Katholische Universität
Eichstätt-Ingolstadt

Innovative Ansätze und Metaheuristiken für Zustellprobleme der letzten Meile

Innovative Approaches and Metaheuristics
for Last Mile Delivery Problems

Stefan Voigt

Kumulative Dissertation
zur Erlangung des akademischen Grades
Doctor rerum politicarum (Dr. rer. pol.)

Juni 2022

Referent: Prof. Dr. Heinrich Kuhn
Korreferent: Prof. Dr. Manuel Ostermeier

Danksagung

Die vorliegende Dissertation habe ich während meiner Zeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Supply Chain Management und Operations an der Wirtschaftswissenschaftlichen Fakultät Ingolstadt der Katholischen Universität Eichstätt-Ingolstadt verfasst. Während dieser Zeit bekam ich von vielen Seiten Unterstützung, Vertrauen, motivierende Worte und den nötigen Ausgleich. Folgend möchte ich mich bei diesen besonderen Menschen bedanken.

Zuallererst gilt mein besonderer Dank meinem Doktorvater Prof. Dr. Heinrich Kuhn für die Betreuung meiner Dissertation und das Vertrauen, das er mir und seinen Mitarbeitern und Mitarbeiterinnen entgegenbringt. Er fand stets das Gleichgewicht zwischen individueller Forschungsfreiheit und konstruktiver Kritik. Außerdem danke ich Prof. Dr. Manuel Ostermeier, der sich bereit erklärt hat die Rolle des Zweitgutachters zu übernehmen. Vielen Dank auch an Prof. Dr. Michael Sternbeck und Prof. Dr. Pirmin Fontaine für den Beisitz bei der Disputation. Prof. Dr. Pirmin Fontaine gilt außerdem mein Dank für die produktive und angenehme Zusammenarbeit bei der Verfassung der gemeinsamen Beiträge. Diese Zusammenarbeit trug wesentlich dazu bei, dass wir zum sehr breit erforschten Feld der Tourenplanung innovative und relevante Beiträge leisten konnten.

Besonders die freundschaftliche Atmosphäre an unserem Lehrstuhl habe ich in den letzten Jahren zu schätzen gelernt. Vielen Dank für die Kaffeepausen voller Diskussionen wissenschaftlicher und nicht-wissenschaftlicher Natur! Ich danke außerdem Birgit Jürgens, dass sie uns wissenschaftlichen Mitarbeitern den Rücken freigehalten hat. Insbesondere möchte ich folgenden Kollegen und Kolleginnen danken:

- Dr. Markus Frank
- Marcel Lehmann
- Prof. Dr. Mareike Müller
- Tobias Potoczki
- Alexander Rave
- Dr. Fabian Schäfer

Ganz herzlich möchte ich meinen Freunden und meiner Schwester mit Familie danken, die mir die nötige Ablenkung geboten und meine Energiereserven wieder aufgefüllt haben. Besonderer Dank gilt meinen Eltern, ohne deren Förderung und Erziehung ich nicht in der Lage gewesen wäre zu promovieren.

Zuletzt danke ich meiner langjährigen Partnerin und gleichzeitig besten Freundin Lena. Meine Dankbarkeit für Dich lässt sich nicht in Worte fassen. Auf viele weitere gemeinsame Jahre!

Stefan Voigt

Zusammenfassung

Diese kumulative Dissertation beschäftigt sich konzeptionell als auch methodisch mit Fragestellungen, die im Rahmen der Zustellung auf der letzten Meile auftreten. Auf konzeptioneller Ebene werden innovative Ansätze für die letzte Meile vorgestellt und bewertet (Beitrag 1 und 4). Auf methodischer Ebene wird zum einen ein etabliertes Verfahren auf eine neue Problemstellung hin angepasst (Beitrag 4), zum anderen eine neuartige Metaheuristik vorgestellt und auf neue bzw. mehrere bekannte Probleme angewandt (Beitrag 1, 2 und 3).

1. Voigt, S., Frank, M., Fontaine, P., Kuhn, H., 2021. **The Vehicle Routing Problem with Availability Profiles**. Working Paper.
2. Voigt, S., Frank, M., Fontaine, P., Kuhn, H., 2022. **Hybrid Adaptive Large Neighborhood Search for Vehicle Routing Problems with Depot Location Decisions**. Computers & Operations Research, Volume 146.
3. Voigt, S., 2021. **Hybrid Adaptive Large Neighborhood Search for the Traveling Salesman Problem with Time Windows and Adjusted Costs**. In Winkenbach, M., Parks, S., and Noszek, J. (Eds.), Technical Proceedings of the 2021 Amazon Last Mile Routing Research Challenge, XXVI.1–XXVI.12.
4. Voigt, S., Kuhn, H., 2022. **Crowdsourced Logistics: The Pickup and Delivery Problem with Transshipments and Occasional Drivers**. Networks, 79: 403-426.

   Die veröffentlichten bzw. eingereichten Versionen dieser Beiträge können aus Gründen der Konsistenz (z. B. Rechtschreibung, Nomenklatur) geringfügig von den Versionen in dieser Arbeit abweichen. Dies hat keinen Einfluss auf den Inhalt der angenommenen Beiträge. Der Inhalt von Arbeitspapieren hingegen kann sich während des Begutachtungsprozesses noch ändern.

Der erste Beitrag entwickelt einen innovativen, datengetriebenen Ansatz zur Reduzierung fehlgeschlagener Zustellversuche, indem das Vehicle Routing Problem (VRP) um Verfügbarkeitsprofile erweitert wird. Außerdem wird eine neuartige Metaheuristik, die sog. Hybrid Adaptive Large Neighborhood Search (HALNS) vorgestellt und deren Performance analysiert. Die HALNS wird im zweiten Beitrag weiter entwickelt, sodass mehrere Varianten von VRPs mit Standortentscheidungen hinsichtlich der Depots gelöst werden können. Zu diesen Varianten zählen das 2-Echelon VRP, das Location Routing Problem sowie das Multi-Depot VRP. Der dritte Beitrag zeigt die Praxistauglichkeit der HALNS im Rahmen der Amazon Routing Research Challenge. Ziel der Challenge war es, gute Routen zu erzeugen, die außerdem der reellen Fahrweise der Fahrer möglichst nahe kommen. Der vierte Beitrag entwickelt einen innovativen Ansatz, basierend auf der Sharing Economy für die Zustellung auf der letzten Meile. Hierbei besteht die Möglichkeit sog. Gelegenheitskuriere, z.B. Pendler, für die Zustellung einzusetzen. Zur besseren Integration dieser Gelegenheitskuriere können zusätzliche Standorte, genauer Umschlagpunkte (Transshipment Points, TPs), genutzt werden. Die genannten Beiträge werden im Folgenden kurz zusammengefasst.

Beitrag 1 - The Vehicle Routing Problem with Availability Profiles

Die Zustellung von Paketen erfordert in vielen Fällen die Anwesenheit des Kunden zum Zeitpunkt der Lieferung. Falls sich der Kunde zu diesem Zeitpunkt nicht zuhause befindet, schlägt der Zustellversuch fehl, wobei zusätzliche Kosten und Aufwand für den Paketdienstleister anfallen. Kosten entstehen in Abhängigkeit der gewählten Strategie des Dienstleisters, so können beispielsweise weitere Zustellversuche unternommen werden oder das Paket zu einem Paketshop gebracht werden. Zusätzlich entstehen Unannehmlichkeiten für den Kunden, da er entweder länger auf seine Lieferung warten muss oder sie selbst bei einem Paketshop abholen muss. Paketdienstleister sind typischerweise mit einer hohen Rate fehlgeschlagener Zustellversuche konfrontiert, da sie nur sehr begrenzte Möglichkeiten haben mit dem Empfänger eine Zustellzeit zu vereinbaren. Selbst wenn diese Möglichkeit besteht, werden kaum Zustellzeiten vereinbart, da harte Zeitfenster die gefahrene Strecke beträchtlich erhöhen können.

Im Beitrag wird dem Problem fehlgeschlagener Zustellversuche bei der B2C-Paketzustellung durch die Verwendung kundenindividueller Verfügbarkeitsprofile (engl. Availability Profiles, APs) begegnet. Verfügbarkeitsprofile bestehen aus Zeitfenstern, denen jeweils eine Wahrscheinlichkeit zugeordnet ist, dass die Zustellung erfolgreich ist, wenn sie in dem jeweiligen Zeitfenster erfolgt. Die Wahrscheinlichkeiten, die den jeweiligen Zeitfenstern zugeordnet werden, lassen sich mit Hilfe verschiedenster Datenquellen schätzen (Wasserverbrauch, Stromverbrauch, Smart Devices, etc.).

Ziel des Beitrages ist es, den Nutzen von Verfügbarkeitsprofilen für die Planung der Liefertouren zu bewerten. Dazu wird das Vehicle Routing Problem mit Verfügbarkeitsprofilen (VRPAP) formuliert und ein mathematisches Modell aufgestellt, das den Tradeoff zwischen den Kosten für Transport und fehlgeschlagener Lieferungen einschließt. Die Entscheidungen umfassen dabei drei Fragestellungen: (1) welches Fahrzeug beliefert welchen Kunden? (Zuordnungsproblem) (2) in welcher Reihenfolge werden die Kunden angefahren? (Reihenfolgeproblem) und zusätzlich (3) zu welchem Zeitpunkt werden die Kunden angefahren? (Problem des Beginns der Servicezeit). Eine untere und obere Grenze dienen als Vergleichsmaß und erlauben Aussagen zum maximal möglichen Kosteneinsparungspotenzial des Modells. Das Modell enthält als Spezialfall das Rundreiseproblem, welches $\mathcal{NP}$-schwer ist, sodass sich größere Instanzen nicht in annehmbarer Rechenzeit lösen lassen. Zur Lösung wird daher eine neuartige Metaheuristik, die sog. HALNS vorgeschlagen. Sie vereint Elemente genetischer Algorithmen und der Adaptive Large Neighborhood Search.

Der Lösungsansatz wird zunächst auf verwandte Problemstellungen, namentlich das VRP mit harten Zeitfenstern (VRPTW) und die Variante mit mehreren Zeitfenstern (VRPMTW), sowie weichen Zeitfenstern (VRPSTW) angewandt, um dessen Leistungsfähigkeit zu untersuchen. Die HALNS erzielt dabei 20 neue beste Lösungen für das VRPMTW bzw. VRPSTW und vergleichbar gute Ergebnisse wie State-of-the-Art-Verfahren für das VRPTW. Anschließend wird der Ansatz auf synthetisch generierte Instanzen des VRPAPs angewandt und gezeigt, dass Kosteneinsparungen möglich sind, sowie die Rate fehlgeschlagener Zustellversuche in Abhängigkeit der Struktur der Verfügbarkeitsprofile und geografischen Verteilung der Kunden gesenkt werden kann. Eine Fallstudie mit empirischen Daten zeigt, dass selbst bei Verwendung wenig ausgereifter Prognoseverfahren signifikante Kosteneinsparungen von etwa 5% möglich sind, sowie fehlgeschlagene Zustellversuche um ca. 12% reduziert werden können.

Beitrag 2 - Hybrid Adaptive Large Neighborhood Search for Vehicle Routing Problems with Depot Location Decisions

Die im Beitrag *The Vehicle Routing Problem with Availability Profiles* vorgestellte HALNS wird nun auf VRPs mit Standortentscheidungen hinsichtlich der Depots erweitert. Solche Probleme treten auf der letzten Meile beispielsweise auf, falls Zustellungen von mehreren Depots aus erfüllt werden können oder eine gestufte Auslieferung vorgenommen wird, die über ein zentrales Depot abläuft, welches mehrere, üblicherweise kleinere Depots (sog. Satelliten) versorgt. Die HALNS wird um problemspezifische Operatoren erweitert, sodass sie für das Two-Echelon Vehicle Routing Problem (2E-VRP) geeignet ist. Im Beitrag wird gezeigt, dass das Location Routing Problem (LRP) und das Multi-Depot Vehicle Routing Problem (MDVRP) als Spezialfälle des 2E-VRP angesehen werden können und daher mit einem einzigen Verfahren gelöst werden können, statt wie in der Literatur üblich, mit spezialisierten Verfahren.

Die genannten Tourenplanungsprobleme weisen eine spezielle zweistufige Struktur auf, die es erschwert lokalen Minima zu entkommen. Die erste Stufe entspricht der Standortentscheidung, die zweite Stufe der Routing-Entscheidung. Dabei beeinflusst die zweite Stufe die erste Stufe so stark, dass eine Änderung der Standortentscheidung (erste Stufe) schwierig ist, ohne vorübergehend eine wesentliche Verschlechterung der Lösung zuzulassen, sobald über das Routing (zweite Stufe) entschieden wurde. Eine klassische ALNS arbeitet üblicherweise in einem Simulated Annealing Framework, sodass große Verschlechterungen nur zu Beginn möglich sind und eine breite Erkundung des Lösungsraumes im Verlauf des Verfahrens erschwert ist. Die vorgeschlagene HALNS nutzt die Struktur solcher zweistufigen Probleme aus. Die HALNS verwendet dabei eine Population von Lösungen, um einen größeren Bereich des Lösungsraums, d.h. verschiedene Standortkonfigurationen, zu erkunden. Die einzelnen Individuen werden durch eine effiziente ALNS generiert. Die Individuen dieser Population werden einer simplen, aber effektiven Kreuzungs- und Selektionsphase unterzogen.

Experimente mit diversen Instanzen aus der Literatur zeigen die Leistungsfähigkeit der HALNS. Die HALNS erzeugt ähnlich gute Lösungen wie Heuristiken, die speziell für das 2E-VRP, LRP oder MDVRP entwickelt wurden. Dies zeigt, dass die breitere Anwendbarkeit auf mehrere Probleme nicht notwendigerweise die Qualität der Lösung beeinträchtigt. Im Vergleich zu klassischen, d.h. nicht hybriden Implementierungen der ALNS, ist die HALNS diesen überlegen, was den Wert der Hybridisierung demonstriert.

Beitrag 3 - Hybrid Adaptive Large Neighborhood Search for the Traveling Salesman Problem with Time Windows and Adjusted Costs

In diesem Artikel wird die HALNS auf ihre Praxistauglichkeit hin untersucht. Zu diesem Zweck werden die Daten der Amazon/MIT Routing Research Challenge genutzt. Ziel der Challenge ist es, Routen zu erzeugen, die kostengünstig sind und außerdem den Routen erfahrener Fahrer möglichst nahe kommen. Das zugrunde liegende Problem wird als Traveling Salesman Problem mit Zeitfenstern (TSPTW) modelliert, welches ein Spezialfall des Vehicle Routing Problem mit Verfügbarkeitsprofilen (VRPAP) darstellt. Um das Verhalten der Fahrer nachzuahmen, werden

verschiedene Strategien implementiert, die sich aus den gegebenen Daten zu qualitativ hochwertigen Routen ableiten lassen. Mittels dieser Strategien werden zunächst die Kosten modifiziert, ehe die so angepassten Instanzen des TSPTW mit der HALNS gelöst werden. Experimente zeigen eine gute Anpassung an die tatsächlich gefahrenen Routen. Es wird ein durchschnittlicher Score von 0.0678 auf Testdaten mit allen 2718 qualitativ hochwertigen Routen erreicht. Insgesamt konnte damit der 14. Platz von 229 teilnehmenden Teams erreicht werden.

Beitrag 4 - Crowdsourced Logistics: The Pickup and Delivery Problem with Transshipments and Occasional Drivers

Der vierte Beitrag beschäftigt sich mit einem alternativen, innovativen Ansatz zur Reduzierung der Kosten im Umfeld von Paketzustellungen. Es nutzt die sog. Sharing Economy, deren Grundgedanke es ist, ungenutzte Ressourcen bzw. Kapazitäten zu teilen. Im Beitrag wird auf Pendler oder Reisende abgestellt, die sich dazu bereiterklären, für eine gewisse Entlohnung Zustellungen zu übernehmen, die sich in der Nähe ihrer ursprünglich geplanten Route befinden. Diese Gelegenheitskuriere bieten also eine Art Mitfahrgelegenheit für Pakete an. Um diese Gelegenheitskuriere (engl. occasional drivers, ODs) besser nutzen zu können, werden Umschlagpunkte (engl. transshipment points, TPs) eingeführt.

Ein Paketdienstleister, aus dessen Sicht das Problem betrachtet wird, möchte die Kosten minimieren, die mit der Auslieferung verbunden sind. Kosten setzen sich aus der von den regulären Fahrern zurückgelegten Strecke und die an die Gelegenheitsfahrer gezahlten Beträge zusammen. Der Paketdienstleister nutzt sowohl eine Fahrzeugflotte bestehend aus regulären Fahrern als auch zusätzlich eine Plattform, auf der sich ODs bereiterklären Zustellungen zu übernehmen, solange dies für sie nur mit einem gewissen Umweg im Vergleich zur ursprünglichen Route verbunden ist. An bereits existierenden TPs können ODs oder reguläre Fahrer die Ladung übergeben oder übernehmen. Als TPs können dabei bspw. Paketshops oder Packstationen dienen.

Das vorliegende Problem wird als gemischt-ganzzahliges Modell formuliert und als Pickup and Delivery Problem with Transshipments and Occasional Drivers (PDPTOD) bezeichnet. Das PDPTOD enthält analog zum VRPAP das Rundreiseproblem und ist damit $\mathcal{NP}$-schwer. Der Lösungsansatz basiert auf einer klassischen, d.h. nicht hybriden ALNS, da TPs die Komplexität derart erhöhen, sodass mehrere Durchläufe der ALNS, die für die Erzeugung der Population nötig wären, nicht in einem angemessenen Zeitrahmen durchgeführt werden können.

Der Fokus des Beitrags liegt in der Erzeugung von Erkenntnissen wie sich die Anzahl und die Standorte der TPs auf die Kostenvorteile auswirken, die durch die Integration von ODs in den Lieferprozess erzielt werden. Es zeigt sich, dass die Kosteneinsparungen in hohem Maße von der angenommenen Flexibilität und der gewählten Entlohnung für ODs abhängig sind.

Abstract

This cumulative dissertation addresses both conceptual and methodological issues that arise in the context of last-mile delivery. At the conceptual level, innovative approaches to last mile delivery are presented and evaluated (contributions 1 and 4). On the methodological level, on the one hand, an established procedure is adapted to a new problem (contribution 4), and on the other hand, a novel metaheuristic is presented and applied to new or several known problems (contributions 1, 2 and 3).

1. Voigt, S., Frank, M., Fontaine, P., Kuhn, H., 2021. **The Vehicle Routing Problem with Availability Profiles**. Working Paper.

2. Voigt, S., Frank, M., Fontaine, P., Kuhn, H., 2022. **Hybrid Adaptive Large Neighborhood Search for Vehicle Routing Problems with Depot Location Decisions**. Computers & Operations Research, Volume 146.

3. Voigt, S., 2021. **Hybrid Adaptive Large Neighborhood Search for the Traveling Salesman Problem with Time Windows and Adjusted Costs**. In Winkenbach, M., Parks, S., and Noszek, J. (Eds.), Technical Proceedings of the 2021 Amazon Last Mile Routing Research Challenge, XXVI.1–XXVI.12.

4. Voigt, S., Kuhn, H., 2022. **Crowdsourced Logistics: The Pickup and Delivery Problem with Transshipments and Occasional Drivers**. Networks, 79: 403-426.

   The published or submitted versions of these manuscripts may differ slightly from the versions in this thesis for consistency (e.g., spelling, nomenclature). This does not affect the content of accepted manuscripts. In contrast, the content of working papers is subject to changes during the review process.

The first paper develops an innovative data-driven approach to reduce failed delivery attempts by extending the well-known vehicle routing problem (VRP) with availability profiles. A novel metaheuristic, called hybrid adaptive large neighborhood search (HALNS), is presented for solving the VRP with availability profiles. The HALNS is further developed in the second paper so that several types of VRPs with location decisions regarding depots can be solved. These variants include the two-echelon VRP, the location routing problem, and the multi-depot VRP. The third paper shows the practicality of the HALNS in the context of the amazon routing research challenge. The goal of the challenge was to generate good routes similar to the real-world driving behaviour of drivers. The fourth contribution develops an innovative approach based on the sharing economy for last mile delivery. Here, the possibility exists to use so-called occasional drivers, e.g. commuters, for the delivery. To better integrate these occasional drivers, additional locations, more precisely transshipment points (TPs), can be used.

Contents

List of Figures

List of Tables

Contribution 1 - The Vehicle Routing Problem with Availability Profiles

1 The Vehicle Routing Problem with Availability Profiles

Stefan Voigt, Markus Frank, Pirmin Fontaine, Heinrich Kuhn

Abstract In business-to-consumer (B2C) parcel delivery the presence of the customer at the time of delivery is implicitly required in many cases. If the customer is not at home, the delivery fails – causing additional costs and effort for the parcel service provider as well as inconvenience for the customer. Parcel service providers typically report high failed-delivery rates, as they have limited possibilities to arrange a delivery time with the recipient. We address the failed-delivery problem in B2C parcel delivery by considering customer-individual availability profiles (APs) that consist of a set of time windows, each associated with a probability that the delivery is successful if conducted in the respective time window. To assess the benefit of APs for delivery tour planning, we formulate the vehicle routing problem with availability profiles (VRPAP) as a mixed integer program (MIP), including the tradeoff between transportation and failed-delivery costs. We provide analytical insights concerning the model's cost savings potential by determining lower and upper bounds. In order to solve larger instances we develop a novel hybrid adaptive large neighborhood search (HALNS). The HALNS is highly adaptable and also able to solve related time-constrained vehicle routing problems, i.e., vehicle routing problems with hard, multiple and soft time windows. We show its performance on these related benchmark instances and find a total of 20 new best-known solutions. We additionally conduct various experiments on self-generated VRPAP instances to generate managerial insights. In a case study using real-world data, despite little information on the APs, we were able to reduce failed deliveries by approximately 12% and overall costs by 5%.

Minor Revision: Transportation Science
URL (Working Paper): `https://doi.org/10.2139/ssrn.3793033`

1.1 Introduction & Motivation

In the course of digitization, e-commerce revenues have risen steadily worldwide and are expected to increase even further. They are predicted to account for 19.6% of global retail sales in the business-to-consumer (B2C) segment in 2021, with an annual growth rate of 25.7% (von Abrams 2021). This trend is the main reason why the number of worldwide parcel deliveries exceeded 131 billion in 2020 (Pitney Bowes 2021). The question regarding how customers receive all their online purchases physically, widely known as the *last mile* problem, becomes an even greater challenge as a result. The dominant delivery mode is home delivery due to its convenience for the customer (Hübner et al. 2016). However, the presence of the customer at the time of delivery is often implicitly required (e.g., for security reasons), resulting in the so-called attended home delivery problem (e.g., Agatz et al. 2011). If the customer is not at home, the delivery fails - causing additional costs and effort for the courier, express and parcel service provider (CEP) as well as inconvenience for the customer. Nowadays, several online stores allow customers a choice between alternative CEPs. This means they have a high incentive to avoid failed deliveries since otherwise the customer might change to a competing CEP. Reported failed-delivery rates in this sector (excluding CEPs that practice unsecured deliveries) can surmount 50% (Okholm et al. 2013). As a possible solution, CEPs could use customer-related data to increase the probability that a customer is at home at the time of the delivery. Pan et al. (2017) already showed that such data can be used to minimize failed deliveries in delivery tour planning, but use a sequential optimization approach, i.e., they first minimize failed-delivery costs and then solve the resulting vehicle routing problem with time windows (VRPTW). This approach neglects the potential increase in distance induced by following time windows (TWs), as well as the potentially increasing number of vehicles. Besides this work, the problem of unsuccessful home deliveries, let alone using customer data to reduce them, has received little attention in research so far. To address this routing problem in the context of B2C parcel delivery, we introduce the vehicle routing problem with availability profiles (VRPAP). We consider customer-individual availability profiles (APs) that consist of a set of discrete, non-overlapping TWs. Each TW is associated with a certain probability of the delivery being successful if conducted in the respective TW. We minimize the sum of routing costs and customer-individual expected failed-delivery costs. The VRPAP explicitly addresses the tradeoff between transportation and failed-delivery costs in this way. The VRPAP is a generalization of the VRPTW and therefore $\mathcal{NP}$-hard. We develop the hybrid adaptive large neighborhood search (HALNS) for solving this problem efficiently, and compare it to related benchmark algorithms from the literature. Further, we conduct several analyses regarding the impact of failed-delivery costs on vehicle routing with simulated and empirical data.

The contributions of our work are as follows: (1) We show how customer-related data can be used to decrease the share of failed deliveries while considering the cost of routing. We explicitly model the tradeoff between costs for routing and failed deliveries, and define a novel problem class in the setting of parcel delivery, i.e., the VRPAP. (2) We model the problem as a mixed integer program (MIP) and (3) analytically provide insights into its theoretical cost savings potential. (4) We develop a highly adaptable metaheuristic solution framework that is suitable not only for

solving practical-size instances of the VRPAP, but also for solving the VRPTW, the VRP with multiple time windows (VRPMTW), and variants of the VRP with soft time windows (VRPSTW). It produces 12 new best-known solutions for VRPMTW and 8 for VRPSTW benchmark instances. (5) Experiments show the cost savings potential when CEPs consider APs of their customers during the delivery process.

The remainder of this paper is structured as follows: Section 1.2 details the problem setting considered. Section 1.3 gives an overview of research areas concerning relevant application- and model-specific literature. Section 1.4 then presents the VRPAP model formulation. Section 1.5 outlines the HALNS, and Section 1.6 presents the numerical analyses. Section 1.7 finally summarizes our work and indicates further research directions.

1.2 Problem Description

In the following we introduce the VRPAP. Section 1.2.1 describes the delivery process within this context. Section 1.2.2 then details the failed-delivery issue and describes how APs can contribute to increasing the rate of successful deliveries. Afterwards, Section 1.2.3 derives the problem-specific and decision-relevant costs. Lastly, Section 1.2.4 summarizes the operational planning problem considered.

1.2.1 Delivery Process

In e-commerce B2C sales, customers can choose between home delivery and a large set of alternative collection points such as parcel shops, parcel stations, or parcel lockers. Nevertheless, the dominant delivery mode remains home delivery. In this case, the CEP builds round trips from a single depot to a number of customers' home addresses. The tours are carried out by drivers in trucks that bring the deliveries right up to the customer's front door. At each customer, the driver needs to get out of the truck, pick out the customer's parcel, go to the customer's door, wait for the customer to open, hand over the parcel and return. The service time required to perform these tasks depends on the local conditions and varies significantly. In the majority of cases, customers have to receive their delivery in person as the customer's signature is required as a confirmation of receipt (e.g., Hermes 2020). Even for CEPs that in general practice unsecured or unattended home deliveries, the presence of the customer is mandatory for certain deliveries, e.g., for sensitive or highly valuable goods. Unattended home delivery requires a secure place to deposit the parcel such as an accessible storage location. Although this place may theoretically be available, the CEP may not have the customer's consent to deposit their parcels there. So in most cases the delivery fails if the customer is not at home at the time of delivery.

1.2.2 Failed Deliveries & Availability Profiles

Generally, the *failed-delivery rate* is not a standardized performance metric in the CEP industry. CEPs may include all possible reasons for the delivery attempt to fail in this measure, apart from the customer being unavailable, e.g., the delivery address being wrong, not locatable or inaccessible,

or the customer refusing to accept the delivery. At the other extreme, some CEPs count a delivery as failed only when the parcel ends up in the depot again after the last failed delivery attempt. At this point, several additional processes may already have been carried out, which depend on the CEP's general terms and conditions, and the lived practice. Amongst others, the driver would search for a willing neighbor, bring the delivery to a parcel shop in the neighborhood or return the package to the depot to start a new attempt on another day. That way, up to three delivery attempts may take place until the parcel is finally returned to the sender (e.g., Hermes 2020). In all of these cases, it is reasonable to count a delivery attempt as failed since this immediately requires an additional effort. Edwards et al. (2009) report on failed first-time delivery attempts in the magnitude of 10% to 50% where specific delivery times were not prearranged. More recent numbers of Okholm et al. (2013) show an average of 12% and a maximum of more than 50% failed first-time delivery attempts for CEP deliveries in the European Union. It is possible to reduce the probability of failed deliveries by introducing TWs of which customers may choose an appropriate one. However, strict TWs cause significantly higher transportation costs due to less efficient tours (Punakivi and Saranen 2001). This and the CEP's position between sender and receiving customer, often without any contact data of the customer, are the main reasons why TWs are typically not offered in parcel delivery (Wong 2008). Instead of fixed, pre-arranged TWs, customer-individual APs that indicate the probability of a customer being at home can be considered when building delivery schedules (Florio et al. 2018). APs can consist of discrete, non-overlapping TWs with associated probabilities that the customer is available for receiving a delivery within the given TW. APs can be generated in various ways, e.g., by using historical delivery data or based on socio-economic data of the delivery area (Cardenas et al. 2016, van Duin et al. 2016), by electricity consumption data on a household level (Pan et al. 2017) or GPS location data (Praet and Martens 2019). The TW granularity, i.e., the number of TWs within the daily delivery period, should be chosen depending on the available data.

1.2.3 Decision-relevant Costs and Constraints

Costs Considering the distribution process described above, we identify two main cost factors for attended home delivery with failed deliveries: transportation costs for the movement between locations, and expected failed-delivery costs for delivery attempts that may be unsuccessful. Transportation costs arise for each delivery tour to be performed and include the typical costs for traveling between the locations as well as customer-individual service costs to hand over the delivery at the customer's location. Failed-delivery costs depend heavily on the CEP's policy and practice in dealing with failed deliveries. Possible cost-relevant scenarios include, but are not restricted to:

- *Delivery to a neighbor*: Additional service time for the driver.
- *Delivery to a parcel shop*: Time to reload the parcel into the vehicle, transportation costs to reach the parcel shop, fixed fee per parcel when the parcel shop is a partner.
- *Return to the depot & start new delivery attempt next day(s)*: Time to reload the parcel into the vehicle, handling costs at the depot, transportation costs to reach the customer again on

the next day(s), additional service time at the second attempt.

Additionally, a second delivery attempt on the same day either within the original tour or performed by another driver who takes over the delivery of failed parcels would be conceivable, but is not a common practice. In all cases, the driver has to leave a notification in the customer's mailbox or trigger an electronic notification for the customer (Hermes 2020). A higher setting of failed-delivery costs may also reflect the increasing importance of CEP's customer satisfaction, as in e-commerce, customers increasingly have the option of selecting the CEP of their choice. As a consequence, failed-delivery costs may include opportunity costs incurred when a customer is lost. The actual costs may further differ between customers due to the availability or unavailability of a parcel shop in the neighborhood of the customer or simply because of greater or less service time required due to local circumstances.

Constraints Various constraints have to be considered within delivery planning of packages to customers' homes. The number of customers on a single delivery tour is restricted by the capacity of the vehicle as well as the length of the delivery period. In the classical VRPTW, waiting times between customers are often necessary to reach all customers within their individual TWs. The problem setting considered here, however, induces waiting times not only to obtain feasible solutions, but also to reduce total costs. The waiting times for all drivers are implicitly limited by the maximum number of vehicles available and the length of the delivery period, as all customers have to be visited within this period by one of the vehicles.

1.2.4 Operational Planning Problem

CEPs engaged in an attended home delivery B2C setting seek to minimize expected costs for the last mile delivery. We consider the single-day operational problem where costs arise from traveling to the customer, serving the customer, and additional costs if the delivery fails. These costs are influenced by decisions regarding (1) the clustering of customers to tours, (2) the delivery sequence on each tour and (3) the selection of delivery TWs. The expected failed-delivery costs depend on the customer's availability within the TW chosen by (3). Service has to start within this chosen TW. In contrast to the VRPTW, the CEP does not have to arrange the tour such that the driver arrives at a specific TW. Instead, arrival is possible throughout the planning horizon, but the customer's availability depends on the TW. Allowing deliveries within the entire delivery period and quantifying the customer's availability via APs enables the CEP to balance between transportation and expected failed-delivery costs.

1.3 Related Literature

The decision problem considered belongs to the class of time-constrained VRPs, in particular VRPs with single or multiple time windows. Section 1.3.1 therefore reviews corresponding time-constrained VRPs first, and Section 1.3.2 the literature that explicitly considers failed deliveries in the context of attended home delivery second. Section 1.3.3 summarizes our findings and specifies the research gap.

1.3.1 Time-constrained Vehicle Routing Literature

The literature on time-constrained VRPs contains models that can be categorized by their assumptions related to the availability of customers within the delivery horizon. With reference to Vidal et al. (2015), we distinguish four problem classes with increasing complexity: the VRP with time windows (VRPTW), the VRP with multiple time windows (VRPMTW), the VRP with soft time windows (VRPSTW) and the VRP with general time windows (VRPGTW). Figure 1.1 visualizes the respective model assumptions of TW constellations and customer availability across the delivery horizon.

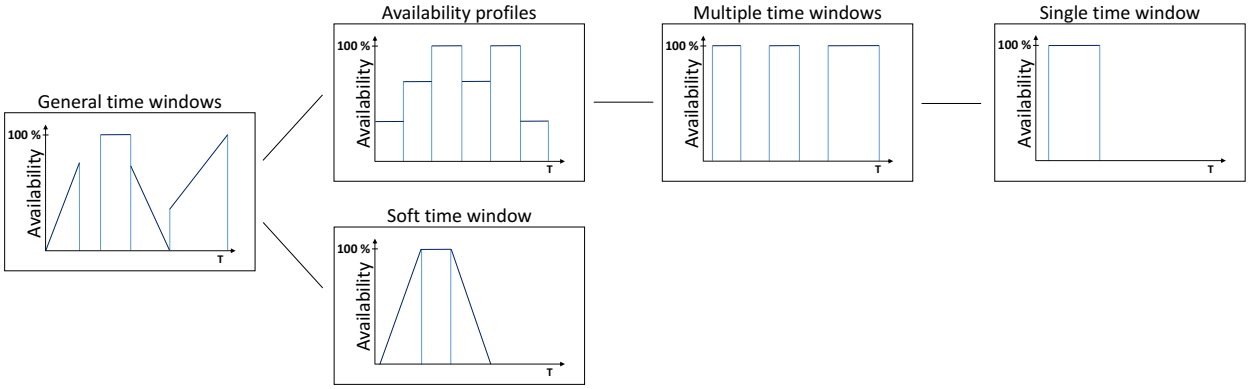


Figure 1.1: Alternative constellations of customer availability across the delivery horizon

The upper branch of Figure 1.1 shows the relation of VRPGTW, VRPAP, VRPMTW, and VRPTW. The VRPAP defines a special case of the VRPGTW with multiple TWs covering the complete delivery horizon and constant failed-delivery costs in every TW. The VRPAP equals the VRPMTW if costs for a failed delivery are infinitely high and TWs are either associated with 0% or 100% customer availability. The VRPAP becomes the VRPTW if there is only one TW with 100% customer availability and all other TWs are associated with 0% customer availability, i.e., there are infinitely high failed-delivery costs for all but one TW. The lower branch of Figure 1.1 shows that the VRPSTW defines another special case of the VRPGTW with a single TW and linearly increasing penalties.

1.3.1.1 VRPTW

Vidal et al. (2013a) present an in-depth review on the diverse set of problem cases discussed in the literature. We however focus on VRPTW publications that present state-of-the-art algorithms and results, which can be used as benchmark for our solution framework proposed. Pisinger and Ropke (2007) present a general heuristic based on the adaptive large neighborhood search (ALNS) for a large class of vehicle routing problems, including the VRPTW. Nagata et al. (2010) present a rather specialized memetic algorithm for the VRPTW. The authors use the Edge Assembly Crossover from Nagata (1997) together with a new penalty function which allows for time warps, i.e., if the vehicle arrives too late at the customer it may use a penalized time warp. Vidal et al. (2013b) build upon the time warp concept and introduce the hybrid genetic search with adaptive diversity control for

VRPs with time features. In contrast to Nagata et al. (2010), their approach works on a large class of time-constrained problems.

1.3.1.2 VRPMTW

De Jong et al. (1996) are the first to formalize the VRPMTW as an MIP. Their objective minimizes total costs, consisting of transportation costs and cost of waiting time. Favaretto et al. (2007) formulate the VRPMTW with periodic constraints. An ant colony optimization algorithm is implemented to solve benchmark instances generated based on VRP instances from the literature, but without considering periodic visits. For the same setting, Belhaiza et al. (2014) develop a hybridized variable neighborhood tabu search heuristic with adaptive memory and solve instances from Favaretto et al. (2007) and newly generated instances based on the VRPTW instances of Solomon (1987), minimizing either travel time or route duration. The latter instances serve as a benchmark for future work. Belhaiza et al. (2017) introduce a hybrid genetic variable neighborhood search for both objectives. Larsen and Pacino (2019) also consider the VRPMTW where the total duration is minimized, as well as a variant where the total travel time is minimized. They implement an ALNS using operators adapted from Ropke and Pisinger (2006) and focus on fast solution evaluations. Belhaiza et al. (2019) present a framework with three different multi-start strategies where they treat solutions as individuals of a genetic population and enhance their VNS with typical destroy-and-repair procedures known from LNS. Hoogeboom et al. (2020) solve the VRPMTW with an adaptive VNS including an exact polynomial time algorithm to determine the optimal service start times of customers in a given route sequence. We refer to this subproblem hereinafter as the optimal start time problem (OSTP). In a similar manner, Schaap et al. (2019) use a LNS with a dynamic programming approach to optimally select a TW for each customer on a route.

1.3.1.3 VRPSTW

Fu et al. (2008) define six different types of soft TWs. Most relevant for and closest to our application are types 1 and 2, where either only late (type 1) or early and late customer visits (type 2) are possible but incur a penalty that is linear to the TW violation. The type 1 variant was first introduced by Taillard et al. (1997) and solved by a tabu search heuristic. The type 2 variant was first considered by Koskosidis et al. (1992) and addressed by solving the clustering problem and the resulting TSPs with soft TWs separately. Like Taillard et al. (1997) they minimize total distance and total penalties simultaneously. Later works on the VRPSTW however use lexicographic objectives. Figliozzi (2010), for instance, first minimizes the number of vehicles, then the number of TWs violated and after that the total distance. The best solution approaches on the commonly used (adapted) Solomon benchmarks for types 1 and 2 that are available currently are Mouthuy et al. (2015), Vidal et al. (2014b) and Kritzinger et al. (2017). Mouthuy et al. (2015) use a multistage very large-scale neighborhood search where neighboring solutions are reached not by a single local move but by a sequence of moves in order to achieve better local optima. They also develop a specialized heuristic for the OSTP arising in the VRPSTW. Vidal et al. (2014b)

propose a hybrid genetic search with an efficient local search. Kritzinger et al. (2017) adapt the variable neighborhood search method with numerous shaking operators. The latter two approaches are developed to solve a large number of VRP variants including the VRPSTW.

1.3.1.4 VRPGTW

The VRPGTW further extends the VRPMTW and the VRPSTW. It allows customer visits at all times, but at a time-dependent cost defined by a penalty function. In contrast to the typically linear penalties in the case of soft TWs, the penalty functions in the VRPGTW can take any form and also be of a non-convex and discontinuous nature. Nevertheless, piece-wise linearity is assumed in all approaches dealing with the VRPGTW described in literature. Ibaraki et al. (2005) are the first authors who explicitly consider the VRPGTW. They minimize distance costs and penalty costs for TWs used and exceeding vehicle capacity. The problem of determining service start times with minimal penalties for a given route, the OSTP, is identified as a critical subproblem in the VRPGTW since customers can be visited at any time and waiting times between the customers are allowed. They are the first to propose a dynamic programming approach to compute the OSTP. Ibaraki et al. (2008) improve the dynamic programming algorithm of Ibaraki et al. (2005) for the OSTP assuming that the piece-wise linear penalty functions are also convex. Hashimoto et al. (2006) extend the VRPGTW by treating the travel times as time-dependent penalty functions, too, and adapt the dynamic programming algorithm from Ibaraki et al. (2005) to their problem setting.

1.3.2 Failed Delivery Literature

The literature has so far scarcely treated the possibility of a failed delivery in the course of tour planning of attended home deliveries. Pan et al. (2017) use electricity consumption data of customers to reduce failed deliveries in the context of e-groceries. They follow a sequential optimization approach where they first minimize failed-delivery costs by delivering to all customers within their TW with the highest availability and then solving the resulting VRPTW. They show that the approach can decrease the first-time failed-delivery rate by up to 26% in an experiment with 15 customers. However, their approach neglects the potential increase in distance induced by predefined TWs, as well as the potential increasing number of vehicles needed to serve all customers. This so-called ping-pong effect can lead to a significant increase in transportation costs (see, for example, Punakivi et al. 2001). Florio et al. (2018) formulate a delivery problem that also uses APs. In contrast to the VRPAP, they use continuous APs. The model defined minimizes failed deliveries where route duration is restricted. They solve the delivery problem with a branch-and-price algorithm for up to 100 customers. The algorithm leads to optimal solutions if revisits are not allowed and provides a heuristic solution for instances where revisits are possible. They find that using APs for designing routes may result in an average reduction of 34% of failed deliveries. The modeling approach of Florio et al. (2018), however, neglects to include the simultaneous optimization of transportation costs and expected failed-delivery costs. Their modeling approach requires artificial constraints on the additional distance tolerable that have to be defined in advance. Özarık

et al. (2021) present a VRP with discrete customer availability profiles and time-dependent penalty costs, where transportation and expected failed-delivery costs are simultaneously minimized. In contrast to the VRPAP, they neglect vehicle capacities and set the failed-delivery costs to the back and forth distance between the depot and the respective customer instead of a more flexible policy-dependent parameter. The authors propose an ALNS to solve the model formulated, and include a local search procedure to optimize the service start times for given customer sequences, which is similar to the approach of Ibaraki et al. (2005). In their experiments they find that incorporating APs in tour planning reduces total costs by up to 40%.

1.3.3 Summary and Research Gap

To sum up, the problem of failed deliveries in attended home delivery has only rarely been studied in the literature. Existing approaches neglect the effect of increasing distances and the increasing number of vehicles required when introducing TWs or miss features of the operational planning problem, e.g., policy-dependent failed delivery costs and vehicle capacities. An exhaustive modeling approach is needed to conduct research towards numerous open research questions, e.g., the importance of the number of available vehicles, the influence of customer-individual failed-delivery costs and the effects of differently structured APs. Also, the benefit of using the VRPAP instead of existing modeling approaches needs to be clarified. We use our VRPAP modeling approach presented below to address these questions. The VRPAP extends the VRPMTW literature by introducing a valuation of all defined TWs. VRPMTW models assume that the selected TW does not directly affect the objective function. In the VRPAP TWs are valued by availability profiles. The VRPGTW has received only scant attention in the literature so far because it lacks convincing applications. The VRPAP can be seen as a special case of the VRPGTW with constant costs for TWs and has a convincing application in attended home parcel delivery. Our proposed HALNS framework is developed to specifically solve the VRPAP, but is also suitable for solving related time-constrained problem settings.

1.4 Decision Model for Vehicle Routing with Availability Profiles

The VRPAP is defined on a directed graph $G(N, A)$ with node set N and arc set A. Node set N consists of the depot 0 and customers $j \in C$. The arc set is defined as $A = \{(i, j) : i \neq j,\ i, j \in N\}$. c_{ij}^{trans} denotes the associated transportation costs with each arc. The depot serves as start and end point of all delivery tours. We assume a given fleet of K delivery trucks that states the maximum number of tours. All trucks have the same restricted vehicle capacity Q, while the customer's $j \in C$ demands d_j may be different. Split deliveries are not allowed. The time of a tour consists of the traveling time t_{ij} for driving from one node to the next, of the customer-individual service time S_j, and possibly the waiting time. The maximum route duration is restricted by the delivery period's length D. We divide the delivery period for each customer into $|W|$ discrete, non-overlapping TWs with lower and upper limits e_{jw} and l_{jw} with associated probabilities p_{jw} that customer j is available for receiving a delivery within the given TW w. The customer's individual failed-delivery cost parameter c_j^{failed} denotes the costs that occur if a delivery attempt fails. We are able to represent different policies in dealing with failing deliveries with the same model by setting c_j^{failed} as policy-dependent. The binary decision variable x_{ij} indicates whether arc (i, j) is used. The binary decision variable y_{jw} becomes 1 if TW w is chosen for delivering to customer j. Table 1.1 summarizes the notation.

Table 1.1: Notation

Sets	
C	Set of customers, $C = \{1, ..., \|C\|\}$
N	Set of nodes, $N = \{0\} \cup C = \{0, ..., \|C\|\}$
W	Set of TWs, $W = \{1, ..., \|W\|\}$
Parameters	
c_{ij}^{trans}	Transportation cost from i to j, $i, j \in N$
t_{ij}	Traveling time from i to j, $i, j \in N$
d_j	Demand of customer j, $j \in C$
S_j	Service duration at customer j, $j \in C$
p_{jw}	Availability probability of customer j during TW w, $j \in C,\ w \in W$
c_j^{failed}	Cost of failed delivery attempt for customer j, $j \in C$
e_{jw}	Earliest start of service of TW w for customer j, $j \in C,\ w \in W$
l_{jw}	Latest start of service of TW w for customer j, $j \in C,\ w \in W$
K	Maximum number of vehicles
Q	Maximum capacity of a vehicle
D	Length of delivery period
Decision Variables	
x_{ij}	Binary variable indicating whether arc (i, j) is used, $(i, j) \in A$
y_{jw}	Binary variable indicating whether TW w is chosen for customer j, $j \in C,\ w \in W$
s_i	Start time of service at node i, $i \in N$
q_i	Accumulated load until node i, $i \in N$

Model VRPAP

$$\text{Minimize } C^{\text{total}}_{\text{VRPAP}} = \sum_{i \in N} \sum_{j \in N} c^{\text{trans}}_{ij} x_{ij} + \sum_{j \in C} \sum_{w \in W} c^{\text{failed}}_{j} (1 - p_{jw}) y_{jw} \tag{1.1}$$

s.t.

$$\sum_{j \in C} x_{0j} \leq K \tag{1.2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in C \tag{1.3}$$

$$\sum_{j \in N} x_{ij} = \sum_{j \in N} x_{ji} \qquad \forall i \in N \tag{1.4}$$

$$q_j - q_i \geq d_j - Q(1 - x_{ij}) \qquad \forall i, j \in C,\ i \neq j \tag{1.5}$$

$$d_j \leq q_j \leq Q \qquad \forall j \in C \tag{1.6}$$

$$s_j + S_j + t_{j0} \leq D \qquad \forall j \in C \tag{1.7}$$

$$s_j - s_i \geq (t_{ij} + S_i) x_{ij} - D(1 - x_{ij}) \qquad \forall i, j \in C,\ i \neq j \tag{1.8}$$

$$\sum_{w \in W} e_{jw} y_{jw} \leq s_j \leq \sum_{w \in W} l_{jw} y_{jw} \qquad \forall j \in C \tag{1.9}$$

$$\sum_{w \in W} y_{jw} = 1 \qquad \forall j \in C \tag{1.10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in N \tag{1.11}$$

$$y_{jw} \in \{0, 1\} \qquad \forall j \in C,\ w \in W \tag{1.12}$$

$$s_i, q_i \in \mathbb{R}^+_0 \qquad \forall i \in N \tag{1.13}$$

The objective function (1.1) minimizes the total expected costs, which consist of total transportation costs and expected costs of failed delivery attempts. Costs for a delivery that failed arise with the absence probability multiplied by the cost of failed deliveries if the respective TW is chosen. Constraint (1.2) restricts the number of vehicles used. Constraints (1.3) ensure that each customer is visited exactly once. Constraints (1.4) conserve flow. Constraints (1.5) and (1.6) ensure feasibility of loads. Constraints (1.7) ensure that vehicles arrive back at the depot again before the end of the delivery period. Constraints (1.8) guarantee that the service of an immediate successor starts only after traveling and service time starting from the predecessor. Constraints (1.9) ensure that the service must start within the chosen TW. Constraints (1.10) ensure that exactly one TW is chosen for each customer. Constraints (1.11) - (1.13) define the domains of the variables.

The VRPAP generalizes the VRPTW (see Section 1.3.1). This means it is also $\mathcal{NP}$-hard. Only small instances can be solved by exact approaches. Heuristics are required to solve larger instances. In the following we derive a lower and upper bound to benchmark solutions achieved by our heuristic approach.

Theorem 1. *Let $C^{\text{trans}}_{\text{VRP}}$ denote the resulting costs, when solving the VRPAP ignoring failed-delivery costs, i.e., $c^{\text{failed}}_j = 0,\ \forall j \in C$. These costs equal the transportation costs when solving the corre-*

sponding VRP. Then, a lower bound for the VRPAP is denoted as follows.

$$C^{\text{lower}}_{\text{VRPAP}} = C^{\text{trans}}_{\text{VRP}} + \sum_{j \in C} c^{\text{failed}}_j (1 - \max(p_{j1}, ..., p_{j|W|})) \tag{1.14}$$

Proof.

$$\begin{aligned} C^{\text{total}}_{\text{VRPAP}} &= \sum_{i \in N} \sum_{j \in N} c^{\text{trans}}_{ij} x_{ij} + \sum_{j \in C} \sum_{w \in W} c^{\text{failed}}_j (1 - p_{jw}) y_{jw} \\ &\overset{(a)}{\geq} \sum_{i \in N} \sum_{j \in N} c^{\text{trans}}_{ij} x_{ij} + \sum_{j \in C} c^{\text{failed}}_j (1 - \max(p_{j1}, ..., p_{j|W|})) \\ &\overset{(b)}{\geq} C^{\text{trans}}_{\text{VRP}} + \sum_{j \in C} c^{\text{failed}}_j (1 - \max(p_{j1}, ..., p_{j|W|})) = C^{\text{lower}}_{\text{VRPAP}} \end{aligned} \tag{1.15}$$

Inequality (a) states that the TWs with the highest availability probability result in lower or equal costs than the selected TWs in any VRPAP solution. Inequality (b) results from the definition of $C^{\text{trans}}_{\text{VRP}}$ as the the lowest transportation costs, determined by solving the corresponding VRP. □

An upper bound $C^{\text{upper}}_{\text{VRPAP}}$ for the VRPAP can be derived as follows. We can quantify the sum $C^{\text{total}}_{\text{VRP}}$ consisting of the minimum transportation costs $C^{\text{trans}}_{\text{VRP}}$ and the expected failed-delivery costs for each customer using the associated TWs resulting from solving the VRP. Furthermore, we calculate total costs $C^{\text{total}}_{\text{VRPTW}}$ with the minimum possible failed-delivery costs by solving a VRPTW assuming that the respective TWs with maximum availability apply to each customer. Obviously, $\min(C^{\text{total}}_{\text{VRP}}, C^{\text{total}}_{\text{VRPTW}})$ represents an upper bound $C^{\text{upper}}_{\text{VRPAP}}$ for the VRPAP.

1.5 Hybrid Adaptive Large Neighborhood Search

This section presents a novel hybrid adaptive large neighborhood search (HALNS) for solving the VRPAP. Our HALNS combines elements of an adaptive large neighborhood search (ALNS) as introduced by Ropke and Pisinger (2006) and Pisinger and Ropke (2007), the parallelized ALNS (Mühlbauer and Fontaine 2021) and elements of genetic algorithms, i.e., a population management of individuals (also called solutions), and a crossover phase. The general idea is to maintain a population of individuals that are improved via several ALNSs. Compared to classical ALNSs, these ALNSs use the information available within the population during its optimization procedure. The HALNS relies on this feature as the inner ALNS is optimized for finding reasonably good solutions within very short runtimes as the ALNS is run multiple times. In doing so, we pursue two goals. First, the population-based approach expands the search space, and second, the neighborhood-centred ALNS intensifies the search around promising solutions. Both approaches in combination increase the chance to escape local optima.

Algorithm 1 describes the general structure of the HALNS proposed. We consider an initial population P of size n_P. The initial population of individuals is generated by executing an ALNS n_P times, resulting in potentially several different reasonably good solutions (lines 1-3). The ALNS is then used to crossover individuals (line 6-9), while the number of generations without

Algorithm 1: Hybrid Adaptive Large Neighborhood Search

```
1  while |P| < n_P do                                   // Generation of Initial Population
2  |   s ← ALNS()
3  |   P ← P ∪ {s}
4  while gens without Improvement < gen^stop ∧ gens < gen^max do      // GA Generations
5  |   ŝ ← DetermineBestSolution(P)
6  |   while i < n_P do                                  // Crossover Phase
7  |   |   s ← P[i]
8  |   |   s ← ALNS(s, ŝ)
9  |   |   P ← P ∪ {s}
10 |   P ← SelectSurvivors(P)                           // Select surviving individuals
```

improvement is lower than gen^{stop} or the number of generations has not reached its limit, $gen^{\max}$ (line 4). The individuals of a population that are used within the next generation (survivors) are chosen according to their solution quality and their contribution to the diversity of the population (line 10). The proposed ALNS is described in Section 1.5.2. Section 1.5.3 then details the crossover phase. Section 1.5.4 describes the selection logic of survivors.

1.5.1 Solution Representation and Penalized Costs

A solution is represented by a set of routes R, where every customer is covered exactly once by exactly one route. Let r be a route in R, i.e., a sequence of $|r|-1$ customer visits, which starts at the depot ($r_0 = 0$), ends at the depot ($r_{|r|} = 0$) and covers customers in between ($r_{(1,\dots,|r|-1)} \neq 0$). Following the idea of Vidal et al. (2013b), we allow non-feasible solutions in the search space by penalizing the degree of infeasibility with an additional weight ω, i.e., a penalty cost factor, in the cost function. We then combine the costs for transportation C_r^{trans} and failed delivery C_r^{failed} with possible penalty costs for an overloaded vehicle P_r^{Load} and TW violations P_r^{TW}. The adapted entire costs of a route r, $f(r)$, are hereafter denoted as penalized costs.

$$f(r) = C_r^{\text{trans}} + C_r^{\text{failed}} + \omega(P_r^{\text{Load}} + P_r^{\text{TW}}) \tag{1.16}$$

Let $p_j^{\text{available}}$ be the availability probability resulting from choosing a TW for customer j in r. Furthermore, let P_{ir} be the lateness resulting from being too late for the designated TW w, or being too late back at the depot for node i in r, which is defined as follows.

$$P_{ir} = \begin{cases} \max\{s_i - l_{iw}, 0\} & i \in r, i \neq 0 \\ \max\{s_{r_{|r|-1}} + S_{r_{|r|-1}} + t_{r_{|r|-1},i} - D, 0\} & i = 0 \end{cases} \tag{1.17}$$

The individual terms of route r are then calculated as follows:

- Transportation costs: $C_r^{\text{trans}} = \sum_{i=0}^{|r|-1} c_{r_i, r_{(i+1)}}^{trans}$
- Failed-delivery costs: $C_r^{\text{failed}} = \sum_{j=1}^{|r|-1} c_j^{\text{failed}}(1 - p_j^{\text{available}})$
- Excess truck load: $P_r^{\text{Load}} = \max\{0, \sum_{j=1}^{|r|-1} d_j - Q\}$

- TW lateness: $P_{ir}^{\text{TW}} = \sum_{i=0}^{|r|} P_{ir}$

The penalized cost $f(s)$ of a solution s is then the sum of the penalized costs of all routes in R, i.e., $f(s) = \sum_{r \in R} f(r)$.

1.5.2 ALNS Algorithm

Algorithm 2: ALNS algorithm in simulated annealing framework

```
   Input  : Starting solution s, global best solution ŝ, temperature γ
   Output: best solution s*
1  s* ← s
2  while Iterations without improvement < it^limit do
3  |  ChooseOperators()
4  |  C_R^C ← getRemovalCandidates(s, ŝ)
5  |  (s^new, C_R) ← Remove(s, C_R^C)
6  |  C_R ← sort(C_R)
7  |  s^new ← Insert(s^new, C_R)
8  |  if f(s^new) < f(s*) then
9  |  |  (s, s*) ← s^new
10 |  |  if f(s*) < f(ŝ) then
11 |  |  |  ŝ ← s*
12 |  else if accept(f(s^new), f(s*), Temp) then
13 |  |  s ← s^new
14 |  γ ← β · γ
15 |  UpdateWeights()
```

The ALNS takes the current solution s, the global best solution $\hat{s}$ and the starting temperature γ as input. Initially, the local best solution s^* is set, and the ALNS generates new solutions by iteratively removing and then inserting customers from and into the current solution. In each iteration, a removal operator (see Section 1.5.2.3) and an insertion operator (see Section 1.5.2.5) are randomly chosen (line 3). Hereby, the selection probability depends on the historic performance of the respective operator. Different from classical ALNS implementations, a set of customers that are candidates for removal C_R^C is generated by comparing the current solution s and the global best solution $\hat{s}$ (line 4, see Section 1.5.2.2). The removal operator chosen then removes some or all customers of set C_R^C from the solution (line 5). Additionally, data collected during the search determines the order in which the removed customers C_R are inserted (line 6). The determination of the insertion order (see Section 1.5.2.4) is an extension to existing ALNS implementations that either randomly insert customers or determine the order during the insertion process (e.g., regret-insertion Ropke and Pisinger 2006). Afterwards, the insertion operator chosen inserts the set of previously removed and sorted customers into the solution (line 7). We use a simulated annealing acceptance criterion (see Section 1.5.2.6) with cooling rate β to avoid getting stuck in a local optimum (lines 8-14), like other ALNS implementations (e.g., Ropke and Pisinger 2006, Larsen and Pacino 2019). Finally, the weights are updated (see Section 1.5.2.7) according to the performance of the operators (line 15). The ALNS ends after a predefined number of iterations without any improvements, it^{limit}.

Note that the HALNS generates the initial solution in the first generation by iteratively applying the *Best Route Insertion Operator* (see Section 1.5.2.5) until every customer is served. In the

following generations, the initial solution of an ALNS phase is chosen from the population of solutions.

1.5.2.1 Information Collection

The insertion operators collect three types of information when inserting customers. This information is used during the subsequent removal and insertion operations.

- The historic penalized cost C_j^{hist} is increased on every insertion of customer j by $\Delta_j = f(s_1) - f(s_0)$, where s_1 represents the solution after inserting customer j and s_0 the solution directly before, i.e., without customer j. C_j^{hist} measures the accumulated increase in costs when inserting customer j. These costs approximate the additional cost effort when inserting customer j into a given solution.
- $C_j^{\min}$ quantifies the minimum insertion costs of customer j found so far. It is updated every time a new insertion position with lower costs is found, i.e., every time $\Delta_j < C_j^{\min}$.
- n_j counts the number of times customer j is reinserted.

1.5.2.2 Determination of Removal Candidates

The set of candidates to be removed, C_R^C, defines a newly designed feature in the ALNS proposed. The set is determined by comparing the current solution s with the global best solution $\hat{s}$. We use two variants for selecting which customers are added to set C_R^C. One of these two alternatives is randomly chosen in every iteration. The probability depends on the performance and is adapted during the search (see Section 1.5.2.7).

Sequence Comparison A customer is added to C_R^C if its successor in s differs from its successor in $\hat{s}$. This indicates that the customer is oddly placed compared to the position in the best solution found so far. Customers who do not differ from their successors in $\hat{s}$ are nevertheless added to set C_R^C with a probability p^{binom} for diversification purposes. We use this procedure since we expect that a certain number of edges in the best solution found so far will be similar to the optimal solution. We therefore combine the information from different solutions in a less random fashion than in a crossover but still include the information of the whole population.

Time Window Comparison A second variant of the procedure to determine C_R^C is implemented where the TW assignment of s and $\hat{s}$ is compared instead of the immediate successor. To be more specific, a customer will be added to C_R^C if the customer has a different TW assigned in s and $\hat{s}$.

1.5.2.3 Removal Operators

The maximum number of customers to be removed, q^{binom}, is sampled from a binomial distribution in every iteration (Voigt and Kuhn 2022). The binomial distribution takes two parameters, namely the sample size that we set to $|C|$ and the probability p^{binom} that is adapted during one ALNS run. The dynamic adaptation of this parameter is a new component that we use to cope with

different instance characteristics (e.g., APs, geographical distributions, vehicle capacities). In some instances only a few customers should be removed, reducing the ALNS to a more local search. In other instances, a high number of customers have to be removed in order to change the solution. The number of customers that are actually removed is expressed by $q^{\mathrm{r}} = \min\{q^{\mathrm{binom}}, |C_R^C|\}$. The ALNS uses four known removal operators (Ropke and Pisinger 2006), which we adapt to our problem setting. We have chosen simple yet efficient operators and further optimize them to find reasonable solutions within short runtimes.

Random Removal The *Random Removal Operator* randomly selects q^{r} customers out of set C_R^C and removes these customers from the current solution. This operator is fast and fosters diversification.

Historic Cost Removal The *Historic Cost Removal Operator* uses the historic information gained during the insertion procedures (see Section 1.5.2.1) and removes q^{r} customers from set C_R^C in decreasing order of average historic penalized costs, i.e., in decreasing order of $\frac{C_j^{\mathrm{hist}}}{n_j}$. The operator thus uses average historic penalized cost rather than just the costs encountered during the previous insertion phase. Otherwise the operator would show the tendency of removing the customers who have been inserted at the end of the previous insertion step, because these customers are constrained most concerning available insertion possibilities and therefore naturally show higher costs.

Worst Cost Removal The *Worst Cost Removal Operator* uses the historic information collected during insertion, too. The operator calculates the change in transportation costs, Δ_j, if customer j is removed from the current solution. It then compares this cost difference to the minimum encountered penalized cost during the search $C_j^{\min}$. Customers are removed in descending order of $\Delta_j - C_j^{\min}$, i.e., customers with a high difference in costs, who therefore seem to be oddly placed in the solution, are removed. The operator is similar to the well-known *Worst Removal Operator*, but overcomes the tendency of removing the same customer all over again only because the customer is for example far away from every other customer and therefore naturally increases the costs the most.

Shaