TU WIEN Informatics

A Matheuristic for Battery Exchange Station Location Planning for Electric Scooters

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Matthias Rauscher, BSc
Matrikelnummer 01527543

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Mitwirkung: Projektass. Dipl.-Ing. Thomas Jatschka, BSc

Wien, 27. Dezember 2021

Matthias Rauscher	Günther Raidl

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Informatics

A Matheuristic for Battery Exchange Station Location Planning for Electric Scooters

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Matthias Rauscher, BSc
Registration Number 01527543

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Assistance: Projektass. Dipl.-Ing. Thomas Jatschka, BSc

Vienna, 27th December, 2021

Matthias Rauscher Günther Raidl

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Matthias Rauscher, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Dezember 2021

Matthias Rauscher

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliothek, Your knowledge hub
TU WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ein besonderer Dank ergeht an meine Betreuer Günther Raidl und Thomas Jatschka, welche mich durch zahllose Stunden der Diskussion und Ratschläge zu einem erfolgreichen Projekt geführt haben. Ihr unschätzbares Wissen und die wertvollen Ideen haben es mir erlaubt, alle Herausforderungen, welche diese Arbeit mit sich führte, zu meistern.

Ich möchte mich auch bei Honda R&D Co., Ltd.[1] für die Kooperation und Finanzierung im Bezug auf dieses Projekt, und dementsprechend auch auf diese Arbeit, bedanken. Besonderer Dank gebührt Yusuke Okamoto, Hiroaki Kataoka, Tadashi Hayashida von der Honda Motor Company und Tobias Rodemann vom Honda Research Institute Europe[2] für die zahl- und lehrreichen Meetings während der Projektdurchführung.

Weiters möchte ich mich bei Bernhard Kreutzer bedanken, welcher im Rahmen seiner Masterarbeit ebenfalls am Projekt beteiligt war. Große Teile der MILP-Formulierung sind seinen Anstrengungen zu verdanken.

Ich möchte mich auch bei der Algorithms and Complexity Forschungsgruppe der TU Wien[3] für die Bereitstellung der notwendigen Cluster-Infrastruktur zur Durchführung der Experimente.

Vor allem möchte ich mich bei meiner Mama dafür bedanken, dass sie mich mein ganzes Leben lang unterstützt hat und mir in schwierigen Zeiten immer Mut und Hoffnung geben konnte. Natürlich möchte ich mich auch bei meiner liebsten Julie bedanken, welche mein Leben unsagbar bereichert und mich während dieser Arbeit ständig begleitet und unterstützt hat.

[1] https://www.jp.honda-ri.com/en/
[2] https://www.honda-ri.de/
[3] https://www.ac.tuwien.ac.at/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I want to express my sincere gratitude to my supervisors Günther Raidl and Thomas Jatschka who have spent numerous hours in discussion and guidance towards a successful outcome of this project. With their valuable input and knowledge they allowed me to overcome all challenges encountered while working on this project.

I also want to thank Honda R&D Co., Ltd.[4] for their cooperation and funding of the project, and therefore partly of this thesis. I especially want to thank Yusuke Okamoto, Hiroaki Kataoka, Tadashi Hayashida representing the Honda Motor Company and Tobias Rodemann from Honda Research Institute Europe[5] for the frequent and insightful meetings during this project.

Furthermore, I want to thank Bernhard Kreutzer who was also working on this project as part of his master thesis. Large parts of the MILP formulation of the problem can be attributed to his efforts.

I further want to thank the Algorithms and Complexity Group at TU Wien[6] for providing the necessary cluster infrastructure required to perform the evaluations in this thesis.

Due above all, I am very grateful to my mum for supporting me my whole life and for always providing me perspective and hope in times of need. Of course I also want to thank my dearest Julie who is tremendously enriching my life and who supported me on my journey through this thesis.

[4] https://www.jp.honda-ri.com/en/
[5] https://www.honda-ri.de/
[6] https://www.ac.tuwien.ac.at/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

In dieser Arbeit betrachten wir das *Battery Exchange Station Location Problem* (BEXSLP) welches sich mit der Platzierung von Batteriewechsel-Stationen für elektrische Roller beschäftigt. Ziel ist es, eine dreiteilige Zielfunktion unter Erfüllung eines festgelegten Mindestbedarfs zu minimieren. Nutzer können an Batteriewechsel-Stationen leere Batterien ihrer Roller unmittelbar gegen vollständig geladene Batterien tauschen. Diese entleerten Batterien werden dann bei der Station wieder aufgeladen und nach entsprechender Ladezeit anderen Nutzern wieder zur Verfügung gestellt. Hierfür betrachten wir einen Zeithorizont von einem Tag, welcher in gleich lange Zeitintervalle diskretisiert wird (typischerweise wird ein Tag auf 24 Stunden aufgeteilt). Der (Mindest)bedarf ist durch Fahrten von Nutzern mit definiertem Start- und Endziel innerhalb eines bestimmten Zeitslots, in welchem die Nutzer das Fahrzeug aufladen müssen, gegeben.
Stationen können an unterschiedlichen Orten gebaut werden. Abhängig vom Ort, an dem eine Station gebaut wird, gibt es Unterschiede in Bezug auf verschiedene Eigenschaften, wie zum Beispiel die Baukosten, Anzahl der möglichen Batterieslots oder Zeiten, zu welchen Nutzer ihre Batterien tauschen können. Die Planung erfolgt unter Berücksichtigung von drei unterschiedlichen Aspekten welche linear gewichtet in einer gemeinsamen Zielfunktion minimiert werden. Diese drei Aspekte sind (a) die Baukosten für Stationen sowie Erweiterungsmodule, (b) die Kosten für das Laden von Batterien und (c) die Zeitsumme der Umwege, welche dadurch entstehen, dass Nutzer zu einer Ladestation fahren müssen.
In dieser Masterarbeit entwickeln wir eine *Matheuristic*, welche exakte Techniken der mathematischen Programmierung mit heuristischen Methoden kombiniert, um das BEXSLP zu lösen. Wir verwenden eine Large Neighborhood Search (LNS), welche mittels einer Konstruktionsheuristik eine initiale Lösung erzeugt und dann mittels eines Zerstör-und-Reparier-Schemas versucht diese Lösung iterativ zu verbessern. Hierbei werden Teile einer bestehenden Lösung aufgelöst und anhand einer Menge vielversprechender Stationen wieder repariert. In der LNS verwenden wir gemischt-ganzzahlige lineare Programmierung (mixed integer linear programming, MILP) mit relaxierten Eigenschaften in Bezug auf die Anzahl der Stationen und Erweiterungsmodule. Anschließend wird die Lösung für das relaxierte Modell mittels heuristischer Methoden repariert, um eine gültige Lösung abzuleiten. Wir präsentieren mehrere Strategien um vielversprechende Stationen für das Zerstören beziehungsweise Reparieren von Lösungen auszuwählen, welche sich auf einzelne Aspekte unserer mehrteiligen Zielfunktion fokussieren.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Wir erstellen neuartige Testinstanzen basierend auf Vorgehensweisen bestehender Literatur. Anhand dieser Testinstanzen zeigen wir, dass die entwickelte Matheuristic für größere Instanzen um zehn bis dreißig Prozent bessere Resultate erzielt als die Verwendung eines universellen MILP-Lösers.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

In this thesis, we consider the *Battery Exchange Station Location Problem* (BEXSLP) which considers planning the setup of new stations for exchanging batteries of electric scooters with the aim of minimizing a three-part objective function while satisfying an expected amount of demand. Depleted batteries can directly be exchanged by customers at those stations for fully charged ones. Batteries returned at a station are charged and provided to customers again once they are fully charged. A time horizon of one day is considered, discretized into equally long consecutive time intervals (typically a day is split into 24 hours). Demand refers to user trips with a defined start and end point within a certain time interval at which the users need to exchange the batteries of their vehicles. Stations may be set up at given potential locations, which may differ in certain aspects, such as setup costs, number of provided battery slots or different time intervals in which exchanging batteries is possible for customers. This task is done with regard to minimizing three objectives, which are combined in a weighted linear fashion and the requirement that a certain minimal amount of demand must be fulfilled. These three objectives are (a) the setup cost for stations and extension modules, (b) the cost for charging batteries and (c) the total duration of detours for users induced by travelling to a station to exchange batteries.

In this thesis, a matheuristic is developed which combines exact mathematical programming techniques with heuristic methods to solve the BEXSLP. More specifically, a Large Neighborhood Search (LNS) is implemented. The LNS uses an initial solution created by a construction heuristic and iteratively tries to improve the solution quality by applying a destroy and repair scheme, i.e., parts of an incumbent solution are destroyed and repaired with a set of promising stations. In the LNS we make use of a mixed integer linear program (MILP) with relaxed properties regarding the number of stations and modules. Afterwards, heuristic methods are applied to derive feasible solutions from the solutions of the relaxed model. Multiple strategies are presented which specifically focus on individual parts of our multi-component objective to systematically find more promising stations when destroying and repairing solutions.

A set of test instances is designed based on approaches from literature. Using these instances, we show that the matheuristic approach achieves between ten to thirty percent better results for larger instances than using a general-purpose MILP solver.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

Adoption of electric vehicles has significantly increased in the past years and is expected to grow further in the years to come [RHL20]. While users mention a positive environmental impact as a reason for adopting electric vehicles, certain aspects may hinder a wide-spread adoption, such as long charging times [CBW17, LLCG17]. A possible solution for this problem is the construction of electric vehicles which allow users to replace empty batteries with fully charged ones, thus avoiding long waiting times due to recharging. Today, this swapping strategy is not common for electric cars as the method of construction of the respective batteries (weight, complexity, etc.) often hinders users in doing so. However, there are promising possibilities for electric scooters, as batteries can be constructed compactly enough to allow everyday users to easily replace them themselves. Thus, if such batteries are drained, users can remove the empty batteries of their scooters and replace them with fully charged ones provided at designated battery exchange stations. The empty batteries are left at the station for charging and are offered to other customers when fully charged.

An important task in designing such systems is to determine where such battery swapping stations should be set up and how many batteries shall be provided at each location. It has to be considered that different aspects influence this decision, as on the one hand providers of such systems are interested in minimizing setup costs for setting up the necessary equipment and infrastructure. Further, it might be desirable to minimize costs of electricity for charging the batteries, by, for example setting up the battery provider network in a way to allow charging during time periods with cheaper electricity prices. On the other hand, users wish to minimize the time requirement for reaching such battery swapping stations. Battery exchange stations are typically not set up directly on routes taken by individual users but are placed near common and convenient public spaces such as supermarkets or conventional gas stations.

Further, certain limitations may be posed on stations depending on the location such as the maximum number of providable batteries or on the times during which users are allowed to exchange batteries.

As a planning horizon, one day, discretized into equally long consecutive time intervals is considered (typically a day is split into 24 hours).

In general, we assume that users specify their trips in terms of starting location, target location and approximate time and are assigned by the system to an appropriate station for changing batteries.

We further consider different types of vehicles with respect to the number of batteries required to operate.

We refer to the here presented problem as the *Battery Exchange Station Location Problem* (BEXSLP). Such problems can be specified in terms of a mixed integer linear program (MILP). While such approaches are able to find (and guarantee) optimal solutions, they tend to suffer in terms of scalability for larger instances. For such systems it is however reasonable to assume that several hundred or even thousands of potential locations for constructing stations, as well as multiple thousand users have to be considered for planning purposes. We therefore present an approach to tackle these scalablity issues in the form of a matheuristic, i.e., a combination of a heuristic approach with exact mathematical techniques.

In our matheuristic, we obtain an initial solution to the BEXSLP by first solving a linear relaxation of the MILP and afterwards repairing the solution to ensure feasibility. Afterwards, we use a Large Neighborhood Search (LNS) based on a repair and destroy scheme to iteratively improve the solution. In each iteration, parts of an incumbent solution are destroyed and afterwards repaired using a set of promising candidate stations. For repairing the solution, we make use of a linear relaxation of the MILP. Afterwards we again use a heuristic procedure to ensure that the resulting solution is feasible.

For selecting promising stations during the destroy and repair steps we present several strategies, which aim to focus on individual parts, i.e., construction costs, charging costs and induced delays, of our multi-part objective. We further present two strategies to combine multiple aspects of the problem for selecting promising stations.

To evaluate the performance of the approach, we generate a novel set of test instances, based on approaches from literature. We will see that the matheuristic approach performs superior in terms of optimality gaps to using a general-purpose MILP solver for larger instances of the BEXSLP.

1.1 Structure of the Work

In Chapter 2 we show existing work regarding problems which are related to the BEXSLP. Further, we discuss other problems where matheuristics similar to the one proposed by this thesis have been successfully applied.

Chapter 3 establishes the methodologies used in this thesis. In particular, it gives an overview on (mixed integer) linear programs and basics of how general purpose solvers

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

effectively solve such problems. We will further highlight the heuristic elements used in our approach and show how heuristics and exact mathematical techniques can work together to form matheuristics.

Chapter 4 gives a formal definition of the BEXSLP in the form of a MILP.

In Chapter 5 we introduce and explain our developed matheuristic. We discuss how we use a linear programming relaxation of the presented MILP to find an initial solution in a reasonably fast time and how a large neighborhood search (LNS) following a destroy and repair scheme, again making use of a relaxed problem definition, is applied to iteratively improve solutions. We further discuss steps necessary to construct and guarantee feasible solutions despite the mentioned relaxations. Lastly, we present multiple operators, focusing on different aspects of our multi-component objective to select promising stations in our LNS approach.

Afterwards, Chapter 6 presents how a set of novel benchmarking instances was created to test the performance of the presented approach. We further show experimental results of the established matheuristic approach in comparison to solving the MILP formulation of the BEXSLP with a general purpose solver for MILPs.

Finally, Chapter 7 summarizes the presented approach, results and findings and presents possible future work.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

State of the Art and Related Work

This chapter discusses related work which is relevant with regard to the thesis. We first discuss a previous project done at TU Wien which is related to the BEXSLP but also highlight the differences to the problem concerned in this thesis. We then discuss related work in areas related to the BEXSLP. Then, related work is presented in which matheuristics have been successfully applied.

2.1 Previous Work

Parts of the BEXSLP are based on the Multi-Period Battery Swapping Station Location Problem (MBSSLP) [JORR20]. Similar to the BEXSLP, the goal in the MBSSLP is to identify an optimal location placement for battery swapping stations including their required capacities in order to satisfy a specified amount demand with minimum cost. A MILP is formulated and proposed for solving smaller instances as well as an LNS approach for solving larger instances. However, there exist certain differences between the MBSSLP and the BEXSLP. In the MBSSLP the objective function is only concerned with minimizing setup costs, while in the BEXSLP we consider setup costs, charging costs and total duration of detours for customers, combined in a linear weighted fashion. In the MBSSLP there is instead an additional constraint regarding the maximum allowed detour and a loss of users in dependence of the respective detour length is assumed. Further, in the BEXSLP we consider multiple types of vehicles with respect to the number of batteries needed while in the MBSSLP a single type is considered.

2.2 Related Problems

Generally, the BEXSLP can be classified as a facility location-allocation problem (FLP) [BF12] as we are dealing with an optimization problem with a finite set of users with demand for service and a finite set of possible locations for facilities providing such service. More specifically, as in the BEXSLP we assign users to appropriate charging stations to fulfill their demand, our problem is related to the capacitated multiple allocation Fixed-Charge Facility Location problem [LNdG19]. Considering that we optimize the BEXSLP over a given time horizon, we can further place it in the category of Multi Period Facility Location Problems [NSdG15].

In early research regarding facility location-allocation problems, demand is often expressed as weight at nodes of an underlying network of locations and in turn the goal is to serve the demand at these nodes, such as in the maximal covering location model (MCLM) [CR74]. However, Hodgson notes that in certain scenarios demand can be better expressed as (traffic) flow, such as for convenience stores or gas stations [Hod90]. They therefore introduce the Flow Capturing Location Model (FCLM) for covering demand along paths instead of at fixed nodes of an underlying graph. They assume that a shortest path is taken to get from an origin to a destination and accordingly introduce the notion of origin-destination pairs (O/D pairs). In the BEXSLP we also use the notion of O/D-pairs in the form of planned trips specified by the users beforehand and part of our objective is to minimize detours induced by users deviating from their planned trip to visit a battery charging station.

In the FCLM, the demand/flow of a path can only be met by a single facility. Facilities should in general be placed directly at nodes of a path to potentially cover demand/flow of other paths passing through this node according to Berman et al. [BLF92]. However, there exists an extension of the FCLM in which facilities are additionally limited by capacities and flow/demand may be covered by multiple facilities [HRZ96].

Further, regarding alternative-fuel vehicles, Kuby and Lim [KL05] introduce the Flow Refueling Location Model (FRLM) as an extension to the FCLM. They argue that for alternative-fuel vehicles it might be necessary to refuel more than once on a given path due to the limited vehicle range and further consider round-trip paths to avoid vehicles being stuck at a destination with depleted batteries. In the BEXSLP we consider an urban environment in which we assume that trips are short enough such that swapping batteries is necessary at most once per trip and users are always able to reach their assigned battery station before the vehicle's batteries are fully depleted.

In the original FRLM the goal is to select a fixed size set of refueling stations to maximize the total flow volume refueled and is due to computational complexity limited to small instances. However, there exists a more efficient formulation which allows appliance of the FRLM also to larger instances by Capar et al. [CKLT13]. MirHassani and Ebrazi [ME13] also use a more efficient formulation with the additional distinction that instead of selecting a fixed set of stations to maximize the total covered flow they aim to cover

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

all demand while minimizing the cost. This latter formulation is more closely related to the approach we take in the BEXSLP.

2.3 Matheuristics

To solve the BEXSLP, we propose a matheuristic, combining a metaheuristic, in the form of an LNS, with exact mathematical programming techniques, in the form of a MILP.

