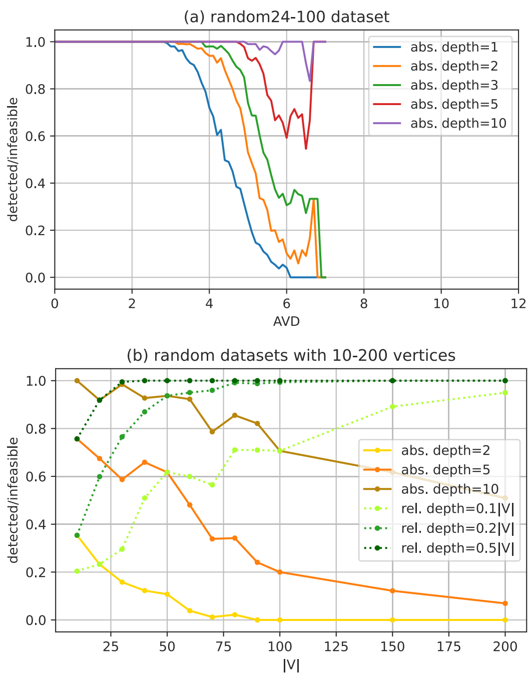


Fig. 4. Backward search behaviour.

Fig. 4(b) illustrates how the percentage of detected infeasible instances at various depths depends on $|V|$. At a fixed depth, the percentage unsurprisingly decreases with increasing $|V|$, but even for $|V| = 200$ about 50% instances are detected at depth 10. Interestingly, the percentage increases when using a relative depth and close to 100% infeasible instances are detected at depth $0.2|V|$, when $|V| > 100$. This experiment indicates that the backward search algorithm's ability to detect infeasible instances of HCP-SD early on in the search improves with increasing $|V|$ and, consequently, the algorithm may be scalable enough to find feasible solutions even for instances with $|V|$ of practical interest.

5.2. Heuristic solver

The proposed exact solver is likely to provide limited scalability when addressing *optimization* problems due to its exhaustive nature. Also, finding near-optimal solutions is often sufficient in practical applications, therefore, heuristic algorithms may be the only computationally feasible approach to obtain them. A common procedure is to design a problem-specific metaheuristic algorithm, that is tailored to a particular application. Various heuristic approaches were successfully applied to problems related to the TSP-SD, such as metaheuristics based on local search [22], evolutionary optimization [23] or swarm optimization [24].

In this paper, we use a generic metaheuristic solver for problems with permutative representation [25], so that we can remain application agnostic regarding multiple variants of TSP-SD. The solver implements several high-level metaheuristics and also a bank of low-level local search operators, perturbations and construction procedures.

These can be readily applied to various problems, whose solution can be encoded as a sequence of potentially recurring nodes. The only user requirement is to specify a set of nodes A, lower and upper bounds L, U of the frequency of their occurrence in a solution sequence $x = (x_1, x_2, \dots, x_n)$, where $x_i \in A$; a fitness function $f(x)$ and an aggregation of penalty functions $g(x)$. The bounds are always respected by the solver, whereas the penalty functions are treated as soft constraints. Their purpose is to direct the search process towards valid solutions. TSP-SD can be described in the solver formalism as follows:

$$A = \{v_1, v_2, \dots, v_n\} = V(G),$$
$$L = (1, 1, \dots, 1) = U,$$
$$f(x) = \sum_{i=1}^{n} \|e_i\|,$$
$$g(x) = \sum_{i=1}^{n} g_i(x), \text{ where}$$
$$g_i(x) = \begin{cases} 0, & \text{if } e_i \in E(G_{\{x_1, x_2, \dots, x_i\}}), \\ M, & \text{otherwise.} \end{cases}$$

Here, the set of nodes to visit A corresponds to the set of vertices $V(G)$. Each node v_i has to be processed exactly once, thus $L_i = U_i = 1$. Then, e_i is the edge $\{x_i, x_{i+1 \mod n}\}$, $G_{\{x_1, x_2, \dots, x_i\}}$ is the residual graph after processing first i nodes in x and M is a large constant introduced to penalize using an already deleted edge e_i in x. The goal is to minimize the total length of the cycle given by x and force all penalties $g_i(x)$ to zero, if possible.

For the weak TSP-SD, both the set of nodes A and the respective bounds L, U are defined in the same way as in the TSP-SD, but the definition of $f(x)$ and $g_i(x)$ differs:

$$f(x) = \sum_{i=1}^{n} \|p_i\|,$$
$$g_i(x) = \begin{cases} 0, & \text{if } p_i \text{ exists in } G_{\{x_1, x_2, \dots, x_i\}}, \\ M, & \text{otherwise.} \end{cases}$$

Here, p_i is the shortest path from x_i to $x_{i+1 \mod n}$ in the residual graph $G_{\{x_1, x_2, \dots, x_i\}}$, which is found using the A* algorithm [26]. Thus, the time complexity of weak TSP-SD fitness evaluation is higher than TSP-SD by $\mathcal{O}(|E|)$. Only the first and last vertex of p_i are processed. If p_i does not exist, a large constant M is added to the penalty $g(x)$ via $g_i(x)$. The goal is to minimize the total length of the closed walk given by x.

6. Statistical analysis of HCP-SD

In this section, we investigate properties analogous to those previously studied in the literature for HCP, since they are crucial for understanding behaviour and evaluating the performance of the proposed solvers. For the HCP, the probability density function of a randomly generated graph being Hamiltonian was experimentally shown to be sigmoidally shaped around a certain threshold point [14]. This threshold corresponds to the graph's AVD, for which the probability is approximately 0.5. Their experiments indicate that HCP instances close to this boundary are the most expensive to decide for various exact algorithms in terms of computational cost, although isolated clusters of hard instances were also identified far away from it. The location of this threshold has been proved to be $ln(V) + ln(ln(V))$, which is called the Komlós–Szemerédi bound [11].

First, we replicated the experiment from [14], showing the probability density function of Hamiltonicity for a randomly generated graph with 24 vertices. For this purpose, we generated a dataset of 100 random graphs for every number of edges from 1 to 144, resulting in 14 400 graphs with AVD ranging from 0 to 12. The HCP was decided for the whole dataset using the Concorde TSP solver and the result of the experiment is shown in Fig. 5(a) - HCP (exact). The dataset random24-100 of 14 400 random self-deleting graphs with 24 vertices was created analogously, covering the same range of AVDs. On this dataset, HCP-SD was decided with both an exact and heuristic solver and weak HCP-SD with a heuristic solver described in Section 5. The exact solver was always terminated after successfully deciding the problem, whereas the heuristic solver was terminated either after finding a feasible solution, or reaching a time budget of $|V|$ seconds. Therefore, the heuristic solver's results are suitable for assessing the solver's properties, rather than reasoning about the problem itself. Fig. 5(a) indicates that the probability density function of HCP-SD is shaped similarly to that of HCP but is steeper and the threshold point is located further to the right.

The weak HCP-SD appears to have similar properties, but there is no exact solver available, and using the heuristic solver may affect the location of the threshold point, as it may label a feasible instance as infeasible. We can see that instances with AVD less than 3 that were shown to be easy to decide for the exact solver in Fig. 4(a), actually have zero probability of being Hamiltonian. Instances with AVD between 6 and 7, which were shown to be the hardest to decide, are located close to the HCP-SD Hamiltonicity threshold point. Thus, in a similar fashion to HCP, HCP-SD instances close to the threshold point are computationally harder for the exact solver.

Second, 12 more datasets of random self-deleting graphs with 10 to 200 vertices and uniformly randomly sampled f were generated to investigate the Hamiltonicity bound w.r.t. to $|V|$ for both variants of HCP-SD. Each of these datasets was generated to cover an interval that contains the threshold point of both problems and consisted of 2500 instances, evenly distributed across the interval into groups of 50 instances with the same AVD. Again, the HCP-SD was decided with an exact and heuristic solver and the weak HCP-SD with a heuristic solver, and the location of the threshold point was determined for each dataset and problem. The locations of the threshold points are shown in Fig. 5(b), thus showing a bound analogous to the Komlós–Szemerédi bound. The bound HCP-SD (exact) follows a sublinear, presumably logarithmic trend, similar to the Komlós–Szemerédi bound but faster growing. As for the weak HCP-SD, the heuristic data evidently do not provide an accurate estimate of the bound.

The threshold points should never be higher than for the HCP-SD because all self-deleting graphs feasible in HCP-SD are also feasible in weak HCP-SD. The bound HCP-SD (heuristic) illustrates that the heuristic solver consistently struggles with finding feasible solutions close to the real Hamiltonicity bound, found by the exact solver.

7. TSP-SD solvers evaluation

So far, we have focused only on the results relevant to decision problems, but both proposed solvers are designed to address the formulated optimization problems as well. Each solver has unique properties that are investigated in a series of eight experiments on a newly created dataset.[1] The dataset consists of 11 instances of self-deleting graphs with a size ranging from 14 to 1084 vertices. The instances are selected from the TSPLIB library [28], but a uniformly randomly generated delete function f is added. To give an idea about the delete function, Fig. 6 shows the sets of edges deleted by processing four different nodes in the instance berlin52-13.2. In terms of the AVD, most of the instances are generated close to the HCP-SD Hamiltonicity bound of the heuristic solver so that they could be solved by the heuristic solver alone. The following naming format is used: *original_name*$|V|$-AVD.

The heuristic solver offers a portfolio of alternative components, each suitable for a different set of problems with permutative representation. The solver must be tuned to achieve the best performance for a specific problem. The tuning was carried out using the irace package [29] with a tuning budget of 2500 experiments. The configuration obtained is shown in Table 1. The tuner selected the Basic Variable

[1] All datasets and codes are publicly available at [27].

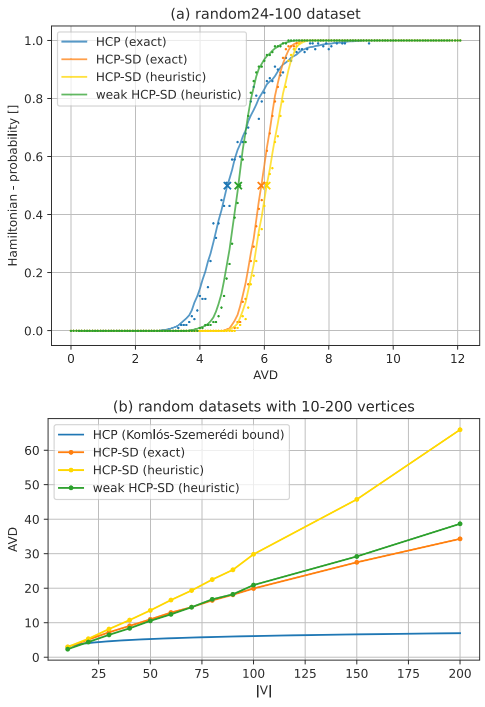


Fig. 5. Comparison of Hamiltonicity bounds.

Table 1
Heuristic solver - tuned configuration.

Component	Value
Metaheuristic	basicVNS ($k_{min} = 7, k_{max} = 10$)
Construction	nearestNeighbor
Perturbation	randomMoveAll ($allowInfeasible = true$)
Local search	pipeVND ($firstImprove = true$)
Operators	centeredExchange ($p \in \{1, 2, 3, 5\}$), moveAll ($p \in \{2, 10\}$)
	relocate($p \in \{1, 2, 3, 4, 5\}$), exchangeIds
	exchange($p, q \in \{(1, 2), (2, 4), (3, 4)\}$)
	reverseExchange($p, q \in \{(1, 2), (2, 2), (3, 3), (3, 4), (4, 4)\}$)

Neighborhood Search (VNS) [30] to use as a high-level metaheuristic and the Pipe Variable Neighborhood Descent (VND) [31] to control the local search. The results of the exact solver were generated on a dedicated machine with Ubuntu 18.04 OS, Intel Core i7-7700 CPU. Experiments using the heuristic solver were generated on an AMD EPYC 7543 CPU cluster. Each instance was solved once by the exact solver and 50 times by the heuristic solver, since it is stochastic. The heuristic solver always had a time budget of $10|V|$ seconds per single run. We present the results in Tables 2, 3 and 4. Individual experiments are referred to by the column letter of the corresponding table. Finally, the relative improvement brought by an experiment B relative to an earlier experiment A in a particular instance i is calculated as $100 \times (1 - \frac{obj_i(B)}{obj_i(A)})$, where $obj_i(A)$ is the objective value on i in A. This value is eventually averaged across the entire dataset.

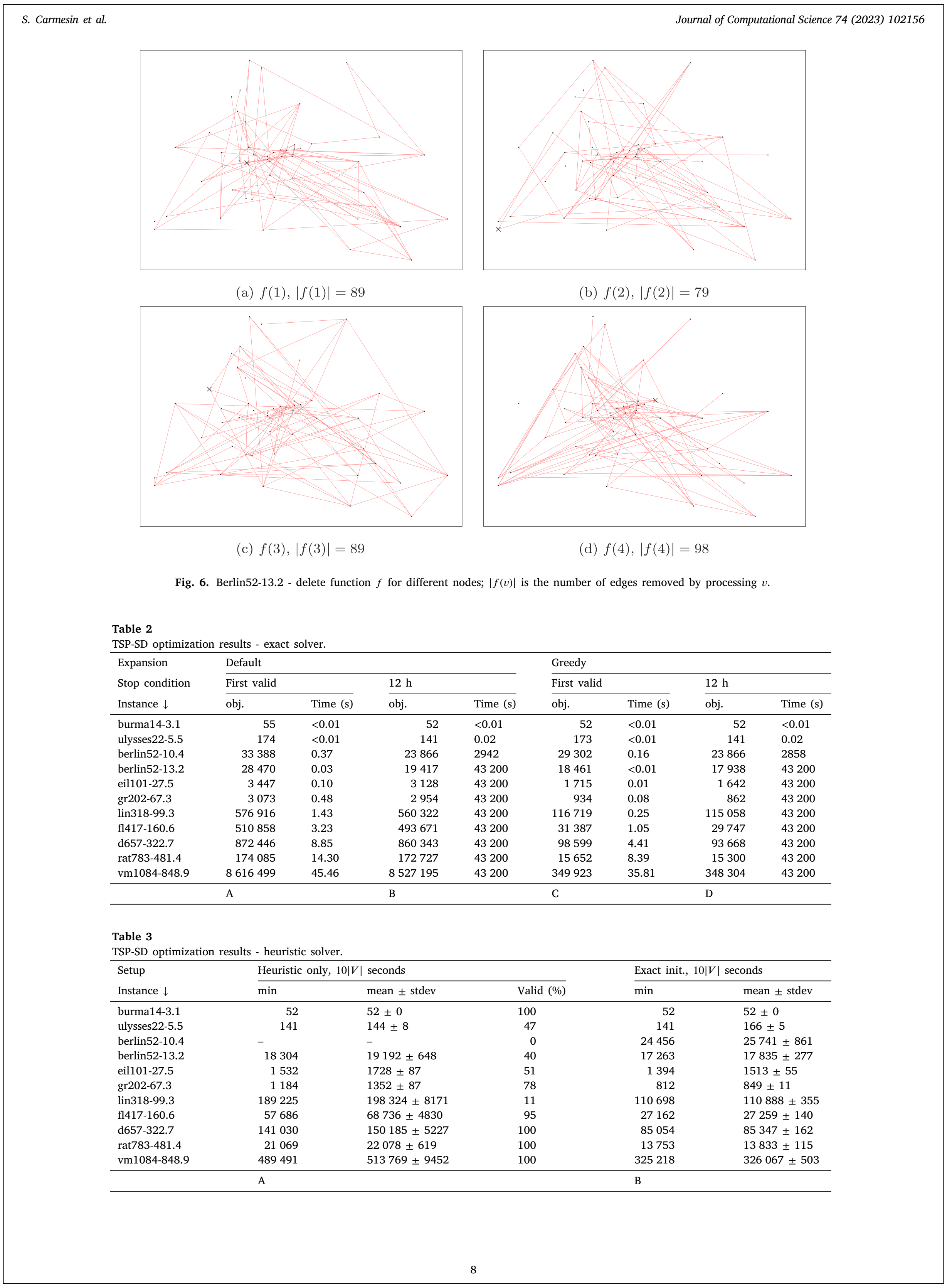
The proposed backward search is introduced as a decision algorithm for the HCP-SD in Algorithm 1. To address the optimization problem TSP-SD, only a slight modification is required. The algorithm does not stop when the first valid solution is found (line 4). Instead, it continues to search until a given time limit is reached while storing the best solution found so far. Another minor modification is the order of expansion at line 14. In the default variant, the nodes $v \in N$ are traversed in arbitrary order, determined by the iterator implementation of the set N. In the following experiments, a greedy expansion is also tested. In this variant, nodes $v \in N$ are sorted according to their distance from $path.first$ and expanded from closest to farthest.

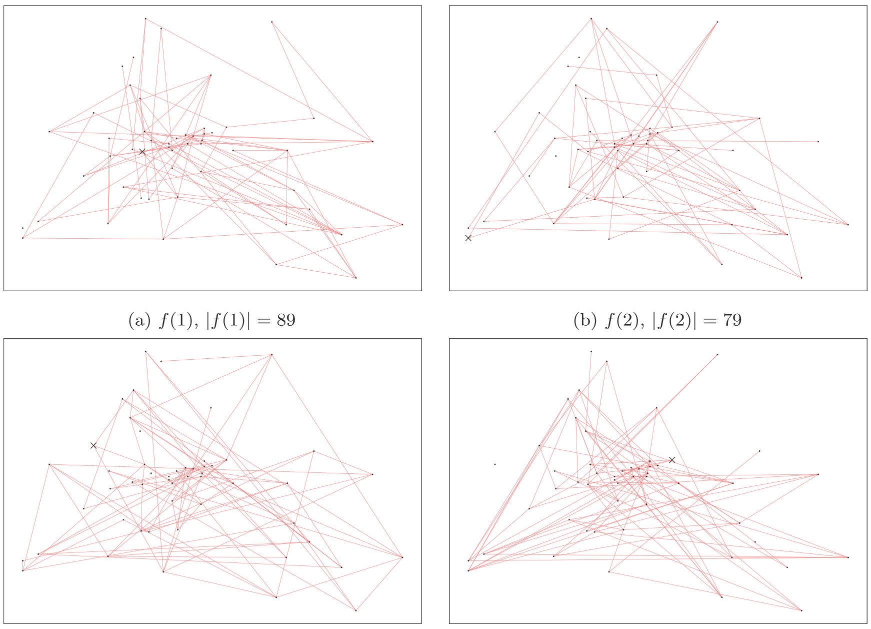
Table 2 documents the performance of the exact TSP-SD solver. The backward search performs the path expansion in default order in experiments in columns A and B, whereas greedy expansion is used in experiments in columns C and D. Column A presents the objective values and computation times needed to find the first valid solution of TSP-SD while using the default expansion. A solution is found within one second for instances with up to $|V| = 202$ and within one minute for all instances in the dataset. The dataset contains two variants of the berlin52 instance with different values of AVD, from which the berlin52-10.4 instance is closer to the Hamiltonicity bound. Finding a valid solution for berlin52-10.4 requires 10 times more time than berlin52-13.2. Thus, AVD seems to be an important factor playing against the backward search. The scalability of the exact solver in this experiment is surprisingly good, as was already indicated in Fig. 4(b).

In Table 2, column B, the exact solver was given a budget of 12 h to solve the TSP-SD for each instance. The first three were solved to optimality, but the remaining eight reached the time limit. On average, the first valid solution was improved by 9.75%, but the improvement decreases with increasing instance size. In the case of the three largest instances, the improvement is only 1%. This experiment only confirms the expectation of poor scalability when using an exact approach in an optimization problem due to its exhaustive nature. Unlike in the previous experiment, the berlin52-10.4 variant was actually easier to solve when addressing the optimization problem, as the backward search tree is presumably pruned more with a lower AVD.

Table 2, column C, depicts the benefit of using the greedy expansion in the backward search. The computation times needed to find the first valid solution are slightly, but consistently better than with the default expansion. More importantly, the objective values are frequently more than ten times better than with the default expansion, which is a considerable improvement brought by a simple heuristic rule. On average, the first valid solutions found with the greedy expansion are better by 56% than with the default expansion. The improvement increases with increasing instance size and is around 90% for the four largest instances. Fig. 7(a) shows the best solution obtained by the exact solver with default expansion, while Fig. 7(b) with greedy expansion. The figures illustrate that using the default expansion is equivalent to generating a random valid solution, whereas the greedy solution behaves reasonably in less dense areas. As shown in Table 2, column D, increasing the time budget to 12 h further improves the objective by 6% on average relative to the first valid greedy solutions. Similarly to random expansion, this improvement decreases with increasing instance size and is less than 1% for the largest instance.

Table 3, column A, presents the results of the heuristic solver alone on the TSP-SD. Each instance was solved 50 times with a time budget of $10|V|$ seconds, e.g. 140 s for the burma14-3.1 instance. The optimal solution was found for the two smallest instances. However, the solver cannot find a valid solution every time and fails entirely to provide any valid solutions in all 50 runs for the berlin52-10.4 instance. In terms of solution quality, the best solutions found by the heuristic solver alone are worse by 26% on average than the first valid solutions found by the greedy exact solver. Furthermore, the mean success rate is only 62%. The heuristic solver is expected to converge faster than the exact solver, but presumably spends a large portion of the time budget on finding a valid initial solution instead. This assumption is confirmed in Table 3, column B, where the heuristic solver is initialized with the first valid solution found by the exact solver (Table 2, column C). Here, the best solutions found by the warm-started heuristic solver in $10|V|$ seconds are better by 5% on average than those obtained by the greedy exact solver in 12 h and by 11.3% than the first valid solutions. Most importantly, the improvement does not decrease with increasing instance size and is consistent across the entire dataset. The





(a) $f(1)$, $|f(1)| = 89$ (b) $f(2)$, $|f(2)| = 79$

(c) $f(3)$, $|f(3)| = 89$ (d) $f(4)$, $|f(4)| = 98$

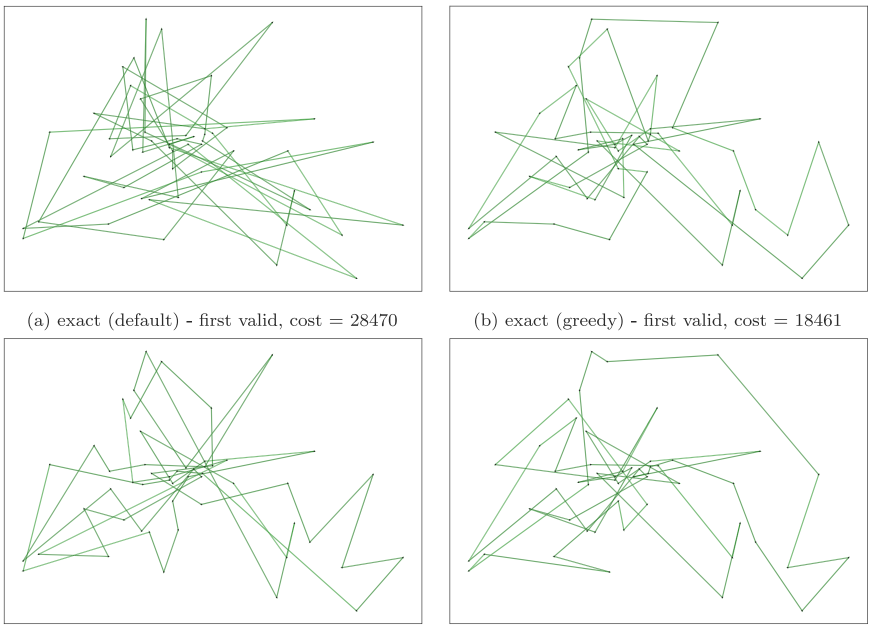
Fig. 6. Berlin52-13.2 - delete function f for different nodes; $|f(v)|$ is the number of edges removed by processing v.

Table 2
TSP-SD optimization results - exact solver.

Expansion	Default				Greedy			
Stop condition	First valid		12 h		First valid		12 h	
Instance ↓	obj.	Time (s)	obj.	Time (s)	obj.	Time (s)	obj.	Time (s)
burma14-3.1	55	<0.01	52	<0.01	52	<0.01	52	<0.01
ulysses22-5.5	174	<0.01	141	0.02	173	<0.01	141	0.02
berlin52-10.4	33 388	0.37	23 866	2942	29 302	0.16	23 866	2858
berlin52-13.2	28 470	0.03	19 417	43 200	18 461	<0.01	17 938	43 200
eil101-27.5	3 447	0.10	3 128	43 200	1 715	0.01	1 642	43 200
gr202-67.3	3 073	0.48	2 954	43 200	934	0.08	862	43 200
lin318-99.3	576 916	1.43	560 322	43 200	116 719	0.25	115 058	43 200
fl417-160.6	510 858	3.23	493 671	43 200	31 387	1.05	29 747	43 200
d657-322.7	872 446	8.85	860 343	43 200	98 599	4.41	93 668	43 200
rat783-481.4	174 085	14.30	172 727	43 200	15 652	8.39	15 300	43 200
vm1084-848.9	8 616 499	45.46	8 527 195	43 200	349 923	35.81	348 304	43 200
	A		B		C		D	

Table 3
TSP-SD optimization results - heuristic solver.

Setup	Heuristic only, $10|V|$ seconds			Exact init., $10|V|$ seconds	
Instance ↓	min	mean ± stdev	Valid (%)	min	mean ± stdev
burma14-3.1	52	52 ± 0	100	52	52 ± 0
ulysses22-5.5	141	144 ± 8	47	141	166 ± 5
berlin52-10.4	–	–	0	24 456	25 741 ± 861
berlin52-13.2	18 304	19 192 ± 648	40	17 263	17 835 ± 277
eil101-27.5	1 532	1728 ± 87	51	1 394	1513 ± 55
gr202-67.3	1 184	1352 ± 87	78	812	849 ± 11
lin318-99.3	189 225	198 324 ± 8171	11	110 698	110 888 ± 355
fl417-160.6	57 686	68 736 ± 4830	95	27 162	27 259 ± 140
d657-322.7	141 030	150 185 ± 5227	100	85 054	85 347 ± 162
rat783-481.4	21 069	22 078 ± 619	100	13 753	13 833 ± 115
vm1084-848.9	489 491	513 769 ± 9452	100	325 218	326 067 ± 503
	A			B	



(a) exact (default) - first valid, cost = 28470

(b) exact (greedy) - first valid, cost = 18461

(c) heuristic - $10|V|$ sec., cost = 18304

(d) exact init. + heuristic, - $10|V|$ sec., cost = 17263

Fig. 7. Berlin52-13.2 - best TSPSD solutions of different solvers and setups.

Table 4
Weak TSP-SD optimization results - heuristic solver.

Setup	Heuristic only, $10\|V\|$ seconds			TSP-SD best init., $10\|V\|$ seconds	
Instance ↓	min	mean ± stdev	Valid (%)	min	mean ± stdev
burma14-3.1	52	52 ± 0	100	52	52 ± 0
ulysses22-5.5	129	129 ± 1	100	129	129 ± 0
berlin52-10.4	18 701	20 328 ± 1174	100	18 354	19 740 ± 480
berlin52-13.2	14 579	15 760 ± 593	100	14 838	16 320 ± 585
eil101-27.5	1 313	1442 ± 69	100	1 240	1295 ± 23
gr202-67.3	886	1060 ± 122	100	779	790 ± 2
lin318-99.3	135 965	143 259 ± 5488	100	104 422	104 945 ± 204
fl417-160.6	26 035	26 891 ± 733	100	25 976	26 001 ± 33
d657-322.7	96 213	99 730 ± 1527	100	83 402	83 534 ± 44
rat783-481.4	15 072	15 409 ± 174	90	13 599	13 620 ± 5
vm1084-848.9	352 794	360 779 ± 3296	76	319 335	319 481 ± 112
	A			B	

previous two experiments reveal the drawbacks of both approaches: the exact solver scales poorly in the optimization problem, whereas the penalty-based heuristic solver does not provide a valid solution reliably. On the other hand, the exact solver provides valid solutions to all instances very fast, and the heuristic solver is much better at refining good-quality solutions. Therefore, using both solvers sequentially, i.e., implementation of a warm start optimization, combines the advantages of both. Fig. 7(c) shows the best solution of berlin52-13.2 obtained by the heuristic solver alone while Fig. 7(d) the best-known solution, obtained by the warm-started heuristic solver. Both solutions remain entangled in the centre area with the most vertices, which may be attributed to the naturally denser randomly generated delete function f in this area, as indicated in Fig. 6.

Table 4 illustrates the benefit of relaxing TSP-SD to weak TSP-SD. Every solution to the TSP-SD is also valid for the weak TSP-SD, but the weak formulation might yield a better optimal value. On the other hand, the fitness evaluation in weak TSP-SD calculates the shortest paths p_i instead of reading the edge weights. Thus, the time complexity of the evaluation is higher by $\mathcal{O}(|E|)$, and the heuristic solver is drastically slower when solving the weak TSP-SD. The performance of the heuristic alone is shown in Table 4A. Regarding the success rate, the heuristic is significantly more successful than with TSP-SD, as the space of valid solutions in the weak TSP-SD formulation is much larger. In Table 4, column B, the best-known TSP-SD solution from the initialized heuristic solver (Table 3, column B) was used as an initial solution. The experiment shows that only the TSP-SD solution of the smallest instance was not improved in the weak TSP-SD formulation. In the remaining instances, the weak TSP-SD solution is better by 7% on average than the best-known TSP-SD solution, so the relaxation is highly beneficial.

8. Conclusions

We introduce new variants of the Hamiltonian Cycle and the Travelling Salesperson Problems with self-deleting graphs, for which formal definitions, theoretical analyses and two solvers were proposed. In the future, we intend to investigate general heuristics for the proposed backward search. We also want to develop a new solver which works in the space of feasible solutions. Finally, we intend to study how to derive self-deleting graphs using motion planning techniques to determine which edges should be deleted.

CRediT authorship contribution statement

Sarah Carmesin: Methodology, Software, Formal analysis, Investigation, Writing – original draft. **David Woller:** Methodology, Software, Formal analysis, Investigation, Writing – original draft. **David Parker:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Miroslav Kulich:** Conceptualization, Methodology, Software, Writing – review & editing, Supervision. **Masoumeh Mansouri:** Conceptualization, Methodology, Software, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code are publicly available and referenced in the manuscript.

Acknowledgements

The research was supported by Czech Science Foundation Grant No. 23-05104S. The work of David Woller has also been supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/122/OHK3/2T/13. Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic. Masoumeh Mansouri is a UK participant in Horizon Europe Project CONVINCE, and her work is supported by UKRI grant number 10042096.

References

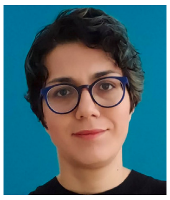
[1] E. Aaron, D. Krizanc, E. Meyerson, DMVP: foremost waypoint coverage of time-varying graphs, in: International Workshop on Graph-Theoretic Concepts in Computer Science, Springer, 2014, pp. 29–41.
[2] O. Michail, P.G. Spirakis, Traveling salesman problems in temporal graphs, Theoret. Comput. Sci. 634 (2016) 1–23.
[3] C.-S. Liao, Y. Huang, The covering Canadian traveller problem, Theoret. Comput. Sci. 530 (2014) 80–88.
[4] D. Chan, Precedence constrained TSP applied to circuit board assembly and no wait flowshop, Int. J. Prod. Res. 31 (9) (1993) 2171–2177.
[5] L.F. Escudero, An inexact algorithm for the sequential ordering problem, European J. Oper. Res. 37 (2) (1988) 236–249.
[6] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, in: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 1994, pp. 163–171.
[7] J. Mikula, M. Kulich, Solving the traveling delivery person problem with limited computational time, CEJOR Cent. Eur. J. Oper. Res. (2022) 1–31.
[8] G.A. Dirac, Some theorems on abstract graphs, Proc. Lond. Math. Soc. 3 (1) (1952) 69–81.
[9] O. Ore, Note on Hamilton circuits, Amer. Math. Monthly 67 (1) (1960) 55, URL http://www.jstor.org/stable/2308928.
[10] L. Pósa, Hamiltonian circuits in random graphs, Discrete Math. 14 (4) (1976) 359–364.
[11] J. Komlós, E. Szemerédi, Limit distribution for the existence of Hamiltonian cycles in a random graph, Discrete Math. 43 (1) (1983) 55–63.
[12] F. Rubin, A search procedure for Hamilton paths and circuits, J. ACM 21 (4) (1974) 576–580.
[13] B. Vandegriend, Finding Hamiltonian Cycles: Algorithms, Graphs and Performance, University of Alberta, 1999.
[14] J. Sleegers, D.v.D. Berg, Backtracking (the) algorithms on the Hamiltonian cycle problem, Int. J. Adv. Intell. Syst. 14 (1–2) (2022) 1–13.
[15] D.L. Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, The Traveling Salesman Problem, Princeton University Press, 2011.
[16] E.M. Arkin, S.P. Fekete, J.S. Mitchell, Approximation algorithms for lawn mowing and milling, Comput. Geom. 17 (1–2) (2000) 25–50.
[17] M. Grötschel, M. Jünger, G. Reinelt, Optimal control of plotting and drilling machines: a case study, Z. Oper. Res. 35 (1) (1991) 61–84.
[18] M. Mansouri, F. Lagriffoul, F. Pecora, Multi vehicle routing with nonholonomic constraints and dense dynamic obstacles, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 3522–3529.
[19] A. Ullrich, J. Hertzberg, S. Stiene, ROS-based path planning and machine control for an autonomous sugar beet harvester, in: Proceedings of International Conference on Machine Control & Guidance, (MCG-2014), 2014.
[20] E. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[21] B. Vandegriend, J. Culberson, The Gn, m phase transition is not hard for the Hamiltonian Cycle problem, J. Artificial Intelligence Res. 9 (1998) 219–245.
[22] K. Helsgaun, An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems, Tech. rep., Roskilde University, Roskilde, 2017, pp. 24–50.
[23] N.R. Sabar, A. Bhaskar, E. Chung, A. Turky, A. Song, A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion, Swarm Evol. Comput. 44 (2019) 1018–1027.
[24] X. Xiang, J. Qiu, J. Xiao, X. Zhang, Demand coverage diversity based ant colony optimization for dynamic vehicle routing problems, Eng. Appl. Artif. Intell. 91 (2020) 103582.
[25] D. Woller, J. Hrazdíra, M. Kulich, Metaheuristic solver for problems with permutative representation, in: Intelligent Computing & Optimization, Springer International Publishing, Cham, 2022, pp. 42–54.
[26] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107, http://dx.doi.org/10.1109/TSSC.1968.300136.
[27] D. Woller, TSP-SD resources, http://imr.ciirc.cvut.cz/Research/TSPSD, Last accessed: 18.10.2023.
[28] G. Reinelt, TSPLIB—A traveling salesman problem library, ORSA J. Comput. 3 (4) (1991) 376–384.
[29] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, M. Birattari, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58, http://dx.doi.org/10.1016/j.orp.2016.09.002.
[30] P. Hansen, N. Mladenović, J. Brimberg, J.A.M. Pérez, Variable neighborhood search, in: Handbook of Metaheuristics, Springer, 2019, pp. 57–97.
[31] A. Duarte, J. Sánchez-Oro, N. Mladenović, R. Todosijević, Variable neighborhood descent, in: Handbook of Heuristics, Springer International Publishing, 2018, pp. 341–367.

Sarah Carmesin is currently a Ph.D student at the University of Birmingham. She received her M.Sc. degree in Computer Science from the Fernuniversität Hagen in 2021. Her research interests lie in the intersections of combinatorics, algorithms and robot coverage planning.

David Woller, M.Sc., received his M.Sc. degree in Cybernetics and Robotics from the Czech Technical University (CTU) in Prague in 2019. He is currently a Ph.D. student at the Czech Institute of Informatics, Robotics, and Cybernetics, CTU. He spent 3 months at a research internship at the Avignon University, Laboratory of Informatics, France. His research interests include Combinatorial Optimization methods, especially metaheuristics for large-scale planning and scheduling optimization problems.

David Parker is a Professor of Computer Science at the University of Oxford. His research is in formal verification, with a particular focus on the analysis of probabilistic systems, and he leads the development of the widely used probabilistic verification tools PRISM and PRISM-games. His current research interests include the development of verification techniques for applications in AI and machine learning, and the use of game-theoretic methods for formal verification.

Miroslav Kulich is currently an assistant professor at the Czech Institute of Informatics, Cybernetics, and Robotics, Czech Technical University (CTU) in Prague. He received his Ph.D. degree in Artificial Intelligence and Biocybernetics at CTU in Prague, Faculty of Electrical Engineering in 2004, and RNDr. degree at Charles University in Prague, Faculty of Mathematics and Physics in 2005. He spent 6 months at a research fellowship at the Helsinki University of Technology, Automation Technology Laboratory, Finland. His research interests include planning for single and multi-robot systems, especially in exploration and search&rescue scenarios and data fusion and interpretation.



Masoumeh Mansouri is currently an associate professor in the School of Computer Science at the University of Birmingham, UK. Previously, she was a researcher at the Center for Applied Autonomous Sensor Systems at Örebro University, Sweden, where she received her Ph.D. as well. She was also a visiting researcher at the Oxford Robotics Institute and had a research stay in Sven Koenig's lab at the University of Southern California. Her research interest includes hybrid methods that integrate automated task/motion/coverage planning, scheduling, as well as temporal and spatial reasoning.

Chapter 8

Where to place a pile?

The last core publication is called Where to place a pile? [c6] and it presents a follow up research to the core publication [c5]. The methods presented in [c5] and [c6] are combined together in [r11].

[c6] Kulich, M., **Woller, D.**, Carmesin, S., Mansouri, M., Přeučil, L., "Where to Place a Pile?", in *2023 European Conference on Mobile Robots (ECMR)*, IEEE, 2023. DOI: `10.1109/ecmr59166.2023.10256330`, **20% contribution, citations: 0 in Web of Science, 0 in Scopus, 0 in Google Scholar.**

This publication focuses on an important subproblem of the open-pit mining application described in Chapter 7, which was not addressed in [c5]. Given a TSP path on a simple undirected graph, the goal is to place a circular obstacle near each node once it is visited. The path must not collide with any of the circles, and the common radius of the circles is to be maximized. This problem is formally defined as the Path-Conforming Circle Placement Problem (PCCP). We also propose a relaxed variant, where only the path segment that is yet to be traversed must not collide with any of the circles already placed (Weak PCCP).

The main theoretical result of the paper is the determination of the lower and upper bounds for the maximal circle radius in the PCCP. In addition to estimating the optimal value, the bounds are utilized to speed up the proposed algorithms. Then, we propose a local search heuristic algorithm for the PCCP with a fixed radius, which attempts to find a valid placement given radius value as input. The crucial part of the algorithm is the fast identification of valid circle centers, which is achieved by intersecting Voronoi diagrams for nodes and path segments with all candidate circle centers. This algorithm is then used in a simple interval bisection algorithm, which addresses the main problem, the PCCP. The overall approach always finds a feasible solution, although finding the highest radius possible is not guaranteed.

The experimental results document the scalability and solution quality w.r.t. the bound estimates on two artificial datasets. For the PCCP, the proposed algorithm is capable of solving instances with up to 1300 nodes to local optimality in seconds. As for the Weak PCCP, the computation times are up to several minutes. The proposed algorithm is sufficiently efficient to be used as a subroutine when addressing Travelling Salesperson Problem with Circle Placement (TSP-CP), which combines the PCCP and the TSP-SD and is studied in [r11]. The TSP-CP fully addresses the motivating application for [c5] and [c6] in autonomous open-pit mining, where the goal is to determine simultaneously drill rig path and obstacle placement. In [r11], we propose a hybrid algorithm for the TSP-CP based on the approaches for TSP-SD and PCCP and extend it for the Dubins vehicle model [98].

Where to Place a Pile?

Miroslav Kulich[1], David Woller[1,2], Sarah Carmesin[3], Masoumeh Mansouri[3], and Libor Přeučil[1]

***Abstract*— When planning missions for autonomous machines in real-world scenarios, such as open-pit mining, painting, or harvesting, it is important to consider how the machines will alter the working environment during their operations. Traditional planning methods treat such changes, like piles built during drilling, as constraints given to the planner that depend on the machine's trajectory. The goal is to find a trajectory that satisfies these constraints. However, our approach formulates the planning problem as finding optimal positions for changes, such as piles, along the machine's trajectory. We propose a heuristic solver and provide extensive experimental evaluations.**

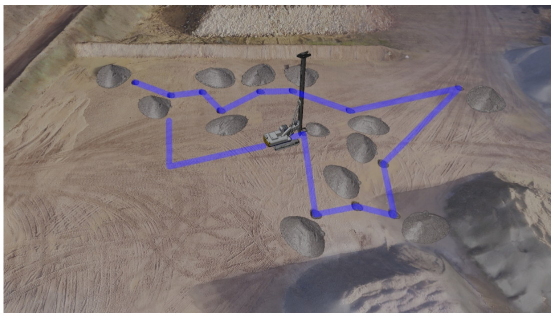


Fig. 1: Drilling scenario.

I. INTRODUCTION

With increasing levels of autonomy, robots are deployed in more and more complex scenarios. In a mining application, one or more drill machines operate in an open-pit mine to drill blast holes in predetermined targets. After the blast holes are drilled, they are filled with explosive material and detonated, and the ore is processed for mineral extraction. The drill machines can autonomously navigate to the targets, level themselves, drill, and retract. However, the drilling process creates piles of excess material around the hole, which must be cleared before the machine can navigate to the next target. The Drill Pattern Planning Problem (DP^3) [1] for a single drill machine or a fleet of these involves computing a time-optimal plan that ensures the machine(s) can reach each drill target, perform the defined operations, and move away from the target without colliding with obstacles, other machines, or the excess material created during the drilling process.

Mansouri et al. [1] propose a method for solving multi-vehicle DP^3 considering the dimensions of the machines, including the size of the dust guard and jacks used for leveling and the time required to perform each task. The authors break down the problem into sub-problems, identifies interdependency among the sub-problems, and interleaves reasoning within each sub-problem. The approach is further improved and defined as MVRP-DDO (Multi Vehicle Routing with Nonholonomic Constraints and Dense Dynamic Obstacles) in [2]. Carmesin et al. [3] introduce new variants of the Hamiltonian Cycle and Travelling Salesperson problems inspired by the open-pit mining application. Specifically, the authors assume dynamic graphs where edges are deleted or made untraversable depending on the already visited vertices. Besides formal definitions of the problems for such graphs, problems' properties are theoretically analyzed, and two solvers are proposed.

Another application where heavy vehicles are not allowed to pass already visited areas is autonomous harvesting. In the harvesting application, harvested areas limit the mobility of harvesting machines, hence affecting the reachability among the nodes representing areas to be harvested. Ullrich et al. [4] propose a graph-search planner that searches a directed graph representing the harvesting area. The designed cost function considers the number of passes through individual edges to tackle multiple passing of edges.

The aforementioned approaches assume the constraints caused by drilling/harvesting are predefined, and the planner can only affect the order in which they appear. In [1], [2], for example, piles are placed at the same positions as blast holes. Similarly, the set of edges to be deleted after visiting a vertex is predefined in [3], while edges are removed as they are traversed in [4]. Our approach is different. We assume that piles are built in the vicinity of blast holes, but their exact positions can vary, and the planning algorithm has to decide where to place them. Specifically, a trajectory for a drilling machine is given, and we ask the following questions:

- Can we place piles of a given radius so that the machine's trajectory is not obstructed by them?
- What is the largest radius for which the machine's trajectory is not obstructed by the piles?
- How should the piles be placed to ensure that the machine's path is not obstructed?

[1] Miroslav Kulich, David Woller, and Libor Přeučil are with Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Jugoslávských partyzánů 1580/3, 160 00 Prague, Czech Republic {miroslav.kulich, david.woller, libor.preucil}@cvut.cz

[2] David Woller is also with Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, 121 35 00 Prague, Czech Republic

[3] Sarah Carmesin and Masoumeh Mansouri are with School of Computer Science, University of Birmingham, Edgbaston, B15 2TT Birmingham, United Kingdom sxc1431@student.bham.ac.uk, m.mansouri@bham.ac.uk

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

2023 European Conference on Mobile Robots (ECMR) | 979-8-3503-0704-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ECMR59166.2023.10256330

We formulate two problems related to the above questions formally in Section II and propose a solver for both problems (Section III). The performance of the solver is extensively evaluated through experiments with a specially designed dataset, and the results are discussed in Section IV. The concluding remarks are presented in Section V.

II. PROBLEM FORMULATION

Let $P = \langle p_1, p_2, \dots p_n \rangle$ be a sequence of points in $\mathbb{R}^2$ forming a polygonal path, i.e., a curve consisting of line segments connecting the consecutive points. The *Path Conforming Circles Placement Problem* (PCCP) is to find a set of circles $\mathcal{K} = \{\kappa_i\}_{i \in I}$, where $I = \{1, \dots n\}$, and κ_i is a circle with center c_i and radius r_i, such that:

(C1) the radii of all circles are equal: $r_i = r \quad \forall i \in I$,
(C2) i^{th} point lies on i^{th} circle: $|p_i c_i| = r \quad \forall i \in I$,
(C3) intersection of any two circles is empty:
$\kappa_i \cap \kappa_j = \emptyset \quad \forall i, j \in I, i \neq j$,
(C4) intersection of any circle with the path is empty:
$\kappa_i \cap P = \emptyset \quad \forall i \in I$, and
(C5) r is maximal.

We say that κ_i is *associated* with p_i.

Let $head_k(P) = \langle p_1, p_2, \dots, p_{k-1} \rangle$ be a head of polygonal path $P = \langle p_1, p_2, \dots p_n \rangle$ and $tail_k(P) = \langle p_k, p_{k+1}, \dots, p_n \rangle$ its tail. The *Weak PCCP* (WPCCP) relaxes condition C4 by allowing for the nonempty intersection of the i^{th} circle with $head_i(P)$. The modified condition for the WCCP is thus:

(4*) $\kappa_i \cap tail_i(P) = \emptyset \quad \forall i \in \langle 1, n \rangle$

Examples of the problem instances and their solutions are shown in Fig. 2. As the WPCCP is less constrained, the maximum radius found for it is higher than the one for the PCCP.

Although the PCCP and WPCCP are new, we can take inspiration from a class of problems seeking the largest empty circle. The classic example of this class is the Largest Empty Circle Problem (LEC) which consists in finding the largest circle C centered in the convex hull of a set of points such that no point lies in the interior of the circle. Shamos [5] propose an algorithm solving the LEC based on an effective search of Voronoi diagrams. Thoussaint [6] subsequently corrected the algorithm showing that Shamos made a wrong assumption about the intersection of a convex hull with a Voronoi diagram, while [7] further improved the algorithm complexity to $O(n[h \log n])$, where n is the number of points and h is the number of convex hull edges. The query variant of the LEC is addressed in [8]. The aim is to preprocess the input points to identify the largest empty circle efficiently. Finally, Augustine et al. [9] address the constrained variant where the circle has to be centered on a given line. All the aforementioned formulations search for a *single* circle while we seek a set of circles. This makes our formulation novel and challenging.

III. APPROACH

In this section, we introduce a solver for both PCCP and WPCCP. We start with the description of a general structure which is the same for both problems. The next subsections III-A to III-C then detail the individual parts of the algorithm. Finally, subsection III-D introduces modifications for the WPCCP.

The algorithm shown in Alg. 1 is motivated by the bisection method [10]. It starts with the estimation of the lower and upper bound of the radius (lines 1 and 2). Then, the bounds are modified by the iterative procedure (lines 3-10). At each iteration, the interval between the bounds is split into two halves by computing the midpoint radius (line 4) and finding the optimal placement of circles with this radius fixed (line 5). If the found placement is valid, the lower bound is replaced by the radius (line 7), and the solution is stored. If the placement is invalid, the upper bound is replaced by the radius (line 8). The process stops when the upper and lower bound difference is below the predefined limit. The stored solution and radius are returned then (line 11).

Algorithm 1: Interval bisection algorithm for the PCCP/WPCCP.

```
Input: $\mathcal{C}$ – set of cells
1  $lb \leftarrow lower\_bound()$
2  $ub \leftarrow upper\_bound()$
3  while $(ub - lb) < \epsilon$ do
4      $radius \leftarrow \frac{lb+ub}{2}$
5      $(\mathcal{P}, valid) \leftarrow find\_placement(radius)$
6      if $valid$ then
7          $lb \leftarrow radius$
8          $(\mathcal{P}_{best}, radius_{best}) \leftarrow (\mathcal{P}, radius)$
9      else
10         $ub \leftarrow radius$
11 return $(\mathcal{P}_{best}, radius_{best})$
```

A. Upper Bound

Circle center c_i has to be closer to p_i than to any other point and line segment on P; otherwise, the circle would intersect P. The valuable tool for determining possible positions of circles' centers satisfying this condition is a Voronoi diagram (VD): the VD for a set of geometries (points and line segments in our case) is a partition of the plane into cells such that each cell contains exactly one input geometry and all points in the plane are closer to the geometry than to any other geometry. $VD(P)$, a Voronoi diagram of points and line segments can be computed by Fortune's sweepline algorithm in $O(n \log(n))$ time and use $O(n)$ space [11].

Fig. 3a shows the VD of a path from Fig. 2. Boundaries of Voronoi cells are formed by points equidistant from two or more input geometries. Boundary curves between two points or two segments are segments, while edges between a point and a segment are parabolic arcs. The center of the largest circle κ_i thus lies on a boundary of i^{th} cell, i.e. the cell VC_i containing i^{th} geometry. Moreover, maximal distances on boundary curves are reached at their end vertices [7].

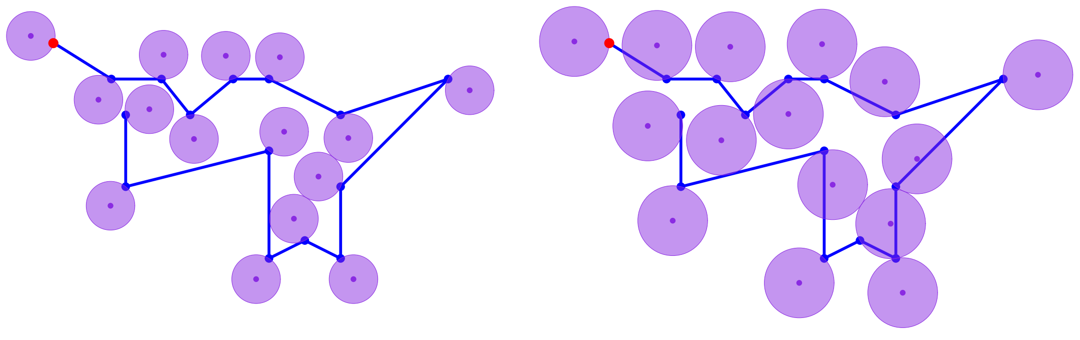

(a) (b)

Fig. 2: Example solutions of (a) the PCCP, and (b) the WPCCP. The paths (in blue) start at the red points.

Given $VC = \{VC_i\}_{i \in I}$ a set of Voronoi cells containing points on path P we can thus determine upper bound ub of (W)PCCP as:

$$ub = \min_{i \in I} \max_{v \in \Delta(VC_i)} |vp_i|, \tag{1}$$

where $\Delta(VC_i)$ is a set of VC_i's vertices and $|\cdot|$ is the Euclidean distance. In other words, the radius of the largest circle touching a point is the distance of the point to the most distant vertex on its Voronoi cell boundary (see Fig. 3b for an example). The upper bound is the smallest of these radii.

By contradiction, we prove there is no valid solution with a radius larger than ub. Assume p_m for which $|p_m c_m| = ub$. c_m is thus the vertex maximizing the distance in Eq. 1, and m is the cell index for which this maximum is minimal. The solution for radius $r > ub$ contains circle $C_m(\bar{c}_m, r)$ associated with p_m. As c_m is the farthest point of VC_m and $\bar{c}_m$ is farther from p_m than c_m, $\bar{c}_m$ lies outside VC_m. There is, therefore, a point or segment on the path closer to $\bar{c}_m$ than to c_m, i.e., circle $\kappa_m(\bar{c}_m, r)$ has a nonempty intersection with the path. The solution is thus invalid, which is a contradiction.

B. Lower Bound

We determine lower bound lb by finding *some* solution. Assume that a circle center for each cell VC_i lies on a ray α_i which bisects the angle formed by p_i and two edges of VC_i's boundary incident to p_i as shown in Fig. 3c. For each pair $p_i, p_j \in P, i \neq j$ we determine circles' centers c_i, c_j such that:

(1) the centers lie on the bisecting rays: $c_i \in \alpha_i$ and $c_j \in \alpha_j$,
(2) the points lie on the circles with the same radius r_{ij}: $|p_i c_i| = |p_j c_j| = r_{ij}$,
(3) the circles touch: $|c_i c_j| = 2r_{ij}$.

The above conditions lead to a quadratic equation for r_{ij} with one positive solution. Nevertheless, one or both centers can lie outside the individual cells, making the solution invalid. The valid radius $\bar{r}_{ij}$ is thus limited:

$$\bar{r}_{ij} = \min\{r_{ij}, |p_i x_i|, |p_j x_j|\}, \tag{2}$$

where x_i is the intersections of ray α_i with $\delta(VC_i)$, the boundary of VC_i). Similarly, $x_j = \alpha_j \cap \delta(VC_j)$.

The lower bound is the smallest valid radius of all pairs:

$$lb = \min_{i \in I, j \in I, i \neq j} \bar{r}_{ij}.$$

Circle center c_i is computed as the intersection of α_i with a circle centered in p_i with radius lb.

To prove the validity of the solution constructed by this procedure, we must ensure that the constraints C1–C4 from the problem formulation are satisfied. C1 and C2 are met trivially, and C4 holds as c_i lies in VC_i due to Eq. 2. C3 is proved by contradiction using the observation that given two circles touching the same point, a circle with a smaller radius is entirely inside a circle with a larger radius. This means that if the smaller circle intersects with another circle, the larger circle also intersects with the same circle. Assume now that two circles $\kappa_i(lb)$ and $\kappa_j(lb)$ with radii lb associated to some points c_i and c_j intersect. According to the above observation, $\kappa_i(lb)$ and $\kappa_j(\bar{r}_{ij})$ thus intersect as well as $\kappa_i(r_{ij})$ and $\kappa_j(r_{ij})$. This is not possible as $|c_i c_j| = 2r_{ij}$.

C. Placement for a fixed radius

In this section, we describe the algorithm which finds a valid placement of circles for given radius r or reports that such placement does not exist. We formulate this problem as a discrete optimization problem and solve it by a local search heuristic – an initially generated solution is iteratively improved by local optimization. Realize that validity of the initial solution is not guaranteed; thus, the search is done in the space of *all* solutions, not only valid ones. Invalid solutions are penalized in the designed objective function forcing the solver to find a valid one if it exists.

The algorithm shown in Alg. 2 starts by generating a set of valid circles' centers for each Voronoi cell of a point on path P (lines 1–2). Given cell VC_i, the set equals the intersection of the cell and a circle with radius r centered in p_i. To determine the intersection efficiently, the cell is split into sectors according to boundary edges as shown in Fig. 4a. The sectors are processed sequentially in clockwise order. Each sector is described by two angles – directions of rays starting

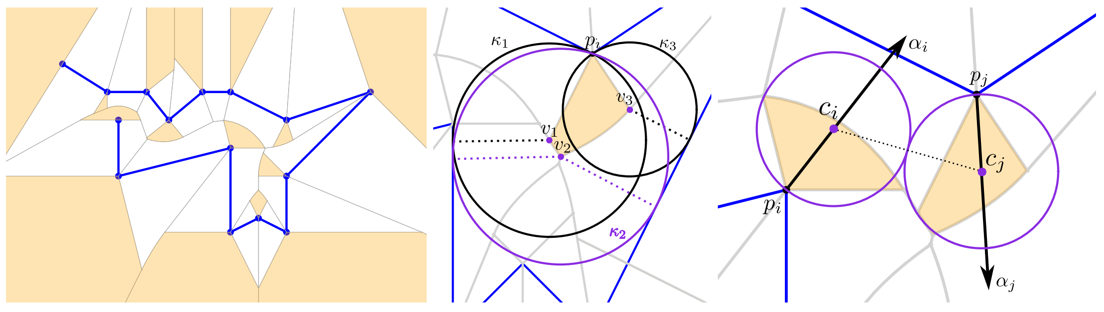


Fig. 3: (a) Voronoi diagram (in gray) of the blue path with highlighted cells of points (in orange). (b) Upper bound computation: Voronoi cell of p_i bounded by closed curve $p_1v_1v_2v_3p_1$, circles κ_1, κ_2, and κ_3 going through p_1 with centres v_1, v_2, and v_3, with κ_2 the largest one. The dotted lines connect the circles' centers with the points where the circles touch the path. (c) Lower bound computation: α_1 and α_2 are axes of cells associated with p_i and p_j respectively, s_i and s_j are centers of largest touching circles.

in p_i and passing through endpoints of the boundary edge e. A sequence of points on circle $\kappa(p_i, r)$ lying in the sector is generated for each sector. A point from the sequence lies in VC_i iff it is closer to p_i than to g, where g is a geometry (point or edge) that shares the boundary edge e with p_i.

The initial solution is generated next (line 3) by selecting one circle center from each $\mathcal{V}_i$. Preliminary experiments show that the selection does not influence the solution quality; we thus simply select the center randomly.

The iterative improvement is made in the loop in lines 4-11. In each iteration, points on the path are processed in a random order (lines 8-9). The randomness of the order is ensured by shuffling the indexes (line 7). When processing i^{th} point, all circles' centers are fixed except the i^{th} one and new c_i is selected from $\mathcal{V}_i$ that minimizes the designed objective function (line 9). It consists of two parts.

The first part penalizes candidate centers of circles having a non-empty intersection with other circles:

$$f^i_{int}(c) = \sum_{k \in I \setminus \{i\}} (\text{Area}(\kappa(c, r) \cap \kappa(c_k, r)) + \varepsilon_k), \qquad (3)$$

where

$$\varepsilon_k = \begin{cases} \varepsilon & \text{if } \kappa(c, r) \cap \kappa(c_k, r) \neq \emptyset \\ 0 & \text{otherwise,} \end{cases}$$

ε is a constant (see its meaning bellow), and Area is the area of circles' intersection.

Assume the optimal placement according to f_{int} in Fig. 4b for motivation of the second part. The intersection of κ_1 and κ_2 is small but nonempty. Center c_1 of circle κ_1 is the most left possible, i.e., in the best position. As the intersection of κ_2 and κ_3 is empty, $f^i_{int}(c_3) = 0$ and moving c_3 farther from κ_2 does not improve f^i_{int}. Moving c_2 farther from κ_1 shrinks κ_1–κ_2 intersection, but enlarges the intersection of κ_2 and κ_3 increasing f^i_{int} in total (notice that c_2 is in the optimal position). The solution is to move c_3 to provide space for moving c_2.

The second part of the objective function thus aims to penalize a circle (a center) with circles in its close vicinity even if it does not intersect them:

$$f^i_{dist}(c) = \sum_{k \in N} \gamma(\mu r - |cc_k|),$$

where $N = \{k | k \in I, k \neq i, |cc_k| \leq \mu r\}$, μ and γ are constants. μ specifies the size of the vicinity as the multiple of r and ensures that f^i_{dist} is non-negative. Even a tiny intersection of circles should be penalized more than many non-intersecting circles close to other circles. f^i_{int} should thus always dominate over f^i_{dist} which is the purpose of γ and ε from Eq. 3. We set $\mu = 2.2$, $\varepsilon = 10^{-5}$, and $\gamma = 10^{-10}$.

The new position of c_i minimizes the sum of the two parts:

$$c_i = \min_{c \in \mathcal{V}_i}(f^i_{int}(c) + f^i_{dist}(c)) \qquad (4)$$

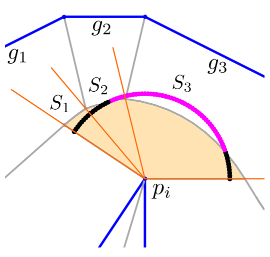
The validity of the solution is determined during the evaluation of f^i_{int}. Simply, the solution is valid if all intersections in Eq. 3 are empty.

We monitor whether the position of some center changed (line 10). If there is no such change during the processing of the entire path, the iterative process finishes (line 11), and the algorithm outputs the result (line 12).

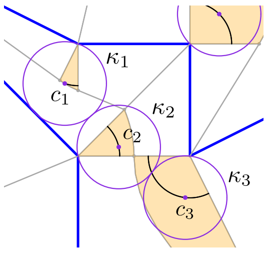
Unfortunately, the cost function in Eq. 4 is not convex, i.e., it can have many local minima. It is thus not guaranteed that Alg. 2 finds a global optimum or even a valid solution if it exists. On the other hand, the algorithm is relatively fast and stochastic. Therefore, we run Alg. 2 several times to increase the probability of a correct result.

D. Modifications for the WPCCP

The presented algorithm is the same for both PCCP and WPCCP, with one exception. The difference lies in the way VC_i , an area of possible positions of circles' centers, is determined for a given point p_i. While a Voronoi diagram (VD) of points and line segments forming the path is computed for the PCCP, and the set of possible centers for a



(a)



(b)

Fig. 4: (a) Determination of valid circle centers for a given radius. The orange lines delimit three sections S_1, S_2, and S_3 of point p_i. The black points are valid centers while the magenta ones are invalid. A center from S_k is valid iff it is closer to p_i than to geometry g_k. (b) Situation where the optimization of f_{int} does not find a valid solution.

Algorithm 2: Local search algorithm for a fixed radius.

```
Input: $VC$ – set of Voronoi cells of points on path $P$
       $r$ – radius
Output: $\langle valid, \Upsilon\rangle$, where $\Upsilon = \{c_i\}_{i\in I}$
       $valid$ – flag whether the solution is valid
       $\Upsilon = \{c_i\}_{i\in I}$ – set of circles' centers

1  foreach $i \in I$ do
2      $\mathcal{V}_i \leftarrow VC_i \cap C(p_i, r)$
3      $c_i \leftarrow$ random($\mathcal{V}_i$)
4  repeat
5      $modified \leftarrow false$
6      $valid \leftarrow true$
7      shuffle($I$)
8      foreach $i \in I$ do
9          $\langle valid_i, c_i\rangle \leftarrow$ optimal_center($\mathcal{V}_i$)
10         $valid \leftarrow valid \wedge valid_i$
           $modified \leftarrow modified \vee$ changed($c_i$)
11 until $!modified$
12 return $\langle valid, \Upsilon\rangle$
```

point is directly its Voronoi cell, the process for the WPCCP is more complex. The placement of a circle for p_i in the WPCCP is influenced only by points and segments not yet visited. This means that VC_i is a Voronoi cell of p_i in the VD constructed for unvisited points and segments. The naïve approach to determine VC_i for all points is to construct the VD of relevant geometries for each point and take its Voronoi cell. The time complexity of this approach is $O(n^2 \log n)$ as a single VD is constructed in $O(n \log n)$ time. Allen et al. [12] propose the VD construction algorithm that incrementally adds new geometries for which the VD is constructed. The amortized complexity of one such insertion is $O(\sqrt{n})$, and thus total complexity of constructing all VC_i's is $O(n\sqrt{n})$. However, we use the naïve approach in our implementation.

IV. Experimental Results

We evaluated the method's performance for both problems on two datasets we created for this purpose. The first dataset consists of 10 instances where the points are distributed evenly. Specifically, we generated hexagonal grids of various sizes, took vertices of these grids as cities, and solved the Travelling Salesman Problem for these cities by the Concorde solver [13]. The TSP solutions specify the instances `hexaXXX`, where `XXX` is the number of points in the instance.

The second dataset (instances `meshXXX`) is generated similarly, but the points are generated as vertices of a conforming constrained Delaunay triangulation (CCDT) generated by [14]. The CCDT distributes points in the plane randomly but tries to keep some minimal distance between them. Examples of both datasets are shown in Fig. 5 together with their solutions.

All experiments were performed within the same computational environment: a notebook with the Intel®Core i5-8250U CPU@1.6 Ghz. The algorithm has been implemented in C++. Twenty runs were run for each instance to provide statistically significant results.

The results are presented in Table I. Each row summarized values of 20 runs for each instance for both problems. lb and up are the lower and upper bounds, r, min, max stand for average, minimal, and maximal found valid radius, σ is the standard deviation of the radius, and $time$ is computational time in seconds.

Several observations can be made from the table. First, the mean radii are closer to the lower bounds than to the upper bounds. If we set $lb = 0\%$ and $ub = 100\%$, the radius is 10-35% for the PCCP, and 13-37% for the WPCCP. This suggests that lb is a better estimate of the optimal radius than ub.

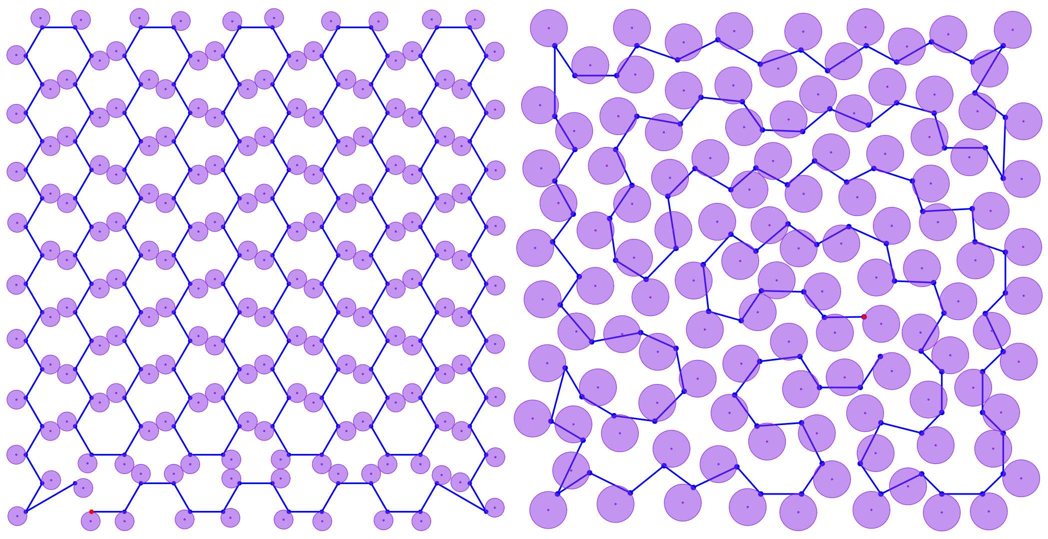
Second, the computational time is two orders higher for the WPCCP. This is caused mainly by the fact that Voronoi cells are larger than in the PCCP, and thus more centers have to be evaluated in Alg.2. Moreover, the convergence of this algorithm is slower due to a higher number of centers.

Although we run Alg. 2 twenty times, the solutions found by the solver differ as the values of σ show. The values are lower for the PCCP, meaning this problem is simpler to solve. Nevertheless, the deviations are relatively low, even for the WPCCP.

V. Conclusions

We study two problems inspired by an open-pit mining scenario. Contrary to the current approaches, we address the problems of where to place piles that do not obstruct a planned trajectory. Together with a heuristic solver for the problems, we provide upper and lower bounds for a given instance. The experimental evaluation shows that the solver finds good solutions for instances of hundreds of piles/circles in a reasonable time.

While the computational time for the PCCP is sufficient for real deployment, there is space for improvement for the WPCCP. We will thus focus on a more efficient generation of



(a) (b)

Fig. 5: Example instances and theirs solutions. (a) PCCP solution of `hexa180` (b) WPCCP solution of `mesh115`.

	PCCP							WPCCP						
problem	lb	ub	r	min	max	σ	time	lb	ub	r	min	max	σ	time
hexa180	15.00	30.00	17.31	17.31	17.31	0.00	0.50	18.02	103.90	30.66	28.75	31.39	0.64	48.82
hexa240	13.68	27.56	15.62	15.62	15.62	0.00	0.65	16.78	55.14	28.20	26.37	28.77	0.55	60.28
hexa308	12.43	25.00	14.35	14.35	14.35	0.00	0.85	15.02	86.45	24.88	24.14	25.65	0.41	99.93
hexa336	11.75	23.32	13.48	13.47	13.51	0.02	0.93	14.22	80.17	23.73	22.43	24.28	0.47	115.93
hexa416	10.64	21.00	12.23	12.23	12.23	0.00	1.14	13.05	57.95	21.32	20.50	22.00	0.46	134.56
hexa448	9.50	18.75	10.99	10.99	10.99	0.00	1.16	11.52	64.63	19.06	18.12	19.84	0.46	172.08
hexa540	9.50	18.99	10.96	10.96	10.96	0.00	1.50	11.43	65.77	18.63	18.00	19.50	0.41	213.87
hexa576	8.62	17.36	9.90	9.90	9.91	0.00	1.55	10.57	35.34	17.24	16.37	17.82	0.44	196.54
hexa836	7.50	15.00	8.66	8.66	8.66	0.00	2.16	9.04	40.41	14.76	13.93	15.41	0.41	338.74
hexa1144	6.48	13.00	7.37	7.37	7.37	0.00	3.05	7.84	34.95	12.43	11.96	12.92	0.27	413.71
mesh115	14.90	31.04	18.68	18.68	18.68	0.00	0.27	20.72	65.84	34.01	31.75	35.55	1.34	16.82
mesh244	9.25	19.23	10.47	10.29	10.72	0.14	0.37	12.17	38.87	20.22	19.57	20.90	0.49	35.75
mesh268	8.61	19.14	9.95	9.95	9.95	0.00	0.45	11.77	46.47	19.66	17.69	20.62	0.98	54.06
mesh293	7.87	17.87	8.87	8.75	8.96	0.08	0.45	10.98	38.87	17.55	17.07	17.84	0.22	41.52
mesh343	7.78	16.00	8.99	8.94	9.00	0.02	0.42	11.16	45.10	17.40	16.45	18.20	0.56	64.43
mesh374	6.99	14.15	8.76	8.65	8.85	0.06	0.47	9.49	31.50	17.40	16.32	18.11	0.42	66.10
mesh400	7.51	15.98	8.70	8.70	8.70	0.00	0.64	9.72	34.28	15.40	14.70	15.72	0.33	65.52
mesh449	7.06	11.78	8.68	8.66	8.70	0.01	0.45	9.00	26.10	15.32	14.86	15.61	0.18	87.37
mesh686	5.49	12.02	6.47	6.46	6.50	0.01	1.05	6.78	25.22	11.35	11.09	11.82	0.17	112.87
mesh1337	3.51	6.77	4.21	4.20	4.22	0.01	1.21	4.71	17.72	7.85	7.55	8.03	0.12	215.91

TABLE I: Experimental evaluation.

candidate centers in the future. Instead of generating them equidistantly, new candidates will be generated adaptively at promising positions based on the cost value of already evaluated centers.

As Voronoi diagrams can be generated for general geometries, a natural extension of the problem is to consider trajectories of other shapes. An interesting example is Dubin's car, for which optimal paths consist of straight lines and circular turns.

ACKNOWLEDGMENT

The research was supported by Czech Science Foundation Grant No. 23-05104S. The work of David Woller has also been supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/122/ OHK3/2T/13. Masoumeh Mansouri is a UK participant in Horizon Europe Project CONVINCE, and her work is supported by UKRI grant number 10042096.

REFERENCES

[1] M. Mansouri, H. Andreasson, and F. Pecora, "Hybrid reasoning for multi-robot drill planning in open-pit mines," *Acta Polytechnica*,

vol. 56, no. 1, pp. 47–56, 2016.

[2] M. Mansouri, F. Lagriffoul, and F. Pecora, “Multi vehicle routing with nonholonomic constraints and dense dynamic obstacles,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3522–3529.

[3] S. Carmesin, D. Woller, D. Parker, M. Kulich, and M. Mansouri, “The hamiltonian cycle and travelling salesperson problems with traversal-dependent edge deletion,” *SSRN Electronic Journal*, 2023. [Online]. Available: https://ssrn.com/abstract=4410404

[4] A. Ullrich, J. Hertzberg, and S. Stiene, “Ros-based path planning and machine control for an autonomous sugar beet harvester,” in *Proceedings of International Conference on Machine Control & Guidance, (MCG-2014)*, 2014.

[5] M. I. Shamos, “Computational geometry.” Ph.D. dissertation, Yale University, 1978.

[6] G. T. Toussaint, “Computing largest empty circles with location constraints,” *International Journal of Computer & Information Sciences*, vol. 12, pp. 347–358, 10 1983. [Online]. Available: https://link.springer.com/article/10.1007/BF01008046

[7] M. Schuster, “The largest empty circle problem,” in *Proceedings of the Class of 2008 Senior Conference*, 2008, pp. 28—-37.

[8] J. Augustine, S. Das, A. Maheshwari, S. C. Nandy, S. Roy, and S. Sarvattomananda, “Localized geometric query problems,” *Computational Geometry*, vol. 46, no. 3, pp. 340–357, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772112001198

[9] J. Augustine, B. Putnam, and S. Roy, “Largest empty circle centered on a query line,” *Journal of Discrete Algorithms*, vol. 8, no. 2, pp. 143–153, 2010, selected papers from the 3rd Algorithms and Complexity in Durham Workshop ACiD 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570866709000847

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, “Numerical recipes: The art of scientific computing,” in *Finding Roots of Equations*. Cambridge University Press, 1992, ch. 9.2, pp. 352–355.

[11] S. Fortune, “A sweepline algorithm for voronoi diagrams,” in *Proceedings of the Second Annual Symposium on Computational Geometry*, ser. SCG ’86. New York, NY, USA: Association for Computing Machinery, 1986, p. 313–322. [Online]. Available: https://doi.org/10.1145/10515.10549

[12] S. R. Allen, L. Barba, J. Iacono, and S. Langerman, “Incremental Voronoi Diagrams,” *Discrete & Computational Geometry*, vol. 58, no. 4, pp. 822–848, 2017. [Online]. Available: https://doi.org/10.1007/s00454-017-9943-2

[13] D. Applegate, R. Bixby, V. Chvatal, and W. J. Cook, “The Concorde tsp solver,” http://www.math.uwaterloo.ca/tsp/concorde.html, 2003, accessed: 2023-05-01.

[14] J. R. Shewchuk, “Delaunay refinement algorithms for triangular mesh generation,” *Computational Geometry*, vol. 22, no. 1, pp. 21–74, 2002, 16th ACM Symposium on Computational Geometry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772101000475

Chapter 9

Results and Discussion

In this chapter, we summarize the contributions of individual core publications and discuss future work perspectives. The presented research can be divided into two streams. In the first stream, represented by the core publications [c1]–[c3], we present problem-specific metaheuristic algorithms for various problems with permutative representation. These are real-world problems motivated by a practical robotic application or an international competition organized by the operations research community. In the second stream, represented by the core publications [c4]–[c6], we propose a generic metaheuristic solver for problems with permutative representation and present the follow-up research.

Problem-specific metaheuristic algorithms

In the first core publication [c1], presented in Chapter 3, we propose a path planning algorithm that ensures the accurate location of radiation sources in a highly specific robotic application. The main contributions lie in adapting an existing metaheuristic algorithm GLNS for various specifics of the application, especially exploiting prior knowledge of source positions and compensating for sensor limits by following a special type of maneuvers along circular arcs. The proposed approach is currently suitable only for robots capable of in-place rotation. A desirable extension is to plan with different vehicle models, with a focus on preserving the scalability of the proposed approach.

The second core publication [c2], presented in Chapter 4, is motivated by the IEEE WCCI competition on the EVRP [27]. In [c2], we propose a GRASP metaheuristic for the EVRP. This algorithm served as the basis for our VNS metaheuristic for EVRP, which finished 1^{st} in the competition. The main contribution lies in designing a highly scalable and computationally efficient metaheuristic algorithm for a novel variant of the VRP with practical relevance. In terms of future work, the most current goal is to finalize the publication of the final VNS algorithm [r10], which is in the review process for two years already, after a major and minor revision.

The last core publication in this stream, [c3] presented in Chapter 5, is motivated by another international competition - ROADEF Challenge 2020 [28]. The competition problem, formulated by the RTE company, is a complex optimization problem that addresses maintenance scheduling of a power transmission network. Our method advanced to the final competition phase and finished 2^{nd} out of 31 teams in the junior category and 8^{th} out of 74 in the overall ranking. The main contribution lies in adapting the ALNS metaheuristic for the maintenance scheduling problem and designing a large number of heuristics and local search operators tailored to the problem. Regarding future work, we also participated in the next competition run, ROADEF Challenge 2022 [28]. This time, we designed an ILS metaheuristic algorithm for the 3D truck loading problem, formulated by the Renault company. We finished 9^{th} out of 51 teams in the overall ranking and 2^{nd} out of 20 in the junior category again. The method was described in a diploma thesis [s14] and a journal publication is in preparation.

In summary, there is no shortage of novel, increasingly complex applications that require the design of custom metaheuristic algorithms. Although the underlying metaheuristics are well established in the literature, their efficient application requires exploiting problem-specific properties through the design of specialized components, which remains an academically interesting area. Within this thesis, we addressed several such problems with similar structure and designed algorithms that proved to belong between state of the art. The process of designing these algorithms, however, share common drawbacks, which are the required expertise and mainly design time, easily spanning several months.

Generic metaheuristic solver and its applications

Instead of developing yet another specialized solver for a single specific problem, our focus shifted to developing a more versatile tool, which could be rapidly deployed to a larger class of similarly structured problems and could be used either in an adequately challenging application or for prototyping a specialized solver. The second stream of research presented in this thesis begins with the core publication [c4] in Chapter 6, in which we propose a generic metaheuristic solver for problems with permutative representation and a formalism to define such problems easily. The contribution lies in designing a modular metaheuristic solver that can be applied to a wide class of optimization problems and does not require the implementation of specialized low-level components. In terms of scalability, the solver performs better than a similarly general IP Gurobi optimizer, but naturally lacks behind specialized metaheuristic solvers. The first version of the solver [c4], initially implemented in [s12], was based on single-solution metaheuristics relying on local search - VNS and ILS. In the follow-up research, currently presented in [s13], it proved to be sufficiently scalable but struggling with problems richer in constraints. An alternative approach based on Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) was proposed and will be further developed and published in the future.

The proposed generic solver was also deployed in the core publication [c5] described in Chapter 7. It was applied to four newly proposed optimization problems on self-deleting graphs: HCP-SD, TSP-SD and their relaxed variants, all of which are motivated by an application in autonomous mining. The main focus of the paper is on proposing the concept of self-deleting graphs, the aforementioned problems, and analyzing their properties both theoretically and statistically. The generic metaheuristic solver was successfully deployed to the HCP-SD and TSP-SD in a matter of days. It was then used in combination with a custom exact construction method for statistical analysis on several large datasets of small to medium-sized instances and was able to solve instances with more than 1000 nodes. Future work is linked to the following core publication.

The last core publication [c6], presented in Chapter 8, addresses a subproblem of the motivating mining application, which was not tackled in [c5]. We formulated the Path-Conforming Circle Placement Problem (PCCP), where the goal is to place circular obstacles of maximal radius along a given tour. In the motivating mining application, PCCP should be solved simultaneously with TSP-SD, but it cannot be captured by the formalism proposed in [c4]. Therefore, we proposed a specialized solver for the PCCP in [c6] and also derived theoretical bounds on the cost of the solution. In the subsequent work, which is currently under review in [r11], we formulate the Travelling Salesperson Problem with Circle Placement (TSP-CP), which combines PCCP with TSP-SD and we propose a new specialized solver that combines the approaches from [c5] and [c6]. We also propose a new specialized solver for the TSP-SD in [r11], which outperforms the generic solver used in [c5] in scalability and solution quality, although the latter is better only by 5% on average, given the same computation time.

In summary, we proposed a generic metaheuristic solver for problems with permutative representation and successfully applied it in follow-up research work. Given that the generic solver can be deployed in days, whereas developing the specialized one may require several months, it turns out that the generic solver can be a reasonable first choice in many practical applications.

Chapter 10

Conclusion

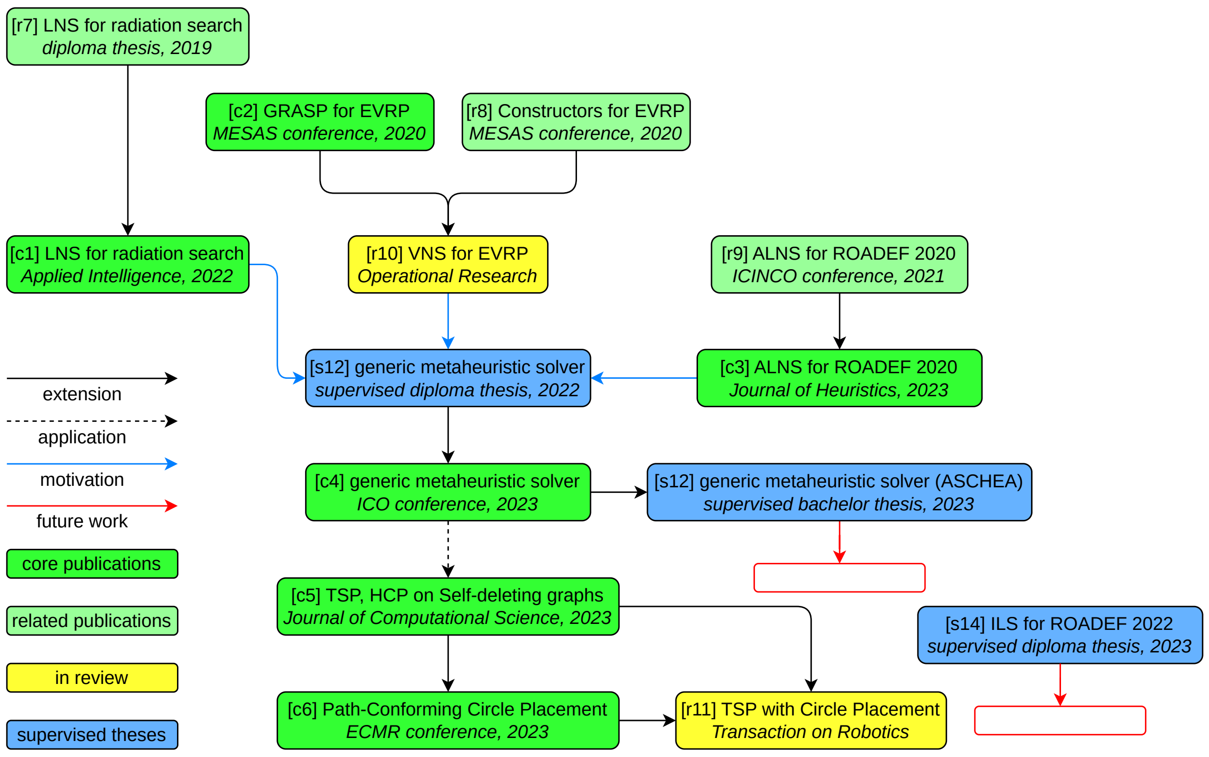
This thesis is presented as a compilation of six core publications, three of which are Q2 journal articles, in accordance with requirements of the Faculty of Electrical Engineering CTU doctoral study code [99]. It addresses the design of metaheuristic algorithms for combinatorial optimization problems with permutative representation. In the first three core publications, we propose state of the art metaheuristic algorithms for various problems from the studied class that are motivated either by a robotic application or by international competitions. In the remaining three core publications, we propose a generic metaheuristic solver for problems with permutative representation, apply this solver to several newly formulated variants of the TSP and HCP on self-deleting graphs and present some follow-up research in the motivating application, which is autonomous mining. The generic solver offers scalability that exceeds general-purpose IP solvers and is comparable to specialized metaheuristic algorithms.

Future work of the presented research lies in multiple directions. In terms of publication work, two journal articles dedicated to specialized metaheuristic algorithms for the EVRP [r10] and TSP-CP [r11] are currently in the review process. Another algorithm, successful in ROADEF Challenge 2022 [28], was recently presented in a supervised diploma thesis [s14] and its publication is also planned. Regarding research work, an extension of the generic metaheuristic solver enabling more robust constraint handling is under development and partially presented in a supervised bachelor's thesis [s13].

Chapter A

Author's publications

All the author's publications are listed below. Each citation is supplemented with the author's contribution percentage and the number of citations according to Web of Science, Scopus, and Google Scholar. In case of journal publications, the impact factor and journal quartile ranking according to the Journal Citation Reports are also stated. The thesis core publications are referenced as [c*], other author's thesis-related publications are referenced as [r*], and the thesis supervised by the author as [s*]. The data presented are current to March 22, 2024. Finally, the logical continuity of all publications is visualized in Figure A.1.



[r7] LNS for radiation search
diploma thesis, 2019
[c2] GRASP for EVRP
MESAS conference, 2020
[r8] Constructors for EVRP
MESAS conference, 2020
[c1] LNS for radiation search
Applied Intelligence, 2022
[r10] VNS for EVRP
Operational Research
[r9] ALNS for ROADEF 2020
ICINCO conference, 2021
extension
[s12] generic metaheuristic solver
supervised diploma thesis, 2022
[c3] ALNS for ROADEF 2020
Journal of Heuristics, 2023
application
motivation
[c4] generic metaheuristic solver
ICO conference, 2023
[s12] generic metaheuristic solver (ASCHEA)
supervised bachelor thesis, 2023
future work
core publications
[c5] TSP, HCP on Self-deleting graphs
Journal of Computational Science, 2023
related publications
[s14] ILS for ROADEF 2022
supervised diploma thesis, 2023
in review
[c6] Path-Conforming Circle Placement
ECMR conference, 2023
[r11] TSP with Circle Placement
Transaction on Robotics
supervised theses

Figure A.1: Logical continuity of all author's publications and supervised thesis

A.1 Thesis core publications

Journal Publications

[c1] **Woller, D.**, Kulich, M., "Path planning algorithm ensuring accurate localization of radiation sources", *Applied Intelligence*, pp. 1–23, 2022, ISSN: 15737497. DOI: 10.1007/S10489-021-02941-Y, **70% contribution, IF 5.3 (Q2 in Computer Science, Artificial Intelligence), citations: 1 in Web of Science, 1 in Scopus, 2 in Google Scholar.**

[c3] **Woller, D.**, Rada, J., Kulich, M., "The ALNS metaheuristic for the transmission maintenance scheduling", *Journal of Heuristics*, vol. 29, no. 2-3, pp. 349–382, 2023. DOI: 10.1007/s10732-023-09514-x, **70% contribution, IF 2.7 (Q2 in Computer Science, Theory & Methods), citations: 1 in Web of Science, 1 in Scopus, 1 in Google Scholar.**

[c5] Carmesin, S., **Woller, D.**, Parker, D., Kulich, M., Mansouri, M., "The Hamiltonian Cycle and Travelling Salesperson problems with traversal-dependent edge deletion", *Journal of Computational Science*, vol. 74, p. 102 156, 2023, ISSN: 1877-7503. DOI: 10.1016/j.jocs.2023.102156, **20% contribution, IF 3.3 (Q2 in Computer Science, Theory & Methods), citations: 0 in Web of Science, 1 in Scopus, 1 in Google Scholar.**

Conference publications

[c2] **Woller, D.**, Kozák, V., Kulich, M., "The GRASP Metaheuristic for the Electric Vehicle Routing Problem", English, in *Modelling and Simulation for Autonomous Systems*, ser. 1, Cham, CH: Springer, 2020. DOI: 10.1007/978-3-030-70740-8_12, **50% contribution, citations: 0 in Web of Science, 1 in Scopus, 4 in Google Scholar.**

[c4] **Woller, D.**, Hrazdíra, J., Kulich, M., "Metaheuristic Solver for Problems with Permutative Representation", in *Intelligent Computing & Optimization*, Springer International Publishing, 2023, pp. 42–54, ISBN: 978-3-031-19958-5. DOI: 10.1007/978-3-031-19958-5_5, **50% contribution, citations: 0 in Web of Science, 0 in Scopus, 1 in Google Scholar.**

[c6] Kulich, M., **Woller, D.**, Carmesin, S., Mansouri, M., Přeučil, L., "Where to Place a Pile?", in *2023 European Conference on Mobile Robots (ECMR)*, IEEE, 2023. DOI: 10.1109/ecmr59166.2023.10256330, **20% contribution, citations: 0 in Web of Science, 0 in Scopus, 0 in Google Scholar.**

A.2 Related publications

[r7] **Woller, D.**, *Search for sources of gamma radiation*, master thesis, Czech Technical University in Prague, 2019. https://dspace.cvut.cz/handle/10467/83422.

[r8] Kozák, V., **Woller, D.**, Vávra, V., Kulich, M., "Initial Solution Constructors for Capacitated Green Vehicle Routing Problem", English, in *Modelling and Simulation for Autonomous Systems*, ser. 1, Cham, CH: Springer, 2020. DOI: 10.1007/978-3-030-70740-8_16, **30% contribution, citations: 0 in Web of Science, 1 in Scopus, 2 in Google Scholar.**

[r9] **Woller, D.**, Kulich, M., "The ALNS Metaheuristic for the Maintenance Scheduling Problem", in *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO*, INSTICC, SciTePress, 2021, pp. 156–164, ISBN: 978-989-758-522-7. DOI: 10.5220/0010552101560164, **70% contribution, citations: 1 in Web of Science, 1 in Scopus, 3 in Google Scholar.**

[r10] **Woller, D.**, Kozák, V., Kulich, M., "Variable Neighborhood Search for the Electric Vehicle Routing Problem", **under review** at *Operational Research - An International Journal (ORIJ).*

[r11] **Woller, D.**, Mansouri, M., Kulich, M., "Making a Mess and Getting Away with it: Traveling Salesperson Problem with Circle Placement for Dubins Vehicles", **under review** at *IEEE Transactions on Robotics (T-RO).*

A.3 Supervised theses

[s12] Hrazdíra, J., *Metaheuristic Algorithms for Optimization Problems Sharing Representation*, master thesis, Czech Technical University in Prague, 2022. https://dspace.cvut.cz/handle/10467/102112.

[s13] Pažout, D., *Evolutionary Algorithms for Optimization Problems with Permutative Representation*, bachelor thesis, Czech Technical University in Prague, 2023. https://dspace.cvut.cz/handle/10467/108749.

[s14] Hromada, T., *ROADEF Challenge 2022: Optimization of truck fleet loading*, master thesis, Czech Technical University in Prague, 2023. https://dspace.cvut.cz/handle/10467/109463.

Bibliography

[15] Schrijver, A., "On the History of Combinatorial Optimization (Till 1960)", in *Discrete Optimization*, ser. Handbooks in Operations Research and Management Science, K. Aardal, G. Nemhauser, and R. Weismantel, Eds., vol. 12, Elsevier, 2005, pp. 1–68. DOI: 10.1016/S0927-0507(05)12001-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705120015.

[16] Kuhn, H. W., "Variants of the Hungarian method for assignment problems", *Naval research logistics quarterly*, vol. 3, no. 4, pp. 253–258, 1956.

[17] Chandrashekar, G., Sahin, F., "A survey on feature selection methods", *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014, 40th-year commemorative issue, ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2013.11.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790613003066.

[18] Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., "The vehicle routing problem: State of the art classification and review", *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016, ISSN: 0360-8352. DOI: 10.1016/j.cie.2015.12.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835215004775.

[19] Gunjan, A., Bhattacharyya, S., "A brief review of portfolio optimization techniques", *Artificial Intelligence Review*, vol. 56, no. 5, 3847–3886, Sep. 2022, ISSN: 1573-7462. DOI: 10.1007/s10462-022-10273-7. [Online]. Available: http://dx.doi.org/10.1007/s10462-022-10273-7.

[20] Macharet, D. G., Campos, M. F. M., "A survey on routing problems and robotic systems", *Robotica*, vol. 36, no. 12, 1781–1803, 2018. DOI: 10.1017/S0263574718000735.

[21] Wolsey, L., "Well-Solved Problems", in *Integer Programming*, John Wiley & Sons, Ltd, 2020, pp. 43–62. DOI: 10.1002/9781119606475.ch3. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/9781119606475.ch3.

[22] Christophides, N. "Worst-case Analysis of a New Heuristic for the Travelling Salesman Problem", in *Proceedings of Symposium on New Directions and Recent Results in Algorithms and Complexity*, 1976. [Online]. Available: https://ci.nii.ac.jp/naid/10006934449.

[23] *Gurobi Optimizer*, https://www.gurobi.com/solutions/gurobi-optimizer/, Accessed: March 22, 2024.

[24] *IBM ILOG CPLEX Optimizer*, https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer, Accessed: March 22, 2024.

[25] *FICO Xpress Optimization*, https://www.fico.com/en/products/fico-xpress-optimization, Accessed: March 22, 2024.

[26] Carles, C., Esquirol, Y., Turuban, M., "Residential proximity to power lines and risk of brain tumor in the general population", *Environmental Research*, vol. 185, p. 109 473, 2020, ISSN: 0013-9351. DOI: 10.1016/j.envres.2020.109473. [Online]. Available: http://dx.doi.org/10.1016/j.envres.2020.109473.

[27] Mavrovouniotis, M., *IEEE WCCI 2020: Competition on Electric Vehicle Routing Problem*, https://mavrovouniotis.github.io/EVRPcompetition2020/, Accessed: March 22, 2024.

[28] *ROADEF Challenge website*, https://www.roadef.org/challenge/, Accessed: March 22, 2024.

[29] Salienko, E., *Mining drilling machine stock photo*, https://www.shutterstock.com/cs/image-photo/mining-drilling-machine-drills-wells-slate-1768699433, Accessed: March 22, 2024, 2023.

[30] Sörensen, K., Sevaux, M., Glover, F., "A history of metaheuristics", in *Handbook of Heuristics*, vol. 2-2, Springer, 2018, pp. 791–808, ISBN: 9783319071244. DOI: 10.1007/978-3-319-07124-4_4. arXiv: 1704.00853. [Online]. Available: https://doi.org/10.1007/978-3-319-07124-4_4.

[31] Sollin, M, "La trace de canalisation", *Programming, Games, and Transportation Networks*, 1965.

[32] Prim, R. C., "Shortest Connection Networks And Some Generalizations", *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957, ISSN: 15387305. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

[33] Rechenberg, I., "Evolution Strategy: Nature's Way of Optimization", in *Optimization: Methods and Applications, Possibilities and Limitations*, Springer, Berlin, Heidelberg, 1989, pp. 106–126. DOI: 10.1007/978-3-642-83814-9_6. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-83814-9_6.

[34] Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.

[35] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, pp. 671–680, 1983, ISSN: 00368075. DOI: 10.1126/science.220.4598.671. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[36] Glover, F., "Future paths for integer programming and links to artificial intelligence", *Computers and Operations Research*, vol. 13, no. 5, pp. 533–549, 1986, ISSN: 03050548. DOI: 10.1016/0305-0548(86)90048-1.

[37] Muklason, A., Irianti, R. G., Marom, A., "Automated Course Timetabling Optimization Using Tabu-Variable Neighborhood Search Based Hyper-Heuristic Algorithm", *Procedia Computer Science*, vol. 161, pp. 656–664, 2019, The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia, ISSN: 1877-0509. DOI: 10.1016/j.procs.2019.11.169. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919318800.

[38] Colorni, A., Dorigo, M., Maniezzo, V., "Distributed Optimization by ant colonies", in *Proceedings of the First European Conference on Artificial Life*, 1991, pp. 134–142.

[39] Jia, Y. H., Mei, Y., Zhang, M., “A Bilevel Ant Colony Optimization Algorithm for Capacitated Electric Vehicle Routing Problem”, *IEEE Transactions on Cybernetics*, 2021, ISSN: 21682275. DOI: 10.1109/TCYB.2021.3069942.

[40] Baxter, J., “Local optima avoidance in depot location”, *Journal of the Operational Research Society*, vol. 32, no. 9, pp. 815–819, 1981, ISSN: 14769360. DOI: 10.1057/jors.1981.159. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1057/jors.1981.159.

[41] Vasquez, M., Buljubasic, M., Hanafi, S., “An efficient scenario penalization matheuristic for a stochastic scheduling problem”, *Journal of Heuristics*, vol. 29, no. 2–3, 383–408, 2023, ISSN: 1572-9397. DOI: 10.1007/s10732-023-09513-y. [Online]. Available: http://dx.doi.org/10.1007/s10732-023-09513-y.

[42] Gu, H., Lam, H. C., Pham, T. T. T., Zinder, Y., “Heuristics and meta-heuristic to solve the ROADEF/EURO challenge 2020 maintenance planning problem”, *Journal of Heuristics*, vol. 29, no. 1, 139–175, 2023, ISSN: 1572-9397. DOI: 10.1007/s10732-022-09508-1. [Online]. Available: http://dx.doi.org/10.1007/s10732-022-09508-1.

[43] Mladenović, N., Hansen, P., “Variable neighborhood search”, *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997, ISSN: 03050548. DOI: 10.1016/S0305-0548(97)00031-2.

[44] *IMR: Intelligent and Mobile Robotics Laboratory, CIIRC, CTU in Prague*, http://imr.ciirc.cvut.cz/, Accessed: March 22, 2024.

[45] Zahradka, D., Andreychuk, A., Kulich, M., Yakovlev, K., “Quality Analysis of Multi-Agent Multi-Item Pickup and Delivery Solutions Using a Decoupled Approach”, *IFAC-PapersOnLine*, vol. 55, no. 38, pp. 61–66, 2022, 13th IFAC Symposium on Robot Control SYROCO 2022, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2023.01.134. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896323001416.

[46] Zahrádka, D., Kubišta, D., Kulich, M., “Solving Robust Execution of Multi-Agent Pathfinding Plans as a Scheduling Problem (PlanRob 2023 workshop at ICAPS)”,

[47] Mikula, J., Kulich, M., “Solving the traveling delivery person problem with limited computational time”, *Central European Journal of Operations Research*, vol. 30, no. 4, 1451–1481, Jan. 2022, ISSN: 1613-9178. DOI: 10.1007/s10100-021-00793-y. [Online]. Available: http://dx.doi.org/10.1007/s10100-021-00793-y.

[48] Feo, T. A., Resende, M. G., “Greedy Randomized Adaptive Search Procedures”, *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995, ISSN: 09255001. DOI: 10.1007/BF01096763. [Online]. Available: https://link.springer.com/article/10.1007/BF01096763.

[49] Parreño, F., Parreño-Torres, C., Alvarez-Valdes, R., “A matheuristic algorithm for the maintenance planning problem at an electricity transmission system operator”, *Expert Systems with Applications*, vol. 230, p. 120 583, 2023, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120583. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423010850.

[50] Zahradka, D., Mikula, J., Kulich, M., "A Metaheuristic Approach for Inspection and Reconnaissance of Organized Areas", in *Modelling and Simulation for Autonomous Systems*, J. Mazal, A. Fagiolini, P. Vašík, *et al.*, Eds., Cham: Springer International Publishing, 2023, pp. 44–63, ISBN: 978-3-031-31268-7.

[51] Shaw, P., "Using constraint programming and local search methods to solve vehicle routing problems", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1520, Springer, Berlin, Heidelberg, 1998, pp. 417–431, ISBN: 3540652248. DOI: `10.1007/3-540-49481-2_30`.

[52] Stützle, T., Ruiz, R., "Iterated greedy", in *Handbook of Heuristics*, vol. 1-2, Springer, Cham, 2018, pp. 547–577, ISBN: 9783319071244. DOI: `10.1007/978-3-319-07124-4_10`. [Online]. Available: `https://link.springer.com/referenceworkentry/10.1007/978-3-319-07124-4_10`.

[53] Smith, S. L., Imeson, F., "GLNS: An Effective Large Neighborhood Search Heuristic for the Generalized Traveling Salesman Problem", *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.

[54] Mikula, J., Kulich, M., "Towards a Continuous Solution of the d-Visibility Watchman Route Problem in a Polygon With Holes", *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5934–5941, 2022. DOI: `10.1109/LRA.2022.3159824`.

[55] Kulich, M., Vidašič, J., Mikula, J., "On the Travelling Salesman Problem with Neighborhoods in a Polygonal World", in *Robotics in Natural Settings*, J. M. Cascalho, M. O. Tokhi, M. F. Silva, A. Mendes, K. Goher, and M. Funk, Eds., Cham: Springer International Publishing, 2023, pp. 334–345, ISBN: 978-3-031-15226-9.

[56] Sörensen, K., "Metaheuristics-the metaphor exposed", *International Transactions in Operational Research*, vol. 22, no. 1, pp. 3–18, 2015, ISSN: 14753995. DOI: `10.1111/itor.12001`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/full/10.1111/itor.12001`.

[57] Moscato, P., Cotta, C., "An accelerated introduction to memetic algorithms", in *International Series in Operations Research and Management Science*, vol. 272, Springer New York LLC, 2019, pp. 275–309. DOI: `10.1007/978-3-319-91086-4_9`.

[58] Fischetti, M., Fischetti, M., "Matheuristics", in *Handbook of Heuristics*, Springer, Cham, 2016, pp. 1–33. DOI: `10.1007/978-3-319-07153-4_14-1`. [Online]. Available: `https://link.springer.com/referenceworkentry/10.1007/978-3-319-07153-4_14-1`.

[59] Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J. R., "A Classification of Hyper-heuristic Approaches", in *Handbook of Metaheuristics*, Springer, Boston, MA, 2010, pp. 449–468. DOI: `10.1007/978-1-4419-1665-5_15`. [Online]. Available: `https://link.springer.com/chapter/10.1007/978-1-4419-1665-5_15`.

[60] Dantzig, G. B., Orden, A., Wolfe, P., "The generalized simplex method for minimizing a linear form under linear inequality restraints", *Pacific Journal of Mathematics*, vol. 5, no. 2, pp. 183–195, 1955, ISSN: 00308730. DOI: `10.2140/pjm.1955.5.183`.

[61] Dikin, I. I., "Iterative solution of problems of linear and quadratic programming", English, *Soviet Mathematics - Doklady*, vol. 8, pp. 674–675, 1967, ISSN: 0197-6788.

[62] Spielman, D. A., Teng, S. H., "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time", *Journal of the ACM*, vol. 51, no. 3, pp. 385–463, 2004, ISSN: 00045411. DOI: `10.1145/990308.990310`. arXiv: `0111050 [cs]`. [Online]. Available: `https://arxiv.org/abs/cs/0111050v7`.

[63] Karmarkar, N., "A new polynomial-time algorithm for linear programming", *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984, ISSN: 02099683. DOI: `10.1007/BF02579150`.

[64] Land, A. H., Doig, A. G., "An Automatic Method of Solving Discrete Programming Problems", *Econometrica*, vol. 28, no. 3, p. 497, 1960, ISSN: 00129682. DOI: `10.2307/1910129`.

[65] Gilmore, P. C., Gomory, R. E., "A Linear Programming Approach to the Cutting-Stock Problem", *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961, ISSN: 0030-364X. DOI: `10.1287/opre.9.6.849`.

[66] Mitchell, J. E., "Branch-and-Cut Algorithms for Combinatorial Optimization Problems", *Handbook of Applied Optimization*, pp. 65–77, 2002.

[67] Atamtürk, A., Nemhauser, G. L., Savelsbergh, M. W., "Conflict graphs in solving integer programming problems", *European Journal of Operational Research*, vol. 121, no. 1, pp. 40–55, 2000, ISSN: 03772217. DOI: `10.1016/S0377-2217(99)00015-6`.

[68] Sherali, H. D., Glover, F., "Higher-order cover cuts from zero-one knapsack constraints augmented by two-sided bounding inequalities", *Discrete Optimization*, vol. 5, no. 2, pp. 270–289, 2008, ISSN: 15725286. DOI: `10.1016/j.disopt.2007.02.002`.

[69] Letchford, A. N., "On Disjunctive Cuts for Combinatorial Optimization", *Journal of Combinatorial Optimization*, vol. 5, no. 3, pp. 299–315, 2001, ISSN: 13826905. DOI: `10.1023/A:1011493126498`. [Online]. Available: `https://link.springer.com/article/10.1023/A:1011493126498`.

[70] Gu, Z., Nemhauser, G. L., Savelsbergh, M. W., "Lifted flow cover inequalities for mixed 0-1 integer programs", *Mathematical Programming, Series B*, vol. 85, no. 3, pp. 439–467, 1999, ISSN: 00255610. DOI: `10.1007/s101070050067`. [Online]. Available: `https://link.springer.com/article/10.1007/s101070050067`.

[71] Marchand, H., Martin, A., Weismantel, R., Wolsey, L., "Cutting planes in integer and mixed integer programming", *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 397–446, 2002, ISSN: 0166218X. DOI: `10.1016/S0166-218X(01)00348-1`.

[72] Ausiello, G., Marchetti-Spaccamela, A., Crescenzi, P., Gambosi, G., Protasi, M., Kann, V., "Approximation Preserving Reductions", in *Complexity and Approximation*, Springer, Berlin, Heidelberg, 1999, pp. 253–286. DOI: `10.1007/978-3-642-58412-1_8`. [Online]. Available: `https://link.springer.com/chapter/10.1007/978-3-642-58412-1_8`.

[73] Helsgaun, K., "An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems", Roskilde University, Tech. Rep., 2017. DOI: `10.13140/RG.2.2.25569.40807`.

[74] Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., "A unified solution framework for multi-attribute vehicle routing problems", *European Journal of Operational Research*, vol. 234, no. 3, pp. 658–673, 2014, ISSN: 03772217. DOI: 10.1016/j.ejor.2013.09.045.

[75] Whitley, D., Yoo, N.-W., "Modeling Simple Genetic Algorithms for Permutation Problems", in *Foundations of Genetic Algorithms*, L. D. WHITLEY and M. D. VOSE, Eds., vol. 3, Elsevier, 1995, pp. 163–184. DOI: 10.1016/B978-1-55860-356-1.50013-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781558603561500133.

[76] Koohestani, B., "A crossover operator for improving the efficiency of permutation-based genetic algorithms", *Expert Systems with Applications*, vol. 151, p. 113 381, 2020, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2020.113381. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420302050.

[77] Mehdi, M., "Parallel Hybrid Optimization Methods for Permutation Based Problems", Ph.D. dissertation, Université des Sciences et Technologie de Lille - Lille I, 2011. [Online]. Available: https://www.researchgate.net/publication/281015427_PARALLEL_HYBRID_OPTIMIZATION_METHODS_FOR_PERMUTATION_BASED_PROBLEMS.

[78] Dreo, J., Liefooghe, A., Verel, S., "Paradiseo: From a modular framework for evolutionary computation to the automated design of metaheuristics: 22 years of Paradiseo", in *GECCO 2021 Companion - Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion*, Association for Computing Machinery, Inc, 2021, pp. 1522–1530, ISBN: 9781450383516. DOI: 10.1145/3449726.3463276. arXiv: 2105.00420.

[79] Parejo, J. A., Ruiz-Cortés, A., Lozano, S., Fernandez, P., "Metaheuristic optimization frameworks: A survey and benchmarking", *Soft Computing*, vol. 16, no. 3, pp. 527–561, 2012, ISSN: 14327643. DOI: 10.1007/s00500-011-0754-8. [Online]. Available: https://link.springer.com/article/10.1007/s00500-011-0754-8.

[80] Scott, E. O., Luke, S., "ECJ at 20: Toward a general metaheuristics toolkit", in *GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion*, ACM, 2019, pp. 1391–1398, ISBN: 978145036-7486. DOI: 10.1145/3319619.3326865. [Online]. Available: https://doi.org/10.1145/3319619.3326865.

[81] Hadka, D., Reed, P. M., Simpson, T. W., "Diagnostic assessment of the borg MOEA for many-objective product family design problems", in *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 2012, ISBN: 9781467315098. DOI: 10.1109/CEC.2012.6256466.

[82] De Beukelaer, H., Davenport, G. F., De Meyer, G., Fack, V., "JAMES: An object-oriented Java framework for discrete optimization using local search metaheuristics", *Software - Practice and Experience*, vol. 47, no. 6, pp. 921–938, 2017, ISSN: 1097024X. DOI: 10.1002/spe.2459. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.2459.

[83] *CEITEC: Central European Institute of Technology, VUT in Brno*, https://www.ceitec.eu/cybernetics-and-robotics/rg390, Accessed: March 22, 2024.

[84] Lazna, T., Fisera, O., Kares, J., Zalud, L., "Localization of ionizing radiation sources via an autonomous robotic system", *Radiation Protection Dosimetry*, vol. 186, no. 2–3, 249–256, 2019, ISSN: 1742-3406. DOI: `10.1093/rpd/ncz213`. [Online]. Available: `http://dx.doi.org/10.1093/rpd/ncz213`.

[85] Gabrlik, P., Lazna, T., Jilek, T., Sladek, P., Zalud, L., "An automated heterogeneous robotic system for radiation surveys: Design and field testing", *Journal of Field Robotics*, vol. 38, no. 5, 657–683, 2021, ISSN: 1556-4967. DOI: `10.1002/rob.22010`. [Online]. Available: `http://dx.doi.org/10.1002/rob.22010`.

[86] Pop, P. C., Cosma, O., Sabo, C., Sitar, C. P., "A comprehensive survey on the generalized traveling salesman problem", *European Journal of Operational Research*, 2023, ISSN: 0377-2217. DOI: `10.1016/j.ejor.2023.07.022`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0377221723005581`.

[87] *Variable Neighborhood Search for the Electric Vehicle Routing Problem - github repository*, `https://github.com/wolledav/VNS-EVRP-2020`, Accessed: March 22, 2024, 2020.

[88] Resende, M. G., Ribeiro, C. C., *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer New York, 2016, ISBN: 9781493965304. DOI: `10.1007/978-1-4939-6530-4`. [Online]. Available: `http://dx.doi.org/10.1007/978-1-4939-6530-4`.

[89] Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R., "Variable Neighborhood Descent", in *Handbook of Heuristics*, Cham: Springer International Publishing, 2018, pp. 341–367.

[90] Windras Mara, S. T., Norcahyo, R., Jodiawan, P., Lusiantoro, L., Rifai, A. P., "A survey of adaptive large neighborhood search algorithms and applications", *Computers & Operations Research*, vol. 146, p. 105 903, 2022, ISSN: 0305-0548. DOI: `10.1016/j.cor.2022.105903`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0305054822001654`.

[91] López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M., "The rpackage irace Package: Iterated Racing for Automatic Algorithm Configuration", *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016. DOI: `10.1016/j.orp.2016.09.002`.

[92] Crognier, G., Tournebise, P., Ruiz, M., Panciatici, P., "Grid operation-based outage maintenance planning", *Electric Power Systems Research*, vol. 190, p. 106 682, 2021, ISSN: 0378-7796. DOI: `10.1016/j.epsr.2020.106682`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0378779620304855`.

[93] Cattaruzza, D., Labbé, M., Petris, M., Roland, M., Schmidt, M., "Exact and Heuristic Solution Techniques for Mixed-Integer Quantile Minimization Problems", *INFORMS Journal on Computing*, 2024. [Online]. Available: `https://hal.science/hal-03665771`.

[94] Gouvine, G., *Mixed-Integer Programming for the ROADEF/EURO 2020 challenge*, 2021. DOI: `10.48550/ARXIV.2111.01047`. [Online]. Available: `https://arxiv.org/abs/2111.01047`.

[95] Zholobova, A., Zholobov, Y., Polyakov, I., Petrosian, O., Vlasova, T., "An Industry Maintenance Planning Optimization Problem Using CMA-VNS and Its Variations", in *Mathematical Optimization Theory and Operations Research: Recent Trends*, A. Strekalovsky, Y. Kochetov, T. Gruzdeva, and A. Orlov, Eds., Cham: Springer International Publishing, 2021, pp. 429–443, ISBN: 978-3-030-86433-0.

[96] Smith, A. E., "Proceedings of the 23rd International Conference of the International Federation of Operational Research Societies", in *Proceedings of the 23rd International Conference of the International Federation of Operational Research Societies*, J. R. Vera and B. Fortz, Eds., ser. IFORS2023, International Federation of Operational Research Societies, 2023. DOI: `10.1287/ifors.2023`. [Online]. Available: `http://dx.doi.org/10.1287/ifors.2023`.

[97] Mansouri, M., Andreasson, H., Pecora, F., "Hybrid reasoning for multi-robot drill planning in open-pit mines", *Acta Polytechnica*, vol. 56, no. 1, p. 47, 2016, ISSN: 1210-2709. DOI: `10.14311/app.2016.56.0047`. [Online]. Available: `http://dx.doi.org/10.14311/APP.2016.56.0047`.

[98] Dubins, L. E., "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", *American Journal of Mathematics*, vol. 79, no. 3, p. 497, Jul. 1957. DOI: `10.2307/2372560`. [Online]. Available: `https://doi.org/10.2307/2372560`.

[99] *Řád doktorského studia Fakulty elektrotechnické Českého vysokého učení technického v Praze, 1. změněné úplné znění účinné od 28. března 2018*, `https://intranet.fel.cvut.cz/cz/rozvoj/rad-doktorskeho-studia_2018a.pdf`, Accessed: March 22, 2024.