The general idea of combining metaheuristics with exact mathematical techniques has already been successfully applied in a multitude of areas [PR05, PRP09, DS10, AS14, BPRR11] and is neither restricted to the problem at hand nor to the proposed usage of an LNS as metaheuristic. For example for the Dynamic Facility Layout Problem Kulturel-Konak [KK17] use a metaheuristic in the form of a tabu search to approximate the locations while the exact locations are calculated via mathematical programming. Further, similar to our idea, Keskin and Çatay [KÇ18] use a matheuristic based on an LNS combined with a general purpose solver for MILPs to solve the respective sub-problems. They apply this technique to the vehicle routing problem with cross-docking.

Turkeš, Sörensen, and Cuervo [TSC21] use a similar approach as our proposed one but apply it to the stochastic facility location problem. They employ an iterated local search technique to look for good location and inventory configurations and use a general purpose solver for MILPs for optimizing the respective assignments.

Hosseini, MirHassan, and Hooshmand [HMH17] present the Capacitated Deviation Flow Refueling Location Model. Their work concerns the placement of alternative fuel stations to provide energy for environmental friendly vehicles. Similar to our approach, they use heuristics in combination with a linear program to solve larger instances. They identify promising sets of candidate sites derived from the solution to the linear relaxation of their model and iteratively add them to the solution in a greedy fashion. For the less restrictive Capacitated Facility Location Problem Lagos et al. [LGC+16] use a local search to identify a subset of promising facilities based on the installation cost and the distance between customers and facilities. Then the subproblems are solved to optimalitiy using mathematic programming techniques. They show that the combined approach performs significantly better than using either a local search heuristic or the mathematical program alone.

Related to our problem specification Ghamami, Zockaie, and Nie [GZN16] aim to minimize an objective which considers the infrastructure investment, battery cost and user cost (in terms of waiting time for a free battery slot). To solve larger instances, they also use a matheuristic to find solutions for their problem, however, they base their metaheuristic on Simulated Annealing. They conclude that the usage of a metaheuristic provides much better scalability than solely using a general purpose solver, serving as motivation to our proposed approach.

Calvete et al. [CGI+20] additionally consider customer preferences in their assignment to stations but do not consider charging costs in their objective. We consider construction

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

costs, charging costs and the total duration of detours and ultimately assign customers in a way to minimize this overall objective. To solve their problem, they also employ a matheuristic which is based on an evolutionary algorithm.

It is also important to emphasize that our approach is different from a multi-objective large neighborhood search (MO-LNS) [SH13]. In MO-LNS a set of nondominated solutions is retained instead of just a single best-so-far solution and in each iteration one of these solutions is selected and optimized to aid the overall search procedure. In our approach however, we combine aspects of multiple objectives in a linear weighted fashion into a single multi-part objective function. Nonetheless, the idea of different repair and destroy operator focusing each on one part of our multi-part objective function is inspired by Rifai, Nguyen, and Dawal [RND16].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 3

Methodological Approach

In this chapter we discuss the various methodological approaches and techniques which are relevant to this thesis. We first establish the necessary basics of mixed integer linear programms (MILPs), which are used as part of our matheuristic. We then explain necessary heuristic methods which are used in our approach. Finally, we discuss how heuristics and exact mathematical techniques can be used together as matheuristics.

3.1 Mathematical Programming Techniques

This section is based on Bertsimas and Tsitsiklis [BT97], Schrijver [Sch98] and Wolsey [Wol20].

In a mathematical programming problem the goal is to find a minimum or maximum value of a real valued function while adhering to a set of defined constraints. As the mathematical models used in this thesis focus on a minimization problem we will also focus on minimization problems in this section. We further focus on *linear* programs, i.e., the objective function and all constraints are linear functions.

We first formulate the notion of a *linear program* (LP) as:

$$\min \mathbf{c}^T\mathbf{x} \tag{3.1}$$

$$\text{such that } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{3.2}$$

$$\mathbf{x} \geq 0 \tag{3.3}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{3.4}$$

The vector $\mathbf{x} = (x_1, .., x_n)$ refers to the *decision variables*. The goal is to find an assignment for the decision variables $\mathbf{x}$ that minimizes the objective function (3.1) while adhering to, i.e., not violating, the constraints defined in (3.2) - (3.3).

An assignment of decision variables $\mathbf{x}$ is called a *solution* to the program. A solution of the program is called *feasible* if all constraints are satisfied, otherwise the solution is called *infeasible*. The set of all feasible solutions is often referred to as the *feasible region* or *feasible space*. A solution is called *optimal* if it is feasible and minimizes the objective function of the program. There potentially exist multiple optimal solutions.

The set of all values which can be assigned to a decision variable x_i is called the *domain* of x_i. The domain of a decision variable may be further restricted by a constraint (see Constraint (3.3)), in which case the variable is referred to as *restricted*. Otherwise, the variable is called *unrestricted*.

Constraints may be defined in the form of equalities or inequalities. Note that each equality constraint $\mathbf{Ax} = \mathbf{b}$ can be expressed by two inequalities: $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{Ax} \geq \mathbf{b}$. Further, inequalities can also be expressed with a reversed sign, as: $\mathbf{Ax} \leq \mathbf{b} \Leftrightarrow -\mathbf{Ax} \geq -\mathbf{b}$. In similar fashion can a minimization problem be formulated as a maximization problem (and vice versa) as: $\min \mathbf{c}^T\mathbf{x} \equiv \max \ -\mathbf{c}^T\mathbf{x}$.

LP problems can be solved in polynomial time, for example by using the interior point method [Gon12]. An alternative is the simplex method introduced by Dantzig [Dan51], which although in theory has an exponential worst case performance, usually performs very well in practice.

An *integer linear program* (ILP) is a linear program where the domain of all decision variables is restricted to the set of integers, i.e., it can be formulated as:

$$\min \mathbf{c}^T\mathbf{x} \tag{3.5}$$
$$\text{such that } \mathbf{Ax} \leq \mathbf{b} \tag{3.6}$$
$$\mathbf{x} \geq 0 \tag{3.7}$$
$$\mathbf{x} \in \mathbb{Z}^n \tag{3.8}$$

A *mixed integer linear program* (MILP) is a linear program where the domain of *some* decision variables is restricted to the set of integers, while the domain of others is the set of real numbers. A MILP can therefore be expressed as:

$$\min \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} \tag{3.9}$$
$$\text{such that } \mathbf{Ax} + \mathbf{By} \leq \mathbf{b} \tag{3.10}$$
$$\mathbf{x}, \mathbf{y} \geq 0 \tag{3.11}$$
$$\mathbf{x} \in \mathbb{Z}^n \tag{3.12}$$
$$\mathbf{y} \in \mathbb{R}^n \tag{3.13}$$

Contrary to solving an LP, solving a MILP is $\mathcal{NP}$-hard [Pap81] and solving an ILP is $\mathcal{NP}$-complete [KM78].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

A common procedure for solving MILPs is to generate a decreasing sequence of upper bounds (primal bounds) and an increasing sequence of lower bounds (dual bounds) and to terminate when the difference between the primal bound and dual bound is lower than some small nonnegative value ϵ.

Further note that a primal bound is an upper bound for minimization problems and a lower bound for maximization problems. Similarly, a dual bound is an upper bound for maximization problems and a lower bound for a minimization problems.

There exist efficient multi-purpose MILP solvers such as Gurobi[1] or CPLEX[2] which often follow a *branch-and-bound* algorithm [LW66]. In essence, such algorithms repeatedly divide the search space into smaller subspaces. By using the currently best known solution, also called the *incumbent solution* and dual bounds, some subspaces can be safely removed from consideration. In particular, if the dual bound of a subspace is worse than the incumbent solution, this subspace cannot contain the optimal solution.

3.2 Heuristics

Using exact mathematical techniques proves unsuitable for many problems, as they cannot be solved that way in a reasonable time. An alternative approach to solving such (combinatorial optimization) problems is the usage of heuristics. While heuristics can not guarantee to find an optimal solution, they still aim to find solutions of high quality and this is usually done in significantly less time than when using exact approaches. It is therefore also often reasonable to use exact mathematical techniques for smaller instances of a certain problem and to switch to heuristic approaches for larger instances where using exact methods would take an unreasonable amount of computation time [PR05].

3.2.1 Construction Heuristics

Construction heuristics are applied to find an initial solution to a problem and thus often serve as a starting point for other heuristic approaches. An initially empty solution is iteratively expanded until eventually a complete solution has been formed. Often, the focus lies rather on finding an initial solution fast than on finding a solution with very good quality. The quality of the solution is then commonly improved in subsequent steps of an heuristic approach. Common examples of construction heuristics are greedy heuristics [CLRS09]. Here, a solution is constructed step-by-step by always choosing the element which appears to be best in the current moment, i.e., in a rather myopic way.

3.2.2 Large Neighborhood Search

Large Neighborhood Search (LNS) [GP+10] is a prominent metaheuristic for addressing difficult combinatorial optimization problems, which builds upon effective lower-level heuristics.

[1] https://www.gurobi.com/
[2] https://www.ibm.com/analytics/cplex-optimizer

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

A basic LNS in essence follows a classical local search framework, but usually much larger neighborhoods are considered in each iteration. The key-idea is to search these neighborhoods not in a naive enumerative way but to apply some "more clever" problem-specific procedure to solve the subproblem induced by each neighborhood in order to obtain the best or a promising heuristic solution from the neighborhood. Some successful approaches for doing so include using (mixed) integer programming techniques, like in the approach presented in this thesis, or dynamic programming [CPvdV02].

Frequently, LNS follows a destroy and repair scheme: A current incumbent solution is partially destroyed, typically by freeing a subset of the decision variables and fixing the others to their current values, and then repaired again by finding best or at least promising values for the freed variables.

3.3 Matheuristics

Matheuristics refer to a group of hybrid approaches, i.e., combinations of two different algorithmic approaches. In particular, matheuristics combine metaheuristics and mathematical programming techniques [MSV10, BR16]. The general motivation in doing so is that while exact mathematical techniques can guarantee optimal solutions, the performance tends to scale badly for larger instances in a lot of problems. On the other hand, heuristics give no guarantee on finding optimal solutions but are often able to find sufficiently good solutions in a reasonable time. The basic idea is therefore to combine advantages of both approaches.

Puchinger and Raidl [PR05] classify two major categories of matheuristics. In *collaborative/cooperative combinations* two methods are not part of each other but run in sequential order or are executed in a parallel or intertwined fashion to exchange information. In *integrative/coercive* combinations there is usually a primary method (*master*) with at least one integrated subordinate method. Therefore, again there exists the possibility of exact mathematical algorithms being subordinates to a master metaheuristic, like in the approach presented in this thesis, or the other way around.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

The Battery Exchange Station Location Problem

In the *Battery Exchange Station Location Problem* (BEXSLP) the task is to plan the setup of new stations for exchanging batteries of electric scooters or to extend existing stations with the aim of minimizing three different objectives while satisfying an expected demand. The three objectives are (a) the setup cost for additional stations and extension modules, (b) the cost for charging batteries, and (c) the total duration of detours for users to exchange batteries.

We consider a time horizon of one day that is discretized into equally long consecutive time intervals, for example hours. These intervals are indexed by $\mathcal{T} = \{1, \dots, t_{\max}\}$. Moreover, we consider the planning horizon to be cyclic, i.e., the predecessor of the first interval is the last one and the successor of the last one the first interval. In order to select subsets of this cyclic planning horizon between two time points, we introduce the notation of a *(cyclic) timespan*. Let $\mathcal{J}[k]$ be the element of an ordered list $\mathcal{J}$ at index k. Then, a (cyclic) timespan $\mathcal{J}[k : k']$ of $\mathcal{J}$ with $k, k' \in \{1, \dots, |\mathcal{J}|\}$ is defined as

$$\mathcal{J}[k : k'] = \begin{cases} \{\mathcal{J}[k], \dots, \mathcal{J}[k']\} & \text{if } k \leq k' \\ \{\mathcal{J}[1], \dots, \mathcal{J}[k'], \mathcal{J}[k], \dots, \mathcal{J}[|\mathcal{J}|]\} & \text{else.} \end{cases} \tag{4.1}$$

We make the simplifying assumption that charging any battery always takes the same time and only completely recharged batteries are provided to customers again. Moreover, as trips in an urban environment are usually rather short, we further assume that trips start and end in the same time interval.

We assume a battery swapping station can be set up at any of n different locations referred to as $L = \{1, \dots, n\}$. Each location $l \in L$ has associated

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- setup cost $c_l \geq 0$ for setting up a station with an initial configuration of BEX modules at this location;
- setup cost $c_l^{\text{modul}} \geq 0$ for each additional BEX module at a location where a station is set up or exists already;
- the capacity in terms of the number of battery slots of the initial station configuration $s_l^{\text{ini}} \in \mathbb{N}$;
- the maximum number of additional BEX modules allowed at location $e_l^{\max} \in \mathbb{N}$;
- a timespan $\mathcal{T}_l^{\text{ex}} = \mathcal{T}[t_l^{\text{ex,start}} : t_l^{\text{ex,end}}]$ with $t_l^{\text{ex,start}}, t_l^{\text{ex,end}} \in \mathcal{T}$, in which the station is open for customers and batteries can be exchanged;
- a timespan $\mathcal{T}_l^{\text{dch}} = \mathcal{T}[t_l^{\text{dch,start}} : t_l^{\text{nch,end}}]$ with $t_l^{\text{dch,start}}, t_l^{\text{dch,end}} \in \mathcal{T}$, indicating daytime charging hours;
- and charging costs $c_l^{\text{dch}} \geq 0$ and $c_l^{\text{nch}} \geq 0$ for batteries during daytime and nighttime (i.e., outside daytime) charging hours, respectively.

We also take into account that at some locations $l \in L$ a station with a corresponding configuration of BEX modules may have already been set up at a previous time. In this case the costs c_l for setting up the station are set to zero, and the initial station configuration s_l^{ini} accounts for all existing slots including the already existing extension modules. If feasible, such a station may still be extended by installing up to $e_l^{\max}$ additional BEX modules.

Customer travel demands are given for origin-destination (O/D) pairs Q; let $m = |Q|$ be the number of these O/D pairs. Moreover, let $w_q \geq 0$ be the expected travel time for each O/D pair $q \in Q$ when taking a most direct route without exchanging batteries. Furthermore, let $\tilde{w}_q^l$ be the expected travel time for the O/D pair $q \in Q$ when making a fastest possible detour to location $l \in L$ for exchanging batteries there. Clearly, $\tilde{w}_q^l \geq w_q$ will hold for any $q \in Q,\ l \in L$.

We only consider one type of battery but different vehicle types that require different numbers of batteries. We assume that all batteries of a vehicle are always together exchanged at the same time. Let $\mathcal{I} \subset \mathbb{N}$ be the set of vehicle types represented by the corresponding numbers of needed batteries. The expected number of users with vehicle type $i \in \mathcal{I}$ that need to change batteries on trip $q \in Q$ during a time interval $t \in \mathcal{T}$ is denoted as d_{qi}^t.

A parameter $\delta_{\min} \in (0, 1]$ controls how much of the total customer demand over all time intervals in $\mathcal{T}$ and vehicle types $\mathcal{I}$ has to be satisfied at least, i.e.,

$$d_{\text{sat}} = \delta_{\min} \sum_{q \in Q} \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} i \cdot d_{qi}^t. \tag{4.2}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Note that we weight demands by the number of batteries of their respective vehicle type, such that vehicles with a smaller number of batteries are not favored during the optimization as vehicles with more batteries require more resource for satisfying their demand.

Due to production limitations, the number of total BEX modules available is restricted. Towards this, $z^{\text{modules}} \in \mathbb{N}$ refers to the maximum number of available BEX modules. Alternatively or additionally, the number of stations to be opened can be limited to at most $z^{\text{stations}} \in \mathbb{N}$; already existing stations recognized by their zero setup cost do not count here.

A solution is primarily given by a pair of vectors $x = (x_l)_{l \in L} \in \{0,1\}^n$ and $y = (y_l)_{l \in L}$ with $y_l \in \{0, \ldots, e_l^{\max}\}$ where $x_l = 1$ indicates that a swapping station is to be used at location l and y_l is the corresponding number of additionally installed BEX modules. Clearly, BEX modules may only be allocated at locations where a swapping station is located, i.e., $x_l = 0 \rightarrow y_l = 0$, or expressed as linear inequality

$$e_l^{\max} \cdot x_l \geq y_l, \qquad l \in L. \tag{4.3}$$

It is assumed that customers who want to exchange batteries specify their trip data (origin, destination, approximate time) online and are automatically assigned to an appropriate station for the exchange (if one exists). This way, a better utilization of the swapping stations can be achieved. Consequently, let assignment variables a_{qli}^t denote the part of the expected demand of O/D pair $q \in Q$ w.r.t. vehicle type $i \in \mathcal{I}$ which we assign to a location $l \in L$ during time interval $t \in \mathcal{T}_l^{\text{ex}}$.

A battery returned to a station $l \in L$ during a time period $t \in \mathcal{T}_l^{\text{ex}}$ can only be provided to a customer again after t^{c} time periods from $\mathcal{T}$ have passed. We denote the set of times in which a battery is being charged when returned to a station at time t as $\mathcal{T}_l^{\text{ch}}(t) = \mathcal{T}[\chi_{\text{start}}(t) : \chi_{\text{end}}(t)]$ where $\chi_{\text{start}}(t)$ is the time in $\mathcal{T}$ at which the battery starts charging, i.e.,

$$\chi_{\text{start}}(t) = (t \bmod t_{\max}) + 1, \tag{4.4}$$

and $\chi_{\text{end}}(t)$ is the last time period in $\mathcal{T}$ at which the battery is being charged, i.e,

$$\chi_{\text{end}}(t) = ((t + t^{\text{c}} - 1) \bmod t_{\max}) + 1. \tag{4.5}$$

We have modeled that returned batteries are unavailable for t^{c} time periods by effectively reducing a station's capacity of batteries available for exchange within the next t^{c} time periods after an exchange. Therefore, it must hold that

$$\sum_{t' \in \mathcal{T}_l^{\text{ch}}(t) \cup \{t\}} \sum_{q \in Q} \sum_{i \in \mathcal{I}} i \cdot a_{qli}^{t'} \leq s_l^{\text{ini}} x_l + s^{\text{modul}} y_l \quad \forall l \in L,\ t \in \mathcal{T}_l^{\text{ex}} \tag{4.6}$$

The goal of the BEXSLP is to minimize three different objectives. The first objective is to minimize the setup costs for stations and their corresponding BEX modules, i.e.,

$$\sum_{l \in L} (c_l x_l + c_l^{\text{modul}} y_l). \tag{4.7}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The second objective is to minimize the total charging costs. For this purpose let c_{lt}^{ch} refer to the costs for charging a battery at station $l \in L$ during time interval $t \in \mathcal{T}$, i.e.,

$$c_{lt}^{\text{ch}} = \begin{cases} c_l^{\text{dch}} & \text{for } t \in \mathcal{T}^{\text{dch}}, \\ c_l^{\text{nch}} & \text{else.} \end{cases} \tag{4.8}$$

Then, considering the assignment variables a_{qli}^t over all locations, O/D pairs, vehicle types, and opening times, the total charging costs are

$$\sum_{l\in L}\sum_{q\in Q}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}_l^{\text{ex}}} c_{lt}^{\text{chret}} \cdot i \cdot a_{qli}^t. \tag{4.9}$$

Finally, besides minimizing the station setup and battery charging costs, our last objective is to also minimize the total travel delay induced by the detours for charging, i.e., the sum of the differences in travel times between the routes taken to charge at the assigned stations and the corresponding direct routes, calculated by

$$\sum_{l\in L}\sum_{q\in Q}\left(\tilde{w}_q^l - w_q\right) \cdot \sum_{t\in\mathcal{T}^{\text{ex}}}\sum_{i\in\mathcal{I}} a_{qli}^t. \tag{4.10}$$

We combine the different objectives in a linear fashion with weights $\alpha_{\text{setup}} > 0$, $\alpha_{\text{charging}} > 0$ and $\alpha_{\text{delay}} > 0$ to obtain the total objective function.

In summary, we express the BEXSLP by the following MILP.

$$\begin{aligned}
\min\ & \alpha_{\text{setup}} \sum_{l\in L}(c_l x_l + c_l^{\text{modul}} y_l) + \\
& \alpha_{\text{charging}} \sum_{l\in L}\sum_{q\in Q}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}_l^{\text{ex}}} c_{lt}^{\text{ch}} \cdot i \cdot a_{qli}^t + \\
& \alpha_{\text{delay}} \sum_{l\in L}\sum_{q\in Q}\left(\tilde{w}_q^l - w_q\right) \cdot \sum_{t\in\mathcal{T}_l^{\text{ex}}}\sum_{i\in\mathcal{I}} a_{qli}^t
\end{aligned} \tag{4.11}$$

$$e_l^{\max} \cdot x_l \geq y_l \qquad \forall l \in L \tag{4.12}$$

$$\sum_{l\in L | t\in\mathcal{T}_l^{\text{ex}}} a_{qli}^t \leq d_{qi}^t \qquad \forall t \in \mathcal{T},\ i \in \mathcal{I},\ q \in Q \tag{4.13}$$

$$\sum_{t'\in\mathcal{T}_l^{\text{ch}}(t)\cup\{t\}}\ \sum_{q\in Q}\sum_{i\in\mathcal{I}} i \cdot a_{qli}^{t'} \leq s_l^{\text{ini}} x_l + s^{\text{modul}} y_l \qquad \forall l \in L,\ t \in \mathcal{T}_l^{\text{ex}} \tag{4.14}$$

$$\sum_{q\in Q}\sum_{l\in L}\sum_{t\in\mathcal{T}_l^{\text{ex}}}\sum_{i\in\mathcal{I}} i \cdot a_{qli}^t \geq d_{\text{sat}} \tag{4.15}$$

$$\sum_{l\in L | c_l > 0} x_l + \sum_{l\in L} y_l \leq z^{\text{modules}} \tag{4.16}$$

$$x_l \in \{0,1\} \qquad \forall l \in L \tag{4.17}$$

$$y_l \in \{0,\ldots,e_l^{\max}\} \qquad \forall l \in L \tag{4.18}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$$0 \leq a^t_{qli} \leq \min \left(\frac{s^{\text{ini}}_l + e^{\max}_l \cdot s^{\text{modul}}}{i}, d^t_{qi} \right) \quad \forall l \in L,\ t \in \mathcal{T}^{\text{ex}}_l,\ i \in \mathcal{I},\ q \in Q \quad (4.19)$$

The objective function (4.11) minimizes the total setup costs, the total charging costs, as well as the total detours of customers as defined by Equations 4.7, 4.9, and 4.10. Inequalities (4.12) link variables x_l and y_l and correspond to (4.3). Constraints (4.13) enforce that the total demand assigned from an O/D pair q to locations does not exceed d^t_{qi} during all time periods. Inequalities (4.14) ensure that the required amount of battery modules is available at all locations over all time periods. The minimal satisfied demand to be fulfilled over all time intervals is expressed by inequality (4.15). In a similar fashion Constraint(4.16) restricts the number available of BEX modules. Alternatively or additionally, one may also specify an upper limit on the number of stations newly opened z^{stations} by

$$\sum_{l \in L | c_l > 0} x_l \leq z^{\text{stations}}. \quad (4.20)$$

Finally, the domains of the variables are given in (4.17)–(4.19).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

A Matheuristic for the BEXSLP

5.1 Large Neighborhood Search

MILP solvers usually perform very well for problem instances up to a certain size but the performance deteriorates quickly after a certain point. As in the BEXSLP potentially large instances with a high number of locations and O/D pairs may be encountered, it is therefore necessary to consider scalability aspects when solving this problem for such instances. In the preliminary study concerning the related MBSSLP [JORR20], a Large Neighborhood Search (LNS) was proposed for solving larger instances. As this approach performed well for the MBSSLP, we aim to also employ such an approach for the BEXSLP. Similarly, in one LNS iteration an incumbent BEXSLP solution is first destroyed by closing stations in the solution and then repaired by choosing new stations to open.

In our case, we use a relaxation of the above presented MILP (4.11) – (4.19) to repair solutions, Thus combining heuristic and exact mathematical techniques. Such approaches are often referred to as *Matheuristics* [PR05] and have been successfully employed in other well-known optimization problems, such as the Capacitated Facility Location Problem [LGC+16] or Vehicle Routing Problems [AS14, DS10]. In the following sections we first show how an initial solution is constructed. Afterwards, we give a detailed description of various destroy and repair operators designed for this problem.

5.2 Construction Heuristic

We base our construction heuristic on the above presented MILP (4.11) – (4.19) for the BEXSLP. Specifically, the idea of our construction heuristic is to solve the linear programming relaxation of this MILP, i.e., we allow the x and y variables to be continuous, and then derive a feasible BEXSLP solution from the solution to this relaxation. For

TU WIEN Bibliothek Your knowledge hub

getting a feasible solution we use a similar approach as presented in [JORR20] where a feasible MBSSLP solution is obtained from a solution to the relaxed model by rounding up all fractional values. However, for the BEXSLP further steps are necessary, as the number of stations and modules may be limited. Let $(\tilde{x}, \tilde{y}, a)$ be a solution to the linear programming relaxation of the MILP (4.11) – (4.19). As mentioned before, in a first step all fractional $\tilde{x}$ and $\tilde{y}$ values are rounded up, i.e., $\lceil \tilde{x} \rceil = (\lceil \tilde{x}_l \rceil)_{l \in L}$ and $\lceil \tilde{y} \rceil = (\lceil \tilde{y}_l \rceil)_{l \in L}$. Next, if Constraint (4.20) is not satisfied, the number of stations is reduced. To enforce the constraint, we sort the vector $\tilde{x}$ of our relaxed solution in descending order and only keep the z_{stations} stations with the highest values, resulting in a new (potentially infeasible) solution (x', y', a') in which all non-selected stations with their associated capacities and allocated demand are discarded.

Further, if Constraint (4.16) is violated, the number of total BEX modules needs to be reduced as well, until only a total of at most z^{modules} modules remain in the solution. As mentioned in Section 4, certain stations may already exist in BEXSLP instances, thus the base modules of these stations are not included in this restriction. Moreover, there may exist stations at locations $l \in L$ for which $s_l^{\text{ini}} < s^{\text{modul}}$. Therefore, removing modules might result in an insufficient number of battery slots to satisfy all of the necessary demand. The general strategy to address this problem is to first reduce the number of modules in the solution to z^{modules} and then, if necessary, to remove stations (including potential extension modules). Afterwards, we add a number of extension modules equivalent to the number of removed modules to other stations in the solution. This way, we replace base modules with extension modules, which offer more battery slots.

Our procedure for reducing the number of modules in a solution is described by Algorithm 2. When reducing the number of modules we prioritize stations which have been newly constructed and which do not allow around the clock exchanging. If no such candidates exist, we first resort to newly constructed stations with unrestricted exchanging times and after that to stations which already pre-exist but possess at least one extension module, as this counts towards z^{modules}. From the selected set of stations we then select the station l with the lowest fractional part of $\tilde{y}_l$ of the original linear programming relaxation solution. We then remove an extension module from this station or if none exist remove the base module and therefore close the station.

Afterwards, while the number of provided battery slots in the new solution is smaller than $\sum_{l \in L} s_l^{\text{ini}} \tilde{x}_l + s^{\text{modul}} \tilde{y}_l$, i.e., less than in the solution of the linear programming relaxation, we proceed as follows: We first close a random station at location l with $s_l^{\text{ini}} < s^{\text{modul}}$. We again prioritize stations which have been newly constructed and do not allow around the clock exchanging. Then we add an equivalent amount of BEX modules, i.e., $x_l + y_l$, to other random stations which may be extended with further modules, prioritizing stations with exchange times which are a superset of the exchange times of the recently closed station. The idea is, that the so extended stations are guaranteed to be able to handle the demand of the closed station. If no such replacement stations exist, we pick a random

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Algorithm 1: Repair BEXSLP Solution

Input : a solution $(\tilde{x}, \tilde{y}, a)$ to the BEXSLP with potentially fractional x and y values
maximum number of allowed modules z_{modules}
maximum number of allowed stations z_{stations}

Output: feasible BEXSLP solution (x, y, a')

$x \leftarrow \lceil \tilde{x} \rceil$
$y \leftarrow \lceil \tilde{y} \rceil$

$x \leftarrow$ keep top z^{stations} according to $\tilde{x}$
$(x, y) \leftarrow ensure\_z\_modules(x, y)$

// Ensure that there are still enough battery slots
while $\sum_{l \in L} s_l^{\text{ini}} x_l + s^{\text{modul}} y_l < \sum_{l \in L} s_l^{\text{ini}} \tilde{x}_l + s^{\text{modul}} \tilde{y}_l$ **do**
  $l \leftarrow$ random station location l prioritizing already existing stations with no around the clock exchange times
  $num\_modules = x_l + y_l$
  $x_l = 0,\ y_l = 0$
  while $num\_modules > 0$ **do**
    $l' \leftarrow$ random station location l' prioritizing locations with exchange times similar to l
    $modules\_to\_add = \min(num\_modules, e_{l'}^{\max} - y_{l'})$
    $y_{l'} = y_{l'} + modules\_to\_add$
    $num\_modules = num\_modules - modules\_to\_add$
  end while
end while
// Use LP to find a new demand assignment for the new x and y variables
$a' \leftarrow$ solve LP w.r.t. (x, y)
return (x, y, a')

one which can be extended with further modules.

Finally, we need to redistribute the allocated demand a. This is done with the MILP (4.11) – (4.19) by restricting the domain of x and y according to the current configuration of stations and modules. The procedure is illustrated in Algorithm 1.

5.3 Destroy and Repair Operators

We introduce several destroy and repair operators according to the following scheme. Let (x, y, a) be a solution to the BEXSLP. Moreover, let $L_0(x) \subseteq L$ be the set of locations with closed stations in x and $L_1(x) \subseteq L$ be the set of locations with open stations in x.

Algorithm 3 shows the basic procedure of our LNS proposed for solving BEXSLP

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Algorithm 2: Ensure z^{modules}

Input : stations and modules (x, y) of a BEXSLP solution
maximum number of allowed modules z_{modules}
$\tilde{y}$ vector of extension modules of linear relaxed solution

Output: modified (x, y) with at most z^{modules} modules

while $\sum_{l \in L | c_l > 0} x_l + \sum_{l \in L} y_l > z^{\text{modules}}$ **do**
  $L_{\text{cl}} \leftarrow$ non empty candidate set chosen according to following priority:

    1. $\{l \in L \mid x_l == 1 \text{ and } c_l > 0 \text{ and } \mathcal{T}_l^{\text{ex}} \neq [1:24]\}$
    2. $\{l \in L \mid x_l == 1 \text{ and } c_l > 0\}$
    3. $\{l \in L \mid x_l == 1 \text{ and } y_l > 0 \text{ and } \mathcal{T}_l^{\text{ex}} \neq [1:24]\}$
    4. $\{l \in L \mid x_l == 1 \text{ and } y_l > 0\}$

  select station at location l with minimal $(\tilde{y}_l - \lfloor \tilde{y}_l \rfloor)$ from L_{cl}
  if $y_l > 0$ **then**
    $y_l = y_l - 1$
  else
    $x_l = 0$
  end if
end while
return (x, y, a)

instances and how our destroy and repair operators are applied. In each iteration of the LNS, while the termination criterion has not yet been reached, an incumbent solution is first destroyed. Specifically, a destroy operator first selects a set of ν locations $L_{\text{destroy}} \subseteq L_1(x)$. Then, each of those stations are destroyed by setting the number of modules to zero and un-allocating all corresponding demand, i.e., $x_l = 0,\ y_l = 0$, and $a^t_{qli} = 0\ \forall l \in L_{\text{destroy}},\ t \in \mathcal{T}^{\text{ex}}_l,\ q \in Q,\ i \in \mathcal{I}$. Afterwards, a repair operator is then applied to make the solution feasible again. For this purpose, the operator first selects a set of ν' locations $L'_{\text{repair}} \subseteq L_0(x) \setminus L_{\text{destroy}}$. To generate the final repair set, we also add all locations in L_{destroy}, i.e. $L_{\text{repair}} = L'_{\text{repair}} \cup L_{\text{destroy}}$. This last step is to guarantee that the MILP used for repairing can always produce feasible solutions, as, if the selected stations L'_{repair} would proof insufficient in this regard, the previous solution could always be restored.

When repairing a solution, it has to be considered how much more demand needs to be satisfied and how much demand from which O/D pairs is still available to be assigned to a station. For this purpose, let $D' = (d'^t_{qi})_{t \in T, q \in Q, i \in \mathcal{I}}$ be the demand not yet assigned to any opened location in the destroyed solution, i.e.,

$$d'^t_{qi} = d^t_{qi} - \sum_{l \in L_1(x) \setminus L_{\text{destroy}}} a^t_{qli}. \tag{5.1}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Algorithm 3: Large Neighborhood Search for the BEXSLP

Input : a BEXSLP instance
a solution (x, y, a) to the given instance
size of the destroy set ν
size of the repair set ν'

Output: a new solution (x', y', a') to the given instance

$(x', y', a') \leftarrow \emptyset$
while *termination criterion not reached* **do**
    $L\_destroy \leftarrow$ set of ν station locations in the current solution
    //Destroy the selected stations:
    for each $l \in L\_destroy$ **do**
        $x_l \leftarrow 0,\ y_l \leftarrow 0$
        $a^t_{qli} \leftarrow 0,\ \forall q \in Q, t \in \mathcal{T}^{\text{ex}}_l, i \in \mathcal{I}$
    end for
    $L\_repair \leftarrow$ set of ν' station locations not in the current solution
    $L\_repair \leftarrow L\_repair \cup L\_destroy$
    $(x, y, a) \leftarrow$ construct relaxed solution w.r.t. L_repair
    $(x, y, a) \leftarrow$ repair_solution(x, y, a)
    if (x, y, a) *is better than* (x', y', a') **then**
        $(x', y', a') = (x, y, a)$
    end if
end while
return (x', y', a')

Moreover, let d^-_{sat} be the amount of demand satisfied in the destroyed solution, i.e.,

$$d^-_{\text{sat}} = \sum_{l \in L_1(x) \setminus L_{\text{destroy}}} \sum_{t \in \mathcal{T}^{\text{ex}}_l} \sum_{q \in Q} \sum_{i \in \mathcal{I}} a^t_{qli}. \tag{5.2}$$

Therefore, the goal of the repair function is to assign at least $d'_{\text{sat}} = d_{\text{sat}} - d^-_{\text{sat}}$ demand from D' to the locations $L' = L_{\text{destroy}} \cup L_{\text{repair}}$. For this purpose, let I be an instance to the BEXSLP. Then, $I[L', D', d'_{\text{sat}}]$ is a residual instance of I in which L, $D = (d^t_{qi})_{t \in T, q \in Q, i \in \mathcal{I}}$, and d_{sat} are replaced with L',D', and d'_{sat}.

To decide which stations to open, with how much capacity and which demand to assign to these stations, we use a similar procedure as for the construction heuristic. We first employ the MILP (4.11) – (4.19) with continuous y variables on $I[L', D', d'_{\text{sat}}]$. Afterwards, the resulting solution is repaired in the same fashion as for the construction heuristic, i.e., as described in Section 5.2.

In the following sections, we will introduce the various destroy and repair operators for the BEXSLP. In Chapter 6 these operators are then experimentally evaluated and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

compared. We first present a randomized approach, followed by operators which focus on individual objectives of our multi-part objective function. The idea is that all of these operators may then be used together within our LNS by selecting different repair and destroy operators in each iteration, thus alternately focusing on a different part of the objective. In contrast to this procedure, we also propose a repair and destroy operator making decisions based on the overall objective.

5.3.1 Randomized Operators

For these operators, the sets L_{destroy} and L_{repair} are generated in a randomized way. The *Randomized Destroy Operator* selects ν station locations uniformly at random from $L_1(x)$ to create the set of station locations to destroy L_{destroy}. In a similar fashion, the *Randomized Repair Operator* selects ν' station locations from the set $L_0(x)$ to create the set L'_{repair}.

5.3.2 Delay-Based Operators

The general idea of the delay-based operators is to remove locations which induce large detours and to replace them with new locations that are placed more conveniently for satisfying the remaining demand. The delay-based operators use tournament selection for generating their respective location sets.

For the *Delay-Based Destroy Operator* the set L_{destroy} is generated over ν iterations. In each iteration first k candidate locations from $L_1(x) \setminus L_{\text{destroy}}$ are selected at random. Afterwards, from this candidate set the location with the largest induced travel delay per unit of assigned demand, i.e.,

$$\frac{\sum\limits_{q\in Q}\left((\tilde{w}_q^l - w_q)\sum\limits_{i\in\mathcal{I}}\sum\limits_{t\in\mathcal{T}_l^{\text{ex}}} a_{qli}^t\right)}{\sum\limits_{q\in Q}\sum\limits_{i\in\mathcal{I}}\sum\limits_{t\in\mathcal{T}_l^{\text{ex}}} a_{qli}^t} \tag{5.3}$$

is added to L_{destroy}. Ties are broken randomly.

The *Delay-Based Repair Operator* works in a similar way. Locations from the set $L_0(x)$ are added to L'_{repair} via tournament selection over ν' iterations by again generating a random candidate set of size k from the set $L_0(x) \setminus L'_{\text{repair}}$ and then selecting the most promising candidate in each iteration. To identify promising locations, the idea is to calculate for a location l the average induced delay over the so far unallocated demand, i.e.,

$$\frac{\sum\limits_{q\in Q}\left((\tilde{w}_q^l - w_q)\sum\limits_{i\in\mathcal{I}}\sum\limits_{t\in\mathcal{T}_l^{\text{ex}}} {d'}_{qi}^t\right)}{\sum\limits_{q\in Q}\sum\limits_{i\in\mathcal{I}}\sum\limits_{t\in\mathcal{T}_l^{\text{ex}}} {d'}_{qi}^t}. \tag{5.4}$$

However, a crucial aspect to consider is that a unit of demand cannot be covered by multiple station locations at once. Therefore, we want to take into account that the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

demand already covered by a previously selected location should not be considered in the subsequent selection steps of our iterative procedure. However, estimating this demand exactly would be too time consuming. Instead, we use a simplified estimation in which we assume that the remaining demand d'_{sat} will be assigned evenly among all ν' stations of the resulting repair set. Further, we assume that one station can either completely cover the remaining demand of an O/D pair or none of it. More formally, let l be a location to be added to L'_{repair}. We then iteratively select the O/D pairs $q \in Q$ which induce minimal delay to l and then discard all of the uncovered demand $\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} d'^{t}_{qi}$ of q. This procedure is repeated until the amount of discarded demand exceeds $\frac{d'_{\text{sat}}}{\nu'}$. This demand is then no longer considered in the future iterations of the tournament selection. Note however, that the demand is only considered discarded for deciding which locations to add to L'_{repair}. When applying the MILP to L_{repair} in order to repair the solution, all of the so far uncovered demand is considered again.

5.3.3 Charging-Based Operators

For the charging-based operators we aim to estimate and in turn minimize the charging costs which can be attributed to each station $l \in L$. Towards this, we follow the same procedure as for the delay-based operators.

The *Charging-Based Destroy Operator* as well as the *Charging-Based Repair Operator* again generate their respective sets via tournament selection over ν and ν' iterations, respectively. In each iteration the charging-based destroy operator selects from a set of k random candidates of $L_1(x) \setminus L_{\text{destroy}}$ the location with the lowest charging costs per unit of assigned demand, i.e.,

$$\frac{\sum_{q \in Q} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}_l^{\text{ex}}} c^{\text{ch}}_{lt} \cdot i \cdot a^{t}_{qli}}{\sum_{q \in Q} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}_l^{\text{ex}}} i \cdot a^{t}_{qli}}. \tag{5.5}$$

In a similar way, the charging-based repair operator selects from a set of k random candidates of $L_0(x) \setminus L'_{\text{repair}}$ the location with the highest potential charging costs per unit of unallocated demand, i.e.,

$$\frac{\sum_{q \in Q} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}_l^{\text{ex}}} c^{\text{ch}}_{lt} \cdot i \cdot d'^{t}_{qi}}{\sum_{q \in Q} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}_l^{\text{ex}}} i \cdot d'^{t}_{qi}}. \tag{5.6}$$

Just as for the delay-based repair operator, we also take into account that the demand already covered by a previously selected location should not be considered in subsequent selection steps. Therefore, we use a similar procedure as used by the delay-based repair operator to remove O/D pairs from future iterations, by iteratively discarding O/D pairs

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$q \in Q$ with the lowest charging costs w.r.t. to a selected candidate location l according to

$$\sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} i \cdot {d'}^{t}_{qi} \cdot c^{\text{ch}}_{lt} \tag{5.7}$$

until the associated discarded demand again exceeds $\frac{d'_{\text{sat}}}{\nu'}$.

5.3.4 Construction-Based Operators

For the construction-based operators we focus on the construction cost portion (Equation 4.7) of our objective function. We again use a tournament selection procedure to select locations in a controlled randomized way.

The *Construction-Based Destroy Operator* generates the set L_{destroy} over ν iterations, adding one location to L_{destroy} in each iteration. To determine the locations to be added, in each iteration first a set of k random candidate locations from $L_1(x) \setminus L_{\text{destroy}}$ is generated. Then the candidate l with the largest construction costs per battery slot, i.e.,

$$\frac{c_l + c^{\text{modul}}_l y_l}{s^{\text{ini}}_l + s^{\text{modul}}_l y_l} \tag{5.8}$$

is added to L_{destroy}. In case of a tie, one location with the highest costs is selected at random.

Equivalently, the *Construction-Based Repair Operator* again generates the set L'_{repair} over ν' iterations. In each iteration a set of k random candidate locations from $L_0(x) \setminus L'_{\text{repair}}$ is generated. To estimate the potential construction costs of a location l, we assume the capacity of a station at location l to be similar to the average capacity of the stations in L_{destroy}, i.e.,

$$y_{\text{avg}} = \frac{\sum\limits_{l \in L_{\text{destroy}}} y_l}{\nu}. \tag{5.9}$$

Consequently, the candidate location l with the lowest potential construction costs per battery slot, calculated by

$$\frac{c_l + c^{\text{modul}}_l \min(y_{\text{avg}}, e^{\max}_l)}{s^{\text{ini}}_l + s^{\text{modul}}_l \min(y_{\text{avg}}, e^{\max}_l)}, \tag{5.10}$$

is added to L'_{repair}. Ties are broken randomly.

5.3.5 Weighted Sum Operators

In our experimental evaluation in Chapter 6 we will not only test each of the previously introduced operators individually, but will also investigate a variant where in each iteration of the LNS the destroy and repair operator is randomly selected from the delay-, charging- and, construction-based operators. In contrast to this approach, we also want

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

to investigate a variant that considers all parts of the objective of the BEXSLP within a single repair/destroy-operator, i.e., the weighted sum operators.

The *Weighted Sum Destroy Operator* follows the same procedure as the previously introduced destroy operators by constructing the set L_{destroy} over ν iterations, always adding one location to L_{destroy} in each iteration. For each iteration a set of k candidate locations from $L_1(x) \setminus L_{\mathrm{destroy}}$ is initially generated. Then, the candidate l that contributes most to the objective value of x in relation to its capacity and assigned demand, i.e.,

$$\begin{aligned}
&\alpha_{\mathrm{setup}} \frac{c_l + c_l^{\mathrm{modul}} y_l}{s_l^{\mathrm{ini}} + s_l^{\mathrm{modul}} y_l} + \\
&\alpha_{\mathrm{charging}} \frac{\sum\limits_{q \in Q} \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} c_{lt}^{\mathrm{ch}} \cdot i \cdot a_{qli}^t}{\sum\limits_{q \in Q} \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} i \cdot a_{qli}^t} + \\
&\alpha_{\mathrm{delay}} \frac{\sum\limits_{q \in Q} \left((\tilde{w}_q^l - w_q) \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} a_{qli}^t \right)}{\sum\limits_{q \in Q} \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} a_{qli}^t}
\end{aligned} \tag{5.11}$$

is added to L_{destroy}.

Similarly, the *Weighted Sum Repair Operator* generates the set L'_{repair} over ν' iterations. In each iteration a set of k random candidate locations from $L_0(x) \setminus L'_{\mathrm{repair}}$ is initially generated. We then aim to estimate how much a candidate location l would contribute to the objective value of the repaired solution in relation to its predicted capacity and the so far uncovered demand. For this purpose we combine the metrics used for the delay-, construction-, and charging-based repair operators:

$$\begin{aligned}
&\alpha_{\mathrm{setup}} \frac{c_l + c_l^{\mathrm{modul}} \min(y_{\mathrm{avg}}, e_l^{\max})}{s_l^{\mathrm{ini}} + s_l^{\mathrm{modul}} \min(y_{\mathrm{avg}}, e_l^{\max})} + \\
&\alpha_{\mathrm{charging}} \frac{\sum\limits_{q \in Q} \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} c_{lt}^{\mathrm{ch}} \cdot i \cdot d'^t_{qi}}{\sum\limits_{q \in Q} \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} i \cdot d'^t_{qi}} + \\
&\alpha_{\mathrm{delay}} \frac{\sum\limits_{q \in Q} \left((\tilde{w}_q^l - w_q) \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} d'^t_{qi} \right)}{\sum\limits_{q \in Q} \sum\limits_{i \in \mathcal{I}} \sum\limits_{t \in \mathcal{T}_l^{\mathrm{ex}}} d'^t_{qi}}
\end{aligned} \tag{5.12}$$

The candidate with the lowest value is then added to L'_{repair}. Ties are broken randomly.

Finally, the objective-based repair operator uses the same procedure used by the delay- and charging-based repair operator to prevent already covered demand from being considered in future iterations of the tournament selection. Considering a selected candidate location

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

l, O/D pairs $q \in Q$ with minimal

$$\begin{aligned}
&\alpha_{\text{setup}} \left(c_l x_l + c_l^{\text{modul}} \min(y_{\text{avg}}, e_l^{\max}) \right) + \\
&\alpha_{\text{charging}} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}_l^{\text{ex}}} c_{lt}^{\text{ch}} \cdot i \cdot a_{qli}^t + \\
&\alpha_{\text{delay}} \left(\tilde{w}_q^l - w_q \right) \cdot \sum_{t \in \mathcal{T}_l^{\text{ex}}} \sum_{i \in \mathcal{I}} a_{qli}^t
\end{aligned} \tag{5.13}$$

are iteratively discarded until the associated discarded demand again exceeds $\frac{d'_{\text{sat}}}{\nu'}$.

5.3.6 Addressing Floating Point Issues

While using a MILP for repairing solutions within our LNS comes with a lot of advantages, there is the caveat that MILP solvers usually introduce small mathematical imprecisions due to floating point issues. If not considered, these imprecisions may accumulate over time and eventually lead to noticeable rounding errors which may ultimately lead to seemingly infeasible solutions. In our MILP (4.11) – (4.19) and its respective relaxations, these issues are most pronounced w.r.t. the variables $a_{qli}^t,\ t \in \mathcal{T}_l^{\text{ex}},\ q \in Q,\ l \in L,\ i \in \mathcal{I}$ and their counterpart ${d'}_{qi}^t\ t \in \mathcal{T}, q \in Q,\ i \in \mathcal{I}$ representing the not yet assigned demand. Therefore, we explicitly adjust the mismatch between a_{qli}^t and ${d'}_{qi}^t$ after each iteration. In particular, we set ${d'}_{qi}^t = \max\left(0, d_{qi}^t - \sum_{l \in L} a_{qli}^t\right) \forall t \in \mathcal{T}, q \in Q, i \in \mathcal{I}$.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

Experiments and Results

6.1 Test Instances

Similar to the approach taken for the MBSSLP [JORR20] we aim to create artificial test instances for the BEXSLP, however some of the properties are chosen based on information provided by Honda R&D.

We create six groups of instances identified by their number of station locations n and number of O/D pairs m as (n, m). In particular we create the instance groups $(50, 100)$, $(100, 200)$, $(200, 400)$, $(300, 600)$, $(400, 800)$, $(500, 1000)$. For each subgroup we generate 30 instances. In total we therefore generate 180 individual instances.

Potential locations of battery swapping stations as well as origin and destination locations of customers are located within a square grid $\{1, \ldots, \lceil \xi \sqrt{n} \rceil\}^2$ with $\xi = 800$.

We generate an undirected network graph $G = (V, E)$ following a similar approach as in the MBSSLP [JORR20]. First, $|V| = 5n$ random points are sampled from the grid. Then, we extract a Euclidean spanning tree from a Delaunay triangulation of V and add its edges to E. Finally, we add n additional randomly chosen pairs $(u, v) \in V \times V$ with $u \neq v$ as edges to E. Should an edge already exist in E, a new node pair is generated.

The set of possible locations for battery swapping stations L is generated by selecting n nodes from V at random. Battery swapping stations may already pre-exist on certain locations, i.e., for such a station at location l it holds that $c_l = 0$. For each location $l \in L$ there is a 10% chance to have a pre-existing station. Otherwise, the costs for building the station c_l at l are chosen uniformly at random from $\{5000, \ldots, 7000\}$.

The cost for adding a BEX module c_l^{modul} at l is chosen uniformly at random from $\{2000, \ldots, 4000\}$, as it was common for the Honda R&D instances that costs for additional modules to be lower than those for constructing stations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The initial number of battery slots s_l^{ini} of a station, either when constructed, or pre-existing, is set to six and the number of battery slots added by an extension BEX module s_l^{modul} is eight. These values are set according to the provided instance information.

We select the maximal number of additional BEX modules $e_l^{\max}$ allowed to be added at a station at location l uniformly at random from $\{1, \ldots, 5\}$.

We assume the cyclic planning horizon $\mathcal{T} = \{1, \ldots, 24\}$ representing a day in 24 time steps. Further, we consider three distinct groups of stations regarding their opening times. Intuitively this may be viewed as each station belonging to a certain company with a certain opening time policy. A station is assigned to a certain group according to a weighted random procedure. In particular a station belongs to one of the following three groups regarding their (cyclic) and continuous opening times:

1. $[1 : 24]$, with a 45% probability
2. $[6 : 20]$, with a 45% probability
3. $[18 : 8]$, with a 10% probability

We define the interval of daytime charging hours as $\mathcal{T}_l^{\text{dch}} = \mathcal{T}[7 : 23]$, i.e., from 7a.m. to 11p.m. for all $l \in L$. Accordingly, the interval of nighttime charging hours is defined as $\mathcal{T}_l^{\text{nch}} = \mathcal{T} \setminus \mathcal{T}_l^{\text{dch}}$. The cost of charging during daytime charging hours c_l^{dch} is chosen uniformly at random from the interval $\{3, \ldots, 5\}$. The cost of charging during nighttime charging hours c_l^{nch} is chosen uniformly at random from the interval $\{1, \ldots, 3\}$.

We define the set of vehicle types $\mathcal{I} = \{2, 4\}$ where each vehicle type has the respective number of batteries.

Origin and destination locations are chosen from a random subset $V' \subseteq V$ with $|V'| = \min(\frac{m}{2}, 5n)$. To each $v \in V'$ a random weight γ_v is assigned according to lognormal distribution with mean μ and standard deviation $\sigma = 0.5$. The weights represent popularity values, i.e., nodes with higher weights have higher incoming and outgoing traffic. In particular, for our instances we specify the mean μ of the lognormal distribution used to generate the popularity values as $\mu = \ln(25)$.

The traffic of an OD-pair $(u, v) \in V' \times V'$, however, does not only depend on the weights of its incident nodes but also on its length. Hence, we also assign weights γ_q to each OD-pair $q = (u, v) \in V' \times V'$ such that γ_q corresponds to $f_{\text{PDF}}(w(p_{uv}))$ with f_{PDF} being the probability density function of a lognormal distribution with mean $\mu = \ln(5000)$ and standard deviation $\sigma = 0.2$. The total demand d_q^{total} of an O/D-pair $q = (u, v)$ (over all $t \in \mathcal{T}$, $i \in \mathcal{I}$) is then calculated as

$$d_q^{\text{total}} = \gamma_u \cdot \gamma_v \cdot \gamma_q. \tag{6.1}$$

We then set Q to be the set of O/D-pairs q of $V' \times V'$ for which d_q^{total} is highest.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

This total demand of each O/D-pair is distributed over the time steps $\mathcal{T} = \{1, \ldots, 24\}$ and recharging a battery requires two time periods, i.e., $t^c = 2$, as was common in instances provided by Honda R&D. We assume each customer to travel twice on the corresponding path, once in the morning to get to work and once in the evening to travel back home and we assume that customers need to swap batteries once per trip. The demand of each time period $t \in \mathcal{T}$ is determined by two normal distributions $\mathcal{N}_{\text{morning}}(8, 1)$ and $\mathcal{N}_{\text{evening}}(18, 2)$, respectively. From each distribution 10 samples t are generated and transformed by

$$t := (\lceil t \rceil \bmod t_{\max}) + 1 \tag{6.2}$$

to fit in our cyclic horizon approach. Afterwards, d_q^{total} is distributed over $\mathcal{T}$ according to the frequency in which the time periods $t \in \mathcal{T}$ appear in the generated samples.

Next, the demand has to be distributed to the individual vehicle types. For this, we assume that the proportion of vehicles with two batteries to vehicles with four batteries is $4:1$. The demand of each vehicle type $i \in \mathcal{I}$ is determined by a binomial distribution $\mathcal{B}(1, \frac{4}{5})$. In other words, in our case a successful outcome of the experiment corresponds to the vehicle type with two batteries and a negative outcome of the experiment corresponds to the vehicle type with four batteries. We generate 100 samples from this distribution and distribute the demand of O/D pair $q \in Q$ at time slot $t \in \mathcal{T}$ according to the frequency in which each vehicle type $i \in \mathcal{I}$ appears in the generated samples.

Instances are designed to be solved with $d_{\min}$ set to 1.0, i.e., all of the given demand has to be satisfied.

We restrict the total number of BEX modules that are allowed to be used z^{modules} such that at most 3% of the total available modules may be built. Towards this, we specify $z^{\text{modules}} = \lfloor 0.03 \cdot (\sum_{l \in L} e_l^{\max} + |\{l \in L \mid c_l > 0\}|) \rfloor$. Note that z^{modules} has been chosen according to information provided by Honda R&D.

We do not explicitly specify a maximum number of stations to be constructed z^{stations}, as the z^{modules} constraint was more relevant for the colleagues at Honda R&D. Further, by specifying the z^{modules} constraint, we implicitly limit the number of newly constructed stations anyway, as the base module of constructed stations is counted towards the z_{modules} constraint.

6.2 Experimental Results and Discussion

In this section we evaluate the performance of our approach for solving the BEXSLP. We evaluate the performance on our own instance scenario discussed in Section 6.1.

The presented algorithms were implemented in Julia[1] 1.6.1 using the JuMP package[2] and Gurobi[3] 9.1 as underlying MILP solver.

[1] https://julialang.org/
[2] https://jump.dev/JuMP.jl/stable/
[3] https://www.gurobi.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

All test runs have been executed on an Intel Xeon E5-2640 v4 2.40GHz machine in single-threaded mode with a global time limit of one hour per run. We set the maximum allowed memory to be used depending on the instance group, see Table 6.1.

Table 6.1: Maximum allowed memory to be used for each instance group.

Instance Size	Maximum allowed memory
(50, 100)	4 GB
(100, 200)	4 GB
(200, 400)	6 GB
(300, 600)	12 GB
(400, 800)	16 GB
(500, 1000)	24 GB

When considering the three parts of the objective function, we found that by changing α_{delay}, the most notable differences in performance can be experienced. For this reason, we evaluate three different alpha configurations which only differ in the α_{delay} parameter:

1. $\alpha_{\text{setup}} = 0.01, \alpha_{\text{charging}} = 0.01, \alpha_{\text{delay}} = 0.1$
2. $\alpha_{\text{setup}} = 0.01, \alpha_{\text{charging}} = 0.01, \alpha_{\text{delay}} = 1.0$
3. $\alpha_{\text{setup}} = 0.01, \alpha_{\text{charging}} = 0.01, \alpha_{\text{delay}} = 10.0$

Therefore, if not explicitly specified otherwise it can be assumed that $\alpha_{\text{charging}} = 0.01$ and $\alpha_{\text{setup}} = 0.01$ is used for all shown results.

Further, if not explicitly specified otherwise we use $\nu = \nu' = 5$ as the number of candidate stations which are considered in a repair or destroy step. We also use $k = 5$ as the number of candidates in a single round of the tournament selection used by the destroy and repair operators.

We evaluate the quality of solutions in terms of optimality gaps. More specifically, let f correspond to the objective value of the solution to some instance found by using any approach (i.e., construction heuristic, LNS or solving the MILP with Gurobi) within the time limit and let $\tilde{f}$ refer to the best found lower bound by Gurobi for the same instance. Then the gap between f and $\tilde{f}$ is calculated by

$$\text{gap} = 100\% \cdot \frac{f - \tilde{f}}{\tilde{f}}. \tag{6.3}$$

All shown results are averaged over all 30 instances for each instance group. Highlighted values refer to the best result per instance group and α_{delay} configuration, i.e., the lowest

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

gap, highest number of iterations and lowest repair time. The repair time always refers to the total time required from selecting the stations considered for repairing a solution, to the time needed to solve the respective MILP and further necessary steps to ensure that none of the posed constraints have been violated in the process, as presented in Algorithm 1. Further, for each instance the average repair time over all iterations is considered.

We will first show the results obtained with our initial construction heuristic. In comparison, we will show results of solving the BEXSLP-MILP formulation with Gurobi. Afterwards, the main part of this section concerns results obtained by using the matheuristic. Here, we first compare the performance of our single objective strategies, i.e., destroy and repair operators which focus on a single aspect of the multi-part objective. We will then present the results of the weighted sum strategy, compared to the those of using the different single objective operators within a single LNS run, i.e., the mixed strategy. Afterwards, we show how our dedicated strategies fare compared to a randomized approach. Then we justify the choice of our parameters before summarizing and discussing the most important results.

6.2.1 MILP and Construction Heuristic

First, we will present the results of solving the MILP formulation of the BEXSLP, (4.11) – (4.19), with Gurobi. We compare these results to the results obtained by using our construction heuristic (CH), introduced in Section 5.2.

Table 6.2 shows the average optimality gaps and corresponding median computation times obtained when using Gurobi to solve the MILP formulation of the BEXSLP compared to those obtained with our construction heuristic (CH). Column n_{opt} refers to the number of instances per instance group which could be solved to optimality with the MILP. It can be seen that for both approaches gaps increase with increasing instance size. Further, gaps generally also increase with increasing α_{delay} value.

Solving the MILP results in optimal solutions for all instances of the group $(50, 100)$ and almost all instances of the group group $(100, 200)$. For instance group $(100, 200)$ we can see that for $\alpha_{\mathrm{delay}} = 0.1$ all but one instances could be solved to optimality and for $\alpha_{\mathrm{delay}} = 10.0$ all but three instances.

However, gaps become significantly higher starting from instance group $(300, 600)$ for $\alpha_{\mathrm{delay}} = 0.1$ and already at instance group $(200, 400)$ for $\alpha_{\mathrm{delay}} = 1.0$ and $\alpha_{\mathrm{delay}} = 10.0$. It also becomes evident, that already for instance group $(200, 400)$ optimal solutions can only be achieved for $\alpha_{\mathrm{delay}} = 0.1$. The highest optimality gaps of 85.31% are obtained for the largest instance group $(500, 1000)$ when using $\alpha_{\mathrm{delay}} = 10.0$.
One can further see that the CH is actually able to achieve qualitative better solutions in shorter time for the majority of instances with $\alpha_{\mathrm{delay}} = 1.0$ and for the largest instances with $\alpha_{\mathrm{delay}} = 10.0$.

Regarding the reported median runtimes in Table 6.2, we can see that for instance groups starting from $(200, 400)$ the MILP solver terminates due to the specified time limit of 3600

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 6.2: Average optimality gaps and median computation times for different α_{delay} configurations obtained by using Gurobi to solve the MILP of the BEXSLP in comparison with our construction heuristic (CH). Column n_{opt} refers to the number of instances per instance group which could be solved to optimality with the MILP.

	gap (%)								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
(n, m)	n_{opt}	MILP	CH	n_{opt}	MILP	CH	n_{opt}	MILP	CH
(50, 100)	30	**0.00**	33.71	30	**0.00**	32.66	30	**0.00**	48.63
(100, 200)	29	**0.07**	34.53	24	**0.47**	31.45	27	**1.02**	55.05
(200, 400)	8	**4.44**	39.56	0	45.06	**41.03**	0	**61.23**	69.46
(300, 600)	1	**22.89**	39.41	0	54.95	**49.50**	0	**80.86**	81.91
(400, 800)	0	**30.05**	40.08	0	57.83	**50.71**	0	84.29	**83.81**
(500, 1000)	0	**37.61**	41.36	0	61.59	**53.42**	0	85.31	**84.72**

	run time (s)					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	MILP	CH	MILP	CH	MILP	CH
(50, 100)	**30.84**	53.56	232.76	**53.79**	122.03	**54.08**
(100, 200)	403.35	**81.52**	2058.73	**80.41**	1617.63	**71.75**
(200, 400)	3600.00	**194.12**	3600.00	**203.38**	3600.00	**150.53**
(300, 600)	3600.00	**413.74**	3600.00	**433.19**	3600.00	**271.91**
(400, 800)	3600.00	**713.32**	3600.00	**729.71**	3600.00	**423.44**
(500, 1000)	3600.00	**1006.51**	3600.00	**999.51**	3600.00	**642.34**

seconds before optimal solutions can be found. Looking at instances of the size $(50, 100)$ and $(100, 200)$ it becomes evident that solutions become in general more difficult to solve as α_{delay} increases. Concerning the CH we can see that run times increase according to the instance size, with the maximum run time taken for the largest instance size (500, 1000) with $\alpha_{\text{delay}} = 0.1$. It can generally be said that run times for $\alpha_{\text{delay}} = 0.1$ and $\alpha_{\text{delay}} = 1.0$ are relatively similar but typically noticeably larger than those achieved with $\alpha_{\text{delay}} = 10.0$.

A possible explanation could be the following. With growing α_{delay} the overall objective is more and more influenced by the delay part. In the CH we use relaxed x and y variables and it is possible that this relaxed version can be more efficiently solved, for example by constructing a large part of "fractional" stations to reduce the overall delay, than solutions for lower α_{delay} values. Thus, the overall solving time for $\alpha_{\text{delay}} = 10.0$ would be lower than for $\alpha_{\text{delay}} = 1.0$. However, for deriving feasible solutions we use the procedure in Section 5.2, i.e., we first round up fractional values and then heuristically remove surplus modules. This naturally somewhat decreases the quality of the solution. Naturally, this effect becomes more evident for larger α_{delay} values, as for these, the relaxed solutions

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

contain a larger number of fractional x and y variables. This would therefore explain, why we obtain larger gaps for larger α_{delay} values, despite the shorter run time of the CH.

Figures 6.1 and 6.2 further show a graphical comparison of the run times and optimality gaps between the MILP and the CH.

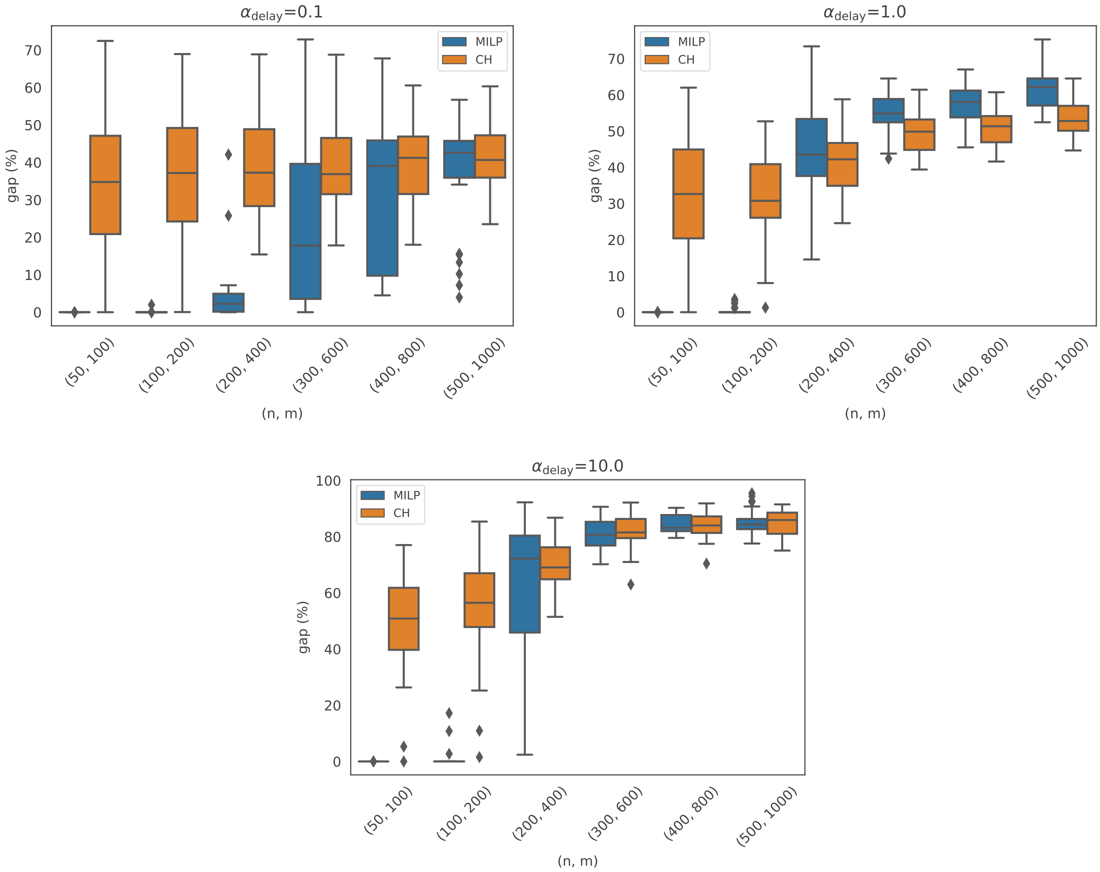

Figure 6.1: Comparison of optimality gaps of solving the BEXSLP with an MILP and our used construction heuristic (CH) w.r.t. different α_{delay} values.

6.2.2 Large Neighborhood Search

The main part of this section is dedicated to results obtained by using the full matheuristic, i.e., the initial CH results further refined by applying the presented operators in an LNS scheme. First, we present and compare results obtained by using single objective strategies, which focus on minimizing individual parts of the BEXSLP's multi-part objective. Then, we show two approaches which focus on all parts of the objective by comparing the strategy making use of the weighted sum operators to a strategy, which uses a randomly chosen single objective operator in every destroy and repair step. After comparing the most promising single and multi objective strategies, we compare our

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

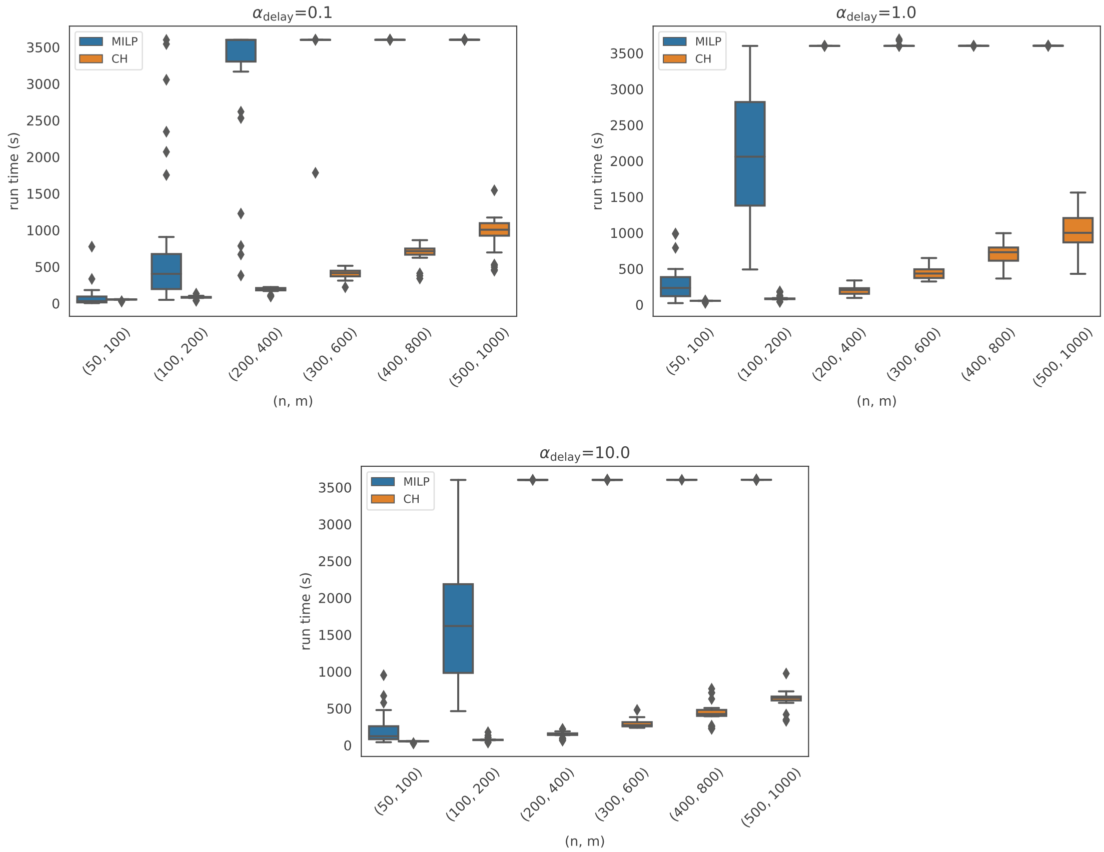

Figure 6.2: Comparison of run times of solving the BEXSLP with an MILP and our used construction heuristic (CH) w.r.t. different α_{delay} values.

results to a random LNS strategy. Finally, we give a summary by presenting the best results of the matheuristic in comparison to the initial CH and the MILP approach.

Single Objective Strategies

In this section we present results of our LNS in which only destroy and repair operators w.r.t. to a single objective of the BEXSLP's multi-part objective function are used. Specifically, we investigate three different LNS strategies, *constr*, *delay* and *charging* using only the construction-, delay- and charging-based destroy and repair operators, respectively.

Table 6.3 shows average optimality gaps, average iterations and median repair times for strategies *constr*, *delay* and *charging* w.r.t. the different α_{delay} configurations. We can see that similar to Table 6.2 optimality gaps generally increase with growing instance size and growing α_{delay} value for all three operators. For $\alpha_{\text{delay}} = 0.1$ *constr* performs superior to *delay* and *charging* for instance groups upwards of (200, 400). For instance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Table 6.3: Average optimality gaps, average number of iterations and median repair times for different α_{delay} configurations for the single objective strategies.

	gap (%)								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
(n, m)	*constr*	*delay*	*charging*	*constr*	*delay*	*charging*	*constr*	*delay*	*charging*
(50, 100)	2.73	3.01	**2.61**	6.54	**5.60**	5.76	12.77	**10.94**	12.05
(100, 200)	2.77	**1.97**	2.80	6.69	**5.61**	6.81	22.41	**18.27**	25.82
(200, 400)	**4.49**	5.72	5.49	**17.43**	18.65	21.31	41.89	**36.78**	47.47
(300, 600)	**5.13**	6.88	6.31	**28.41**	29.13	32.32	62.37	**59.42**	67.99
(400, 800)	**6.50**	8.62	8.39	**33.75**	33.96	36.63	71.48	**70.21**	74.49
(500, 1000)	**7.98**	10.77	10.68	**36.16**	37.03	39.99	74.80	**74.25**	77.59
	iterations								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
(n, m)	*constr*	*delay*	*charging*	*constr*	*delay*	*charging*	*constr*	*delay*	*charging*
(50, 100)	4898	6738	**7252**	4455	5608	**6381**	3858	6034	**6323**
(100, 200)	2897	4292	**4656**	2530	4855	**4953**	1613	**2991**	2672
(200, 400)	1707	2681	**2888**	1574	**2657**	2527	564	**988**	944
(300, 600)	1109	1829	**1906**	1009	1642	**1708**	398	**492**	403
(400, 800)	813	1261	**1284**	710	1138	**1175**	155	186	**192**
(500, 1000)	575	839	**853**	543	780	**783**	119	**149**	**149**
	repair time (s)								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
(n, m)	*constr*	*delay*	*charging*	*constr*	*delay*	*charging*	*constr*	*delay*	*charging*
(50, 100)	0.79	0.55	**0.50**	0.88	0.67	**0.59**	1.03	0.66	**0.63**
(100, 200)	1.44	0.89	**0.81**	1.75	**0.84**	0.86	2.66	**1.43**	1.55
(200, 400)	2.05	1.19	**1.09**	3.05	**1.72**	2.05	7.37	**4.72**	4.95
(300, 600)	2.78	1.57	**1.46**	7.02	3.71	**3.59**	15.67	**10.90**	11.74
(400, 800)	3.43	1.93	**1.89**	9.58	3.70	**3.50**	23.64	19.49	**18.59**
(500, 1000)	4.15	2.57	**2.51**	6.98	4.86	**4.83**	34.21	26.82	**25.87**

group (500, 1000) *constr* yields an optimality gap which is 2.7% lower than the second best operator, namely *charging*. For $\alpha_{\text{delay}} = 1.0$ *constr* still performs generally better than the other operators, however the relative difference to the other strategies, especially when compared to *delay*, decreases. When looking at $\alpha_{\text{delay}} = 10.0$ we can see that *delay* performs best for all instance groups. As expected, we can observe, that *delay* performs better, the higher α_{delay} is, i.e., as minimizing the delay becomes more important LNS operators destroying and repairing stations based on their induced delay produce better results.

When looking at the number of iterations it becomes evident that the number of iterations decreases as the size of an instances increases. While for the smallest instances several

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

thousand iterations can be achieved by every strategy for every α_{delay} configuration, this decreases to iterations in the range of hundreds for the largest instances. It can further be noticed that increasing α_{delay} generally leads to a lower number of iterations. For $\alpha_{\text{delay}} = 0.1$ and $\alpha_{\text{delay}} = 1.0$ the charging operator generally allows for the highest number of iterations while the construction operator yields the lowest number. For $\alpha_{\text{delay}} = 10.0$ the delay operator is generally favored except for instance groups (50, 100) and (400, 800).

Naturally, there is a correlation between the repair times and the number of achieved iterations. Therefore, we can see that *charging* typically has the lowest repair times for $\alpha_{\text{delay}} = 0.1$ and $\alpha_{\text{delay}} = 1.0$ and *delay* generally achieves the lowest repair times for $\alpha_{\text{delay}} = 10.0$. It is further interesting to see that the choice of α_{delay} seems to have a tremendous effect on the respective repair times. For instance group (500, 1000) repairing the solution takes up to 10 times as long as repairing a solution when setting $\alpha_{\text{delay}} = 0.1$.

A possible explanation for larger repair times concerning *constr* could be that this strategy is designed to select stations based on their construction costs per battery slot. In our generated test instances, construction costs of stations are not correlated to the maximum number of extension modules that may be added. This means, that *constr* may generally favor stations which allow to be extended by a large number of extension modules. Consequently, these stations can, if all extension modules were added, possibly be assigned a large amount of demand. This in turn makes it necessary for the MILP used for assigning demand to cover a larger amount of possibilities, i.e., how many extension modules to construct and more potential demand which can be assigned to every station, when considering stations selected by *constr* compared to the other strategies.

Multi Objective Strategies

The so far shown strategies all focus on a single part of our multi-part objective function. However, as the BEXSLP's multi objective function is the weighted sum of multiple individual objectives, a promising approach might be to combine our single objective strategies. One way to combine these strategies is to use the weighted sum destroy and repair operators described Section 5.3.5, resulting in the strategy *wsum*.

An alternative way to combine theses strategies is to use different destroy and repair operators in each iteration of the LNS. Specifically, in each iteration of the LNS we randomly choose either the delay, the construction, or the charging-based repair operator. The destroy operator is also randomly decided in each iteration w.r.t. the counterparts of the repair operators. We refer to this strategy as *mixed*. Note that in particular this means that within a single iteration a repair operator is not necessarily paired with the matching destroy operator.

Table 6.4 shows average gaps and iterations for the strategies *mixed* and *wsum*. One can see that for $\alpha_{\text{delay}} = 0.1$ and $\alpha_{\text{delay}} = 1.0$ the difference in terms of average optimality gaps is typically less than 1% with *mixed* being slightly more favored. For $\alpha_{\text{delay}} = 10.0$ the difference becomes more evident, as *mixed* achieves about 3% better results for

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 6.4: Average optimality gaps, average number of iterations and median repair times for different α_{delay} configurations for the mixed objective strategies.

	gap (%)					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*mixed*	*wsum*	*mixed*	*wsum*	*mixed*	*wsum*
(50, 100)	2.51	**2.42**	**5.84**	6.50	**8.87**	11.87
(100, 200)	2.72	**2.60**	**5.84**	6.06	**17.71**	20.15
(200, 400)	**3.34**	4.75	17.49	**17.30**	**38.52**	41.39
(300, 600)	5.07	**4.84**	**27.35**	28.35	**60.98**	62.21
(400, 800)	**6.59**	6.84	32.79	**32.78**	**70.15**	70.33
(500, 1000)	**8.16**	8.22	**36.30**	36.60	74.65	**74.06**

	iterations					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*mixed*	*wsum*	*mixed*	*wsum*	*mixed*	*wsum*
(50, 100)	**6685**	6162	**5572**	5220	**5966**	4608
(100, 200)	**3965**	3343	**4263**	2908	**2510**	1825
(200, 400)	**2565**	1976	**2271**	1640	**807**	779
(300, 600)	**1687**	1363	**1541**	1133	**458**	419
(400, 800)	**1157**	850	**1053**	795	**197**	176
(500, 1000)	**763**	603	**690**	589	**144**	131

	repair times (s)					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*mixed*	*wsum*	*mixed*	*wsum*	*mixed*	*wsum*
(50, 100)	**0.55**	0.60	**0.64**	0.73	**0.66**	0.83
(100, 200)	**0.95**	1.11	**0.94**	1.38	**1.69**	2.15
(200, 400)	**1.25**	1.68	**2.08**	2.98	**5.20**	6.13
(300, 600)	**1.69**	2.15	**3.73**	5.81	**11.41**	13.36
(400, 800)	**2.15**	3.18	**4.88**	7.97	**18.05**	21.83
(500, 1000)	**2.88**	3.83	**5.71**	7.22	**26.81**	30.66

instance groups (50, 100), (100, 200) and (200, 400) and only performs slightly worse for the largest size (500, 1000).

In terms of iterations it can be seen that *mixed* achieves a higher number of iterations for every instance group and every α_{delay} setting. This is most likely due to the way in which promising stations in the destroy/repair step are selected as combining the operators construction, delay, and charging-based operators in each iteration takes naturally more time than considering only a single operator. This also becomes evident when looking at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the median repair times, where for *mixed* less time is required than for *wsum* in every configuration. The reduced number of iterations when compared to *mixed* might also be an indicator for the slightly worse performance with regard to optimality gaps.

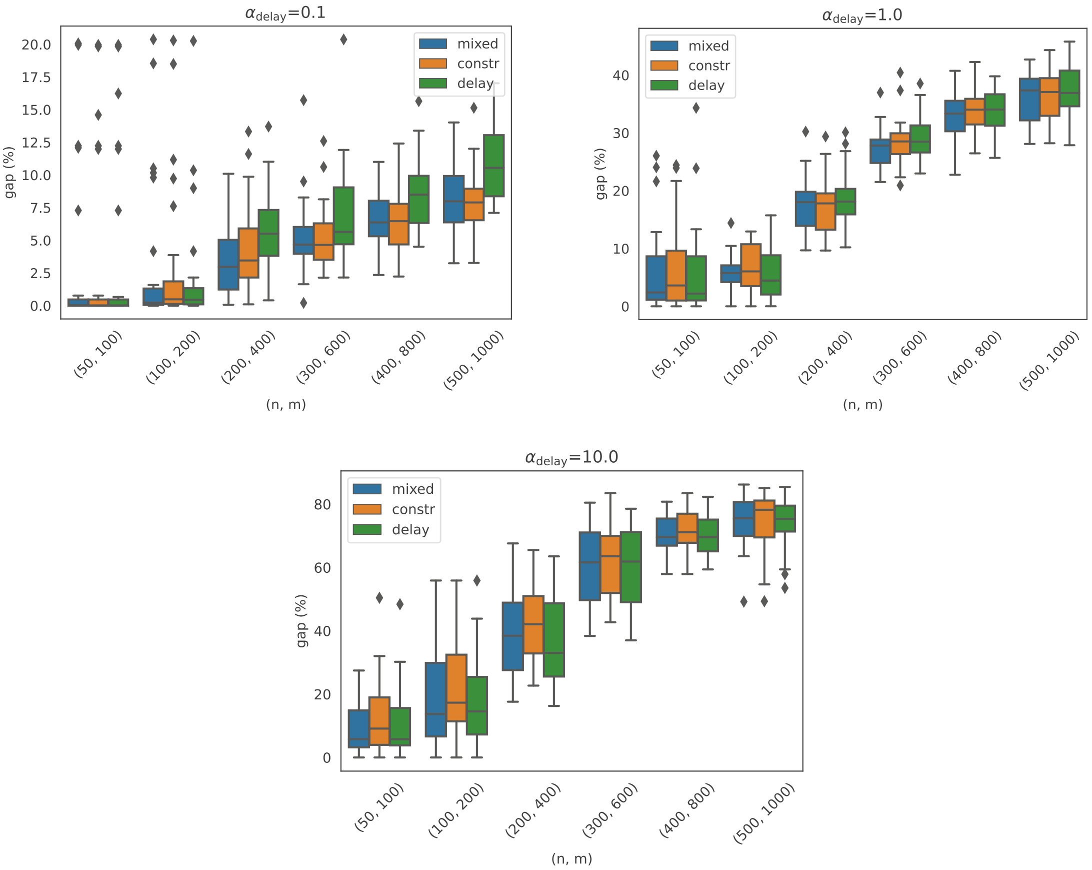

Figure 6.3: Comparison of *mixed*, *constr* and *delay* w.r.t. different α_{delay} values.

Figure 6.3 serves as a comparison of the most successful single objective strategies, *constr* and *delay* and the most promising multi-part objective strategy *mixed*. We have already established in Table 6.3 that *constr* performed best out of all single objective operators for $\alpha_{\mathrm{delay}} = 0.1$ and $\alpha_{\mathrm{delay}} = 1.0$. Here we see that for $\alpha_{\mathrm{delay}} = 0.1$ *mixed* achieves lower medians for instance groups (100, 200), (200, 400) and (400, 800) and matches *constr* for sizes (300, 600) and (500, 1000). When looking at $\alpha_{\mathrm{delay}} = 1.0$ we can see that *mixed* achieves lower medians than *constr* for all instance groups but (200, 400) and (500, 1000) and better results than *delay* for all instances but (100, 200) and (500, 1000).

When looking at $\alpha_{\mathrm{delay}} = 10.0$ we can again confirm that *delay* performs better than *constr* in this setting. However, *mixed* is able to match that performance for all instance groups but (200, 400) and achieves even lower optimality gaps for instance group (100, 200).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Comparison with the Randomized Strategy

We were further interested in investigating whether our dedicated approaches were more successful than a simple strategy, referred to as *random*, that constructs L_{destroy} and L'_{repair} completely at random. As we were interested in showing *statistical significance*, we performed a one-sided Wilcoxon signed-rank test [Con99] on the optimality gaps for each instance group comparing the solutions generated by *mixed* to the solutions generated by *random*.

Table 6.5 summarizes the results. Entries marked with a star denote results where a one-sided Wilcoxon signed-rank test has shown that a respective strategy performed statistically significantly better with a 95% confidence interval. We can see that for $\alpha_{\text{delay}} = 0.1$ and $\alpha_{\text{delay}} = 1.0$ the results for instance groups larger than (100, 200) w.r.t. *mixed* are significantly better than those w.r.t. *random*. For $\alpha_{\text{delay}} = 10.0$, this only holds true for instance groups (400, 800) and (500, 1000). A possible explanation for this is that for smaller instances a large number of iterations can be performed, resulting in smaller differences between *random* and *mixed*.

Performing the one-sided Wilcoxon signed-rank test the other way around, i.e., to test whether *random* is significantly better than *mixed*, w.r.t. a 95% confidence interval shows that this is only the case for instance group (50, 100) and $\alpha_{\text{delay}} = 1.0$.

Regarding the number of iterations, *random* generally achieves the largest number for all α_{delay} variants for the instance sizes less than (400, 800) with the exception of $\alpha_{\text{delay}} = 1.0$ and (300, 600).
This behavior can also be witnessed when looking at the repair times, where the randomized repair variant generally leads to smaller repair times.

Generally we would have expected *random* to always be faster than *mixed* as the procedure of selecting stations for *mixed* is more time consuming than for *random*. However, it has to be noted that the majority of the repair time can be attributed to solving the MILP which assigns the freed demand among the new station candidates. It is possible, that there is some inconsistency in the MILP solving times, which by chance simply leaned towards *mixed* for the larger instance groups.

Figure 6.4 further shows a graphical comparison of *mixed* with *random* regarding optimality gaps.

However, we have already seen in Figure 6.3 that *delay* is able to achieve better results for $\alpha_{\text{delay}} = 10.0$ than *mixed*. For this α_{delay} setting we have therefore also compared *delay* to *random* with regard to statistical significance. Table 6.6 shows the results of this comparison. We can see that with the *delay* strategy we also achieve significantly better results for instance group (300, 600). This indicates that the performance of *mixed* could potentially be improved by choosing the destroy and repair operators in a weighted random fashion instead of completely random in each iteration. However, finding appropriate weights for this is not straightforward and requires careful tuning and testing. Another possibility would be the usage of an Adaptive Large Neighborhood Search

Table 6.5: Average optimality gaps, number of iterations and median repair times for the strategies *mixed* and *random* w.r.t. different α_{delay} configurations. Entries marked with a star denote results where a one-sided Wilcoxon signed-rank test has shown that a respective strategy performed statistically significantly better than the other strategy w.r.t. a 95% confidence interval.

	gap (%)					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*random*	*mixed*	*random*	*mixed*	*random*	*mixed*
(50, 100)	2.74	**2.51**	***4.92**	5.84	9.14	**8.87**
(100, 200)	3.35	**2.72**	**5.69**	5.84	**17.52**	17.71
(200, 400)	4.99	***3.34**	19.14	***17.49**	**38.40**	38.52
(300, 600)	6.65	***5.07**	29.65	***27.35**	**60.97**	60.98
(400, 800)	7.67	***6.59**	34.55	***32.79**	71.52	***70.15**
(500, 1000)	10.44	***8.16**	38.25	***36.30**	75.59	***74.65**

	iterations					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*mixed*	*random*	Mixed	*random*	Mixed	*random*
(50, 100)	6685	**7416**	5772	**6673**	5966	**6951**
(100, 200)	3965	**4863**	4263	**5225**	2510	**3156**
(200, 400)	2565	**2625**	2271	**2439**	807	**874**
(300, 600)	1687	**1721**	**1541**	438	458	**473**
(400, 800)	**1157**	1059	**1053**	985	**197**	166
(500, 1000)	**763**	702	**690**	605	**144**	132

	repair times (s)					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*mixed*	*random*	Mixed	*random*	Mixed	*random*
(50, 100)	0.55	**0.50**	0.64	**0.57**	0.66	**0.57**
(100, 200)	0.95	**0.76**	0.94	**0.77**	1.69	**1.36**
(200, 400)	1.25	**1.22**	2.08	**1.82**	5.20	**4.91**
(300, 600)	**1.69**	1.71	**3.73**	4.52	**11.41**	11.47
(400, 800)	**2.15**	2.44	4.88	**4.33**	**18.05**	22.25
(500, 1000)	**2.88**	3.22	5.71	6.07	**26.81**	31.28

(ALNS) [GP+10]. In an ALNS different weights are assigned to different repair/destroy methods. As the ALNS progresses, weights are adapted dynamically according to the success of the respectively applied method. We did not investigate these techniques further in this work, as ALNS typically requires a large number of iterations, which we do not reach for our largest instances. Finally, it may also be that the individual operators construction, charging and delay, work against each other in *mixed*. Therefore, the use of a tabu list might potentially also improve the performance of *mixed*. We investigate the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

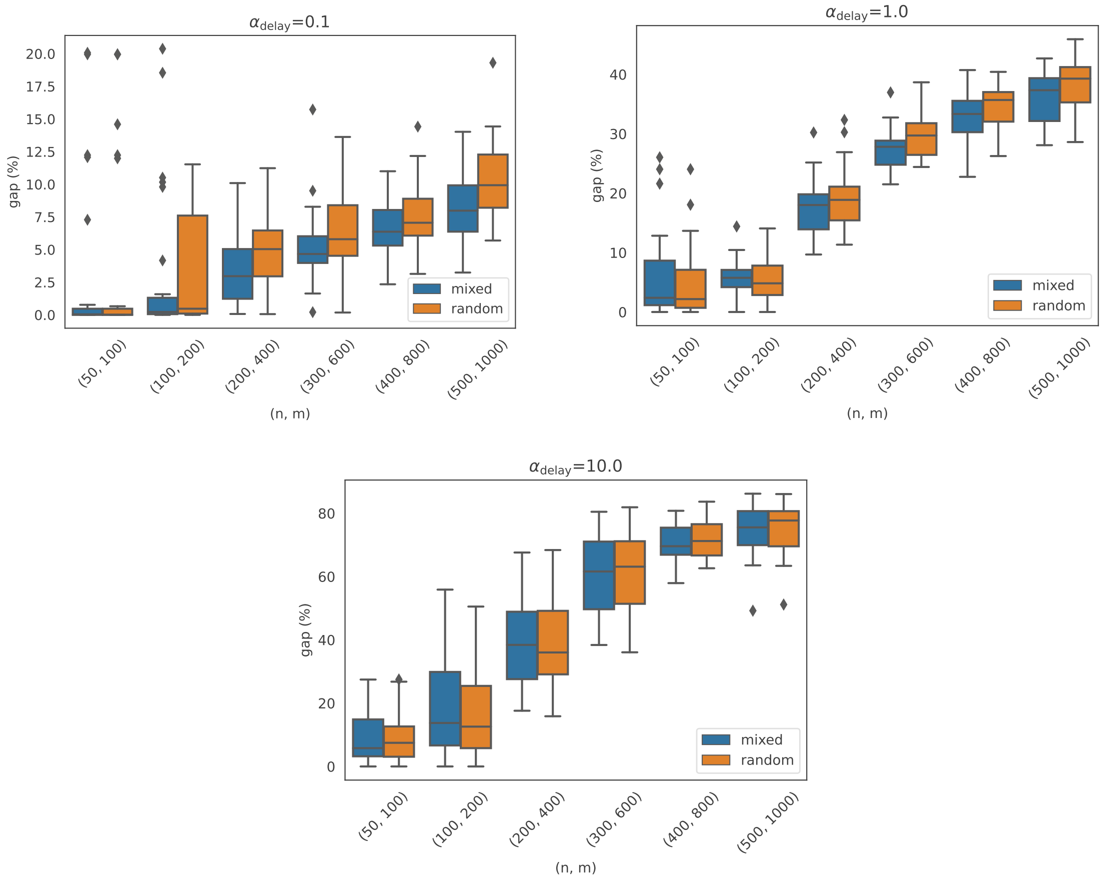

Figure 6.4: Comparison of *mixed* with randomized approaches w.r.t. different α_{delay} values.

use of a tabu list in Section 6.2.2.

We next show how solutions are improved over time. Towards this, Figure 6.5 shows a comparison of the strategies *mixed* and *random* with regard to how the solution is improved by the LNS over time. The plots show the development of the optimality gaps over our selected run time of 3600 seconds averaged over all instances in the respective instance group. To be able to aggregate over all instances at each time, we always use the current best objective value at each time to calculate the respective optimality gaps. Additionally, for the beginning, when no solution exists yet, we always use the solution returned by the construction heuristic. This can be observed in Figure 6.5 as in most plots the gaps do not immediately improve.

We can see that for smaller instances, the initial solution as well as the overall local optimum, can be found within a very short time frame. As the instance group increases, both the initial solution, as well as further improvement of this solution takes considerable more time. Starting from instance group (200, 400), we can however already see that

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Table 6.6: Comparison of gaps between *delay* and *random* for $\alpha_{\text{delay}} = 10.0$. Entries marked with a star denote results where a one-sided Wilcoxon signed-rank test has shown that a respective strategy performed statistically significantly better than the other strategy w.r.t. a 95% confidence interval.

	gap (%)	
	$\alpha_{\text{delay}} = 10.0$	
(n, m)	*random*	*delay*
(50, 100)	9.14	**8.87**
(100, 200)	**17.52**	18.27
(200, 400)	38.40	**36.78**
(300, 600)	60.97	***59.42**
(400, 800)	71.52	***70.21**
(500, 1000)	75.59	***74.25**

in general *mixed* improves the solution faster, and as has already been shown, tends to find better solutions overall. This difference between *mixed* and *random* increases with increasing instance size and is more noticeable for $\alpha_{\text{delay}} = 0.1$ and $\alpha_{\text{delay}} = 1.0$. We can further see again that solution improvement generally slows down with increasing α_{delay} value, most notable in this figure by the curve getting flatter with increasing α_{delay} setting, i.e., for $\alpha_{\text{delay}} = 10.0$ the solution converges considerably slower towards a local optimum.

Choice of k and ν

As we have stated at the beginning of this section, we used $k = 5$ for the group size in the tournament selection which is used in the delay, charging and in turn also the mixed operator. Further, we decided on $\nu = \nu' = 5$ as the number of stations which we destroy and select for repairing in our operators. Here we want to argue why we settled on these configurations.

Table 6.7 shows results for different configurations with regard to k and ν for *mixed*. Note that in all configurations we assume that $\nu = \nu'$. The configuration $k = 5, \nu = 5$ is the configuration which was used for the other presented results in this section.

We can see that the configuration $k = 5, \nu = 5$ performs best in terms of iterations and in most cases, also for median repair times. We can see that, as expected, increasing k to 10 increases the repair times and in turn decreases the number of iterations.

We can further see that increasing ν to 10 significantly increases repair times and in turn leads to a much lower number of iterations. This was as expected, as we use a MILP to repair the solution based on the set of repair candidates. Therefore, if we destroy a

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

larger set of stations and then repair the solution based on a larger set of candidates, the MILP becomes more complex and in turn takes more time to solve.

Regarding the optimality gaps we can further see that the configuration $k = 5, \nu = 5$ generally also performs best with the exception of instance group (300, 600) for $\alpha_{\text{delay}} = 1.0$ and $\alpha_{\text{delay}} = 10.0$, instance group (400, 800) for $\alpha_{\text{delay}} = 0.1$ and instance group (200, 400) for $\alpha_{\text{delay}} = 10.0$. In general the relative difference between the configurations seem to be larger for the smaller instances and are decreasing with increasing instance size.

Table 6.7: Average optimality gaps, average iterations and median repair times for *mixed* w.r.t. different k, ν and α_{delay} settings.

	gap (%)								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
	$k = 10$	$k = 5$		$k = 10$	$k = 5$		$k = 10$	$k = 5$	
(n, m)	$\nu = 5$	$\nu = 5$	$\nu = 10$	$\nu = 5$	$\nu = 5$	$\nu = 10$	$\nu = 5$	$\nu = 5$	$\nu = 10$
(50, 100)	2.77	**2.51**	3.30	6.76	**5.84**	7.42	10.21	**8.87**	11.26
(100, 200)	4.01	**2.72**	3.60	5.88	**5.84**	7.80	18.24	**17.71**	19.93
(200, 400)	4.26	**3.34**	5.18	17.97	**17.49**	17.51	**37.42**	38.52	38.94
(300, 600)	5.15	**5.07**	5.63	28.61	27.35	**27.20**	61.65	60.98	**59.57**
(400, 800)	**6.30**	6.59	6.65	33.25	**32.79**	32.87	70.22	**70.15**	70.27
(500, 1000)	8.27	**8.16**	8.61	36.41	**36.30**	36.98	74.83	**74.65**	74.88

	gap (%)								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
	$k = 10$	$k = 5$		$k = 10$	$k = 5$		$k = 10$	$k = 5$	
(n, m)	$\nu = 5$	$\nu = 5$	$\nu = 10$	$\nu = 5$	$\nu = 5$	$\nu = 10$	$\nu = 5$	$\nu = 5$	$\nu = 10$
(50, 100)	5927	**6685**	2636	5358	**5772**	1212	4764	**5966**	923
(100, 200)	3614	**3965**	986	3578	**4263**	643	2070	**2510**	731
(200, 400)	2080	**2565**	685	1944	**2271**	473	722	**807**	364
(300, 600)	1484	**1687**	457	1300	**1541**	364	434	**458**	198
(400, 800)	1053	**1157**	332	1011	**1053**	296	180	**197**	95
(500, 1000)	723	**763**	260	660	**690**	237	144	**144**	61

	gap (%)								
	$\alpha_{\text{delay}} = 0.1$			$\alpha_{\text{delay}} = 1.0$			$\alpha_{\text{delay}} = 10.0$		
	$k = 10$	$k = 5$		$k = 10$	$k = 5$		$k = 10$	$k = 5$	
(n, m)	$\nu = 5$	$\nu = 5$	$\nu = 10$	$\nu = 5$	$\nu = 5$	$\nu = 10$	$\nu = 5$	$\nu = 5$	$\nu = 10$
(50, 100)	0.63	**0.55**	2.97	0.72	**0.64**	5.13	**0.83**	0.66	4.60
(100, 200)	1.05	**0.95**	5.55	1.13	**0.94**	7.20	**1.95**	1.69	6.06
(200, 400)	1.60	**1.25**	6.49	2.39	**2.08**	8.59	**5.75**	5.20	10.72
(300, 600)	1.92	**1.69**	7.91	4.65	**3.73**	11.88	**13.41**	11.41	20.18
(400, 800)	2.38	**2.15**	10.79	4.77	**4.88**	14.91	**19.97**	18.05	35.26
(500, 1000)	3.05	**2.88**	11.07	5.67	**5.71**	18.05	**25.77**	26.81	54.40

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

We argue that further increasing ν and k would only lead to a further decrease of the number of iterations until an insufficient number of iterations would be achieved. On the other hand, further lowering of k would lead the mixed operator ever further towards a randomized approach, which generally does not improve performance as seen in Table 6.5. We also think that a lower size for ν would not further improve the performance, as with $\nu = 5$ the MILP used for repairing solutions performs reasonably fast, already achieving a sufficient number of iterations.

Tabu List

An intuitive idea to improving the quality of the solutions further would be to aid the respective operators in their task of choosing promising stations to destroy or repair. For *mixed* we combine different operators that allow aim to optimize different parts of the BEXSLP's objective. It would be undesirable if operators were to select the same stations to be destroyed that have just been added to the solution, due to operators aiming for conflicting goals. We therefore tested a strategy based on tabu search [GL98, GP14] for the strategy *mixed*.

The idea of our approach is to lock stations which have recently been selected from being selected again. Specifically, we aim to prevent stations that have just been added to the solution from being destroyed and vice versa. For this we use two separate tabu lists, one for destroy operators and one for repair operators. Stations may only be selected if they are currently not locked by the respective tabu list.

We have implemented the two tabu lists as FIFO-Queues with a fixed length of 5. Therefore, stations in the tabu list cannot be select in the following 5 iterations. In each iteration we add one station from L_{destroy} to the repair tabu list and one station from L'_{repair} to the destroy tabu list. Selection is performed randomly weighted by the number of times each station has already been in the destroy and repair set, respectively.

Table 6.8 summarizes the results. We can see that except for instance group (50, 100) with $\alpha_{\text{delay}} = 1.0$ and instance group (200, 400) with $\alpha_{\text{delay}} = 10.0$ the tabu search did not improve the performance. The differences in performance are generally smallest for $\alpha_{\text{delay}} = 0.1$ and increase with increasing α_{delay} value.

A possible explanation for this could be that with increasing α_{delay} value solutions with a larger number of constructed stations tend to be favourable, as a widespread network of constructed stations generally decreases the detours that customers have to take to reach a station. In this scenario it may be the case that restricting the set of stations to choose from, e.g., by using the tabu list therefore actually has a negative impact on the overall solution quality.

For smaller α_{delay} values a possible explanation could be that the mixed operator by using different operators, having different metrics for choosing stations, inherently already selects different stations in every iteration anyhow and is therefore not positively affected by the tabu list.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

As the proposed tabu procedure did not improve the performance in our case, we ultimately decided on presenting and refining results without the usage of a tabu list.

Table 6.8: Optimality gaps for *mixed* compared to our tested tabu list w.r.t. different α_{delay} settings.

	gap (%)					
	$\alpha_{\text{delay}} = 0.1$		$\alpha_{\text{delay}} = 1.0$		$\alpha_{\text{delay}} = 10.0$	
(n, m)	*mixed*	*mixed* + tabu	*mixed*	*mixed* + tabu	*mixed*	*mixed* + tabu
(50, 100)	**2.51**	3.23	5.84	**5.27**	**8.87**	10.20
(100, 200)	**2.72**	2.92	**5.84**	6.17	**17.71**	18.76
(200, 400)	**3.34**	4.49	**17.49**	18.12	38.52	**38.18**
(300, 600)	**5.07**	5.55	**27.35**	28.48	**60.98**	61.82
(400, 800)	**6.59**	6.76	**32.79**	33.79	**70.15**	71.76
(500, 1000)	**8.16**	9.05	**36.30**	38.44	**74.65**	76.19

6.2.3 Overview Comparison of all Approaches

Summarizing, Figure 6.6 and Table 6.9 give an overview of the results obtained for different approaches towards solving the BEXSLP. MILP denotes the results of solving the BEXSLP with the MILP model, (4.11) – (4.19), with Gurobi. CH refers to the results obtained from the initial construction heuristic, presented in Section 5.2 and *random* and *mixed* refer to the LNS with the random or respectively the mixed strategy.

It becomes evident that with the MILP approach we are able to find (close to) optimal solutions for the smallest instance sizes (50, 100) and (100, 200) for every α_{delay} configuration. However, starting from (200, 400) our LNS approach is able to consistently achieve superior results. For $\alpha_{\text{delay}} = 0.1$ *mixed* achieves gaps which are about 16% lower than those achieved by the MILP approach for instance group (300, 600). For instance group (500, 1000) we obtain results being 29% lower than those obtained by the MILP approach. For $\alpha_{\text{delay}} = 1.0$ we are able to improve on the MILP approach by $25 - 28\%$ when using *mixed* for instances larger than (100, 200). Also for $\alpha_{\text{delay}} = 10.0$ we are able to achieve results which are up to 23% better when using the LNS with *mixed* compared to the MILP results. In this setting the difference decreases with growing instance size. However, for instance group (500, 1000) *mixed* is still better by about 10%, however.

It is also interesting to note that in some cases the construction heuristic is already able to achieve better results than the MILP approach, for example for instance groups larger than (100, 200) for $\alpha_{\text{delay}} = 1.0$. As specified in Section 5.2, we use a numerically relaxed version of the MILP formulation with regard to fractional x and y variables for the CH which we afterwards repair to guarantee a feasible solution. It is therefore possible that by performing our procedure to ensure a feasible solution we already achieve better

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

solutions than the best exact solution which can be found by Gurobi within the specified time limit.

However, we can see that the LNS approach further improves the initial solution obtained by the construction heuristic significantly. For $\alpha_{\text{delay}} = 0.1$, the LNS improves the initial solution by up to 36%. The relative improvement however decreases somewhat with instance size. The least improvement of the initial solution by the LNS can be noted for the largest instances and the $\alpha_{\text{delay}} = 10$ value. However, even for the largest instance size in this configuration we are still able to improve this initial solution by 10%

We can further see that *mixed* performs better than the random approach for all instance groups when setting $\alpha_{\text{delay}} = 0.1$. For $\alpha_{\text{delay}} = 1.0$ *mixed* achieves superior results for instances larger than (100, 200). For $\alpha_{\text{delay}} = 10.0$ *mixed* achieves better results for the largest instance sizes (400, 800) and (500, 1000) and gaps are only marginally higher for the smaller instances.

Table 6.9: Average optimality gaps for the MILP, the presented construction heuristic (CH), *random* and *mixed* for different α_{delay} settings.

	gap (%)											
	$\alpha_{\text{delay}} = 0.1$				$\alpha_{\text{delay}} = 1.0$				$\alpha_{\text{delay}} = 10.0$			
(n, m)	MILP	CH	*random*	*mixed*	MILP	CH	*random*	*mixed*	MILP	CH	*random*	*mixed*
(50, 100)	**0.00**	33.71	2.74	2.51	**0.00**	32.66	4.92	5.84	**0.00**	48.63	9.14	8.87
(100, 200)	**0.07**	34.53	3.35	2.72	**0.47**	31.45	5.69	5.84	**1.02**	55.05	17.52	17.71
(200, 400)	4.44	39.56	4.99	**3.34**	45.06	41.03	19.14	**17.49**	61.23	69.46	**38.40**	38.52
(300, 600)	22.89	39.41	6.65	**5.07**	54.95	49.50	29.65	**27.35**	80.86	81.91	**60.97**	60.98
(400, 800)	30.05	40.08	7.67	**6.59**	57.83	50.71	34.55	**32.79**	84.29	83.81	71.52	**70.15**
(500, 1000)	37.61	41.36	10.44	**8.16**	61.59	53.42	38.25	**36.30**	85.31	84.72	75.59	**74.65**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

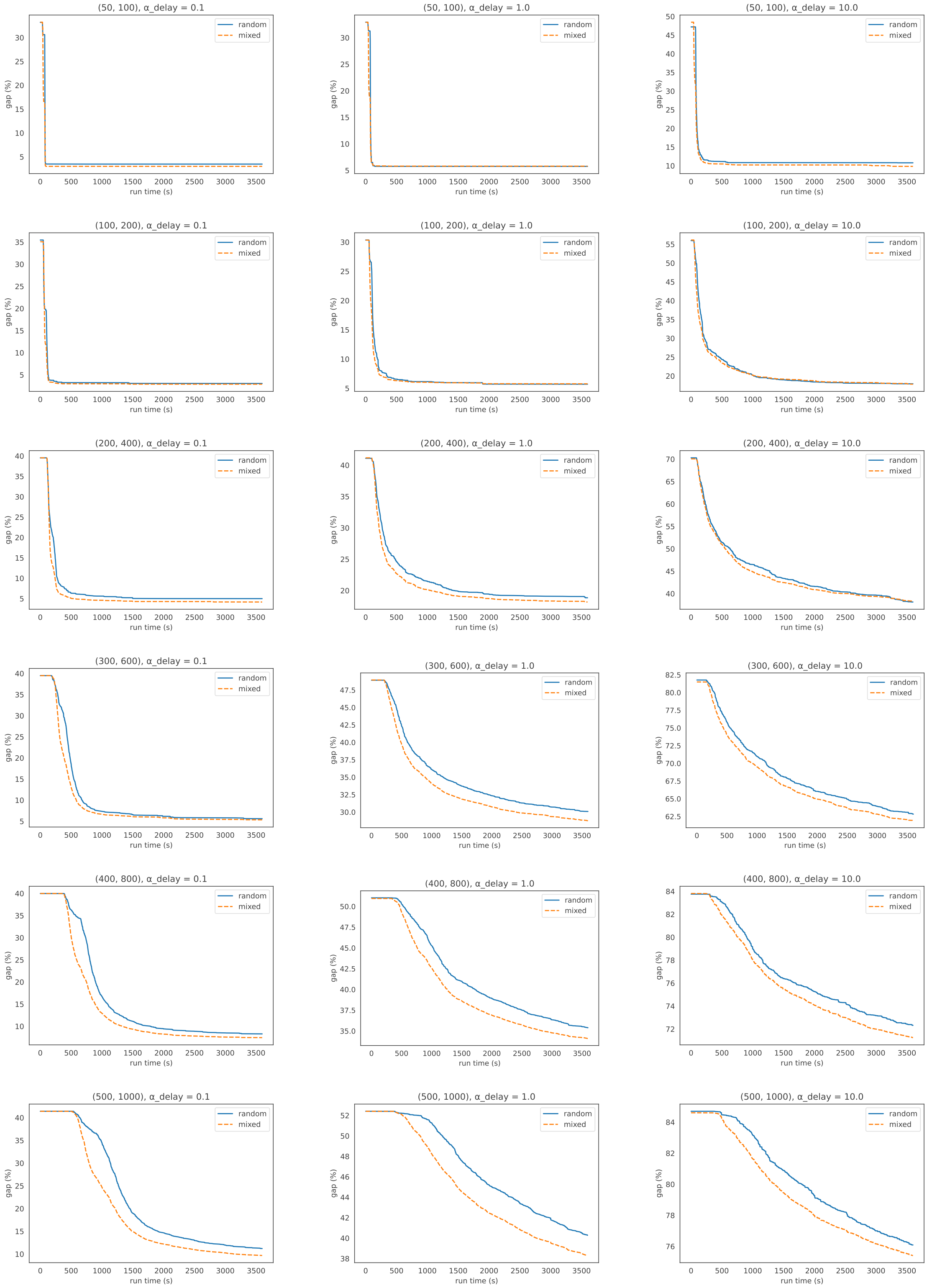

Figure 6.5: Comparison of how solutions are iteratively improved by *random* and *mixed* w.r.t. different α_{delay} configurations and instance groups.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

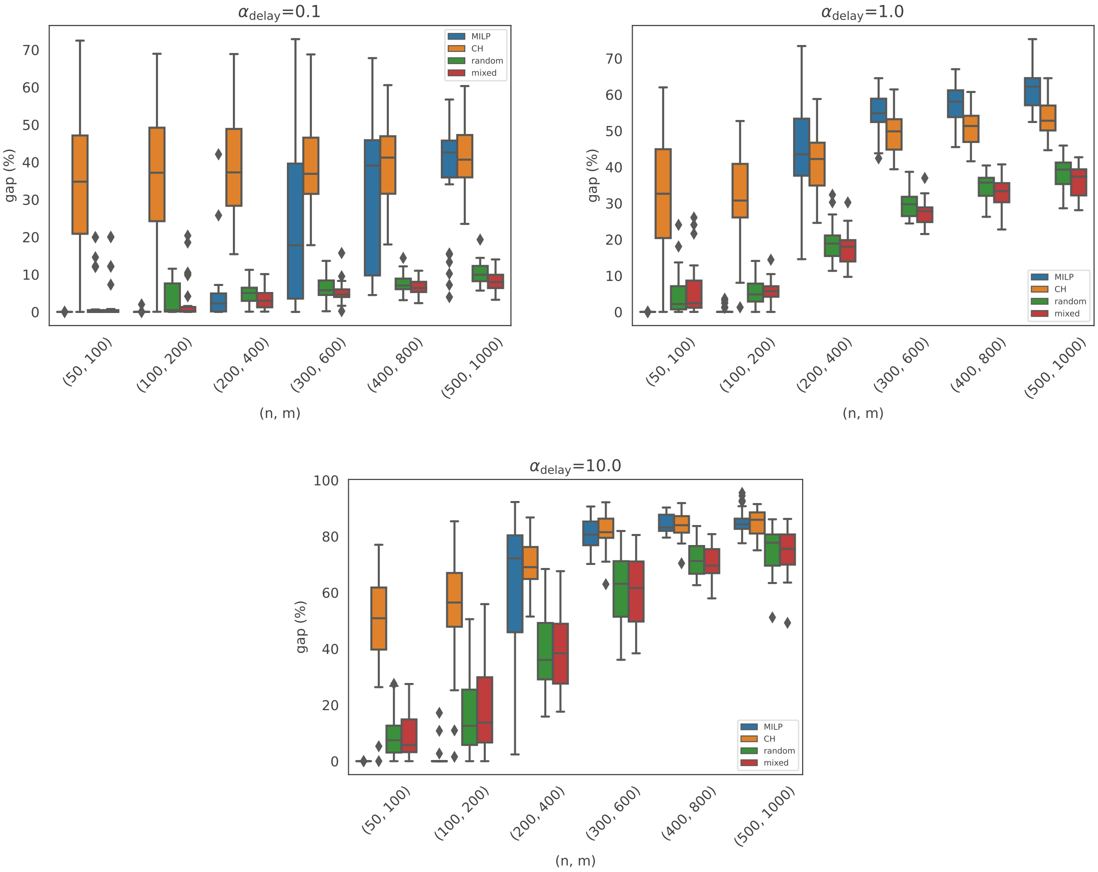

Figure 6.6: Comparison of solving the BEXSLP with an MILP, the presented construction heuristic (CH), *random* and *mixed* with w.r.t. different α_{delay} values.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 7

Conclusion and Future Work

Although the adoption of electric vehicles has increased in the past years and is expected to further grow, long charging times may be a hindering factor to a wide-spread usage. An alternative idea, at least for smaller electric vehicles, like scooters, is to use exchangeable batteries and to allow users to swap their batteries very quickly at dedicated stations.

In this work, we dealt with the *Battery Exchange Station Location Problem* (BEXSLP), which concerns the planning of such stations.

We defined the BEXSLP as a mixed integer linear program (MILP) with a multi-part objective, which is to be minimized, concerning the construction costs, charging costs and delays induced by customers driving to the stations, while satisfying a defined amount of demand. Towards solving the problem, we proposed and implemented a matheuristic, combining the exact solving capabilities of MILP solvers with the scalability of heuristic approaches. In our matheuristic we first apply a construction heuristic to obtain an initial solution and then use a Large Neighborhood Search (LNS), based on a destroy and repair scheme, to refine the solution. In each iteration of the LNS, we destroy a set of stations and select a new set of promising stations to repair the solution. In the construction heuristic and when repairing the solution we use a linear relaxation of the presented MILP which allows us to solve the respective problem much faster. We further presented the necessary steps to derive a feasible solution from this relaxed solution.

Towards selecting promising stations when destroying and repairing a solution we proposed several operators, which focus on different parts of the multi-part objective. We further showed two possibilities to combine multiple aspects (construction costs, charging costs, induced delay) of the objective in a single approach. The mixed strategy selects a different single-objective operator in each destroy and repair step and thus makes use of all operators within a single run of the LNS. The weighted sum strategy uses a linear combination of the objectives in every iteration.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

For evaluating the proposed matheuristic we created a novel set of test instances based on approaches from literature. Early experiments showed that the problem difficulty changes drastically depending on how much focus is laid on the delay part of the objective. We therefore evaluated our matheuristic on different problem settings regarding delay, and, for comparison purposes, evaluated the MILP formulation of the BEXSLP with Gurobi. The results show that it is possible to find close to optimal solutions with the MILP approach for the very small instances. For larger instances, however, our matheuristic far surpasses the MILP approach in terms of solution quality in every evaluated problem setting and achieves between ten to thirty percent lower optimality gaps.

We have further seen that while all evaluated operators perform better than the MILP approach for larger instances, the most success was achieved by combining all single-objective operators within a single LNS run, i.e., the mixed strategy.

7.1 Future Work

We have shown that the matheuristics outperforms the MILP approach in all benchmark settings for larger instances. However, it became evident that a large focus on the delay part of the objective also proved to be most burdensome for our approach. The results here indicate further possibility for improvement. We noticed that while the mixed strategy generally performed best out of the presented approaches, there were instances where the delay strategy, focusing solely on optimizing the delay part of the objective, proved to be slightly better. Possible future work in this regard therefore might be to use a combined operator which, unlike the mixed strategy, does not pick the respective operators completely at random but in a weighted random fashion. One possibility would be to use an Adaptive Large Neighborhood Search (ALNS) which assigns starting weights to the operators and dynamically adapts the weights based on their respective performance. We however believe that finding appropriate weights to be an intricate task which requires an adequate amount of fine tuning.

Regarding the use of a matheuristic, solving the BEXSLP is of course not limited to our proposed approach. An interesting idea might be to use a genetic algorithm (GA) as an alternative metaheuristic framework to our LNS. The GA could be used to select promising stations and a MILP model solved for assigning demand to those stations. In the same fashion could machine learning techniques be employed, which may over time learn which station/location placements prove most promising towards finding a good solution.

Our toolkit used for generating test instances also allows creation of vastly different scenarios which undoubtedly influence the complexity of the problem. We settled on a configuration which encapsulates a lot of aspects of the BEXSLP within a single instance set, in what we believe to be a realistic fashion.

It could also be interesting to research adapted variants of the BEXSLP. For example, it might be interesting to also consider limited charging times, in a similar fashion as

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

we consider times where users can exchange batteries. It would also possible to specify a constraint enforcing an overall budget for building stations and modules. By this, one could for example look for solutions which minimize the delay, but, in terms of construction costs, do not exceed a set amount of money. It might also be interesting to specify a set of locations where competitors have constructed battery exchange stations. One could then model the problem to enforce a minimal or maximal distance to each competing location when planning the own set of locations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliothek, Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek, WIEN Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[AS14] Claudia Archetti and M Grazia Speranza. A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246, 2014.

[BF12] Alireza Boloori Arabani and Reza Zanjirani Farahani. Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1):408–420, 2012.

[BLF92] Oded Berman, Richard C Larson, and Nikoletta Fouska. Optimal location of discretionary service facilities. *Transportation Science*, 26(3):201–211, 1992.

[BPRR11] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied soft computing*, 11(6):4135–4151, 2011.

[BR16] Christian Blum and Günther R Raidl. *Hybrid Metaheuristics: Powerful Tools for Optimization*. Springer, 2016.

[BT97] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific, 1997.

[CBW17] Makena Coffman, Paul Bernstein, and Sherilyn Wee. Electric vehicles revisited: a review of factors that affect adoption. *Transport Reviews*, 37(1):79–93, 2017.

[CGI+20] Herminia I Calvete, Carmen Galé, José A Iranzo, José-Fernando Camacho-Vallejo, and Martha-Selene Casas-Ramírez. A matheuristic for solving the bilevel approach of the facility location problem with cardinality constraints and preferences. *Computers & Operations Research*, 124, 2020.

[CKLT13] Ismail Capar, Michael Kuby, V. Jorge Leon, and Yu-Jiun Tsai. An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. *European Journal of Operational Research*, 227(1):142 – 151, 2013.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[Con99] William Jay Conover. *Practical nonparametric statistics*, volume 350. john wiley & sons, 1999.

[CPvdV02] Richard K Congram, Chris N Potts, and Steef L van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.

[CR74] Richard Church and Charles ReVelle. The maximal covering location problem. In *Papers of the regional science association*, volume 32, pages 101–118. Springer, 1974.

[Dan51] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1951.

[DS10] Karl F Doerner and Verena Schmid. Survey: Matheuristics for rich vehicle routing problems. In *International Workshop on Hybrid Metaheuristics*, volume 6373 of *(LNCS*, pages 206–221. Springer, 2010.

[GL98] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[Gon12] Jacek Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[GP+10] Michel Gendreau, Jean-Yves Potvin, et al. *Handbook of Metaheuristics*, volume 2. Springer, 2010.

[GP14] Michel Gendreau and Jean-Yves Potvin. Tabu search. In *Search methodologies*, pages 243–263. Springer, 2014.

[GZN16] Mehrnaz Ghamami, Ali Zockaie, and Yu Marco Nie. A general corridor model for designing plug-in electric vehicle charging infrastructure to support intercity travel. *Transportation Research Part C: Emerging Technologies*, 68:389–402, 2016.

[HMH17] Meysam Hosseini, Seyyed Ali MirHassani, and Farnaz Hooshmand. Deviation-flow refueling location problem with capacitated facilities: Model and algorithm. *Transportation Research Part D: Transport and Environment*, 54:269–281, 2017.

[Hod90] M. John Hodgson. A flow-capturing location-allocation model. *Geographical Analysis*, 22(3):270–279, 1990.

[HRZ96] M. John Hodgson, Kenneth E. Rosing, and Jianjun Zhang. Locating vehicle inspection stations to protect a transportation network. *Geographical Analysis*, 28(4):299–314, 1996.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[JORR20] Thomas Jatschka, Fabio F. Oberweger, Tobias Rodemann, and Günther R. Raidl. Distributing battery swapping stations for electric scooters in an urban area. In Nicholas Olenev, Yuri Evtushenko, Michael Khachay, and Vlasta Malkova, editors, *Optimization and Applications*, volume 12422 of *LNCS*, pages 150–165. Springer, 2020.

[KÇ18] Merve Keskin and Bülent Çatay. A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172–188, 2018.

[KK17] Sadan Kulturel-Konak. A matheuristic approach for solving the dynamic facility layout problem. *Procedia Computer Science*, 108:1374–1383, 2017.

[KL05] Michael Kuby and Seow Lim. The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences*, 39(2):125–145, 2005.

[KM78] Ravindran Kannan and Clyde L Monma. On the computational complexity of integer programming problems. In *Optimization and Operations Research*, volume 157 of *LNE*, pages 161–172. Springer, 1978.

[LGC+16] Carolina Lagos, Guillermo Guerrero, Enrique Cabrera, Stefanie Niklander, Franklin Johnson, Fernando Paredes, and Jorge Vega. A matheuristic approach combining local search and mathematical programming. *Scientific Programming*, 2016, 2016.

[LLCG17] Wenbo Li, Ruyin Long, Hong Chen, and Jichao Geng. A review of factors influencing consumer intentions to adopt battery electric vehicles. *Renewable and Sustainable Energy Reviews*, 78:318–328, 2017.

[LNdG19] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama. *Location science*. Springer, 2019.

[LW66] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[ME13] S. A. MirHassani and R. Ebrazi. A flexible reformulation of the refueling station location problem. *Transportation Science*, 47(4):617–628, 2013.

[MSV10] Vittorio Maniezzo, Thomas Stützle, and Stefan Voss. *Matheuristics – Hybridizing Metaheuristics and Mathematical Programming*. Springer, 2010.

[NSdG15] Stefan Nickel and Francisco Saldanha-da Gama. Multi-period facility location. In *Location science*, pages 289–310. Springer, 2015.

[Pap81] Christos H Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[PR05] Jakob Puchinger and Günther R Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *International work-conference on the interplay between natural and artificial computation*, volume 3562 of *LNCS*, pages 41–53. Springer, 2005.

[PRP09] Jakob Puchinger, Günther R. Raidl, and Sandro Pirkwieser. Metaboosting: enhancing integer programming techniques by metaheuristics. *Matheuristics*, pages 71–102, 2009.

[RHL20] Nele Rietmann, Beatrice Hügler, and Theo Lieven. Forecasting the trajectory of electric vehicle sales and the consequences for worldwide CO2 emissions. *Journal of Cleaner Production*, 261, 2020.

[RND16] Achmad P Rifai, Huu-Tho Nguyen, and Siti Zawiah Md Dawal. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. *Applied Soft Computing*, 40:42–57, 2016.

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. Wiley, 1998.

[SH13] Pierre Schaus and Renaud Hartert. Multi-objective large neighborhood search. In *Principles and Practice of Constraint Programming*, volume 8124 of *LNCS*, pages 611–627. Springer, 2013.

[TSC21] Renata Turkeš, Kenneth Sörensen, and Daniel Palhazi Cuervo. A matheuristic for the stochastic facility location problem. *Journal of Heuristics*, 27:649–694, 2021.

[Wol20] Laurence A Wolsey. *Integer programming*. Wiley, 2020.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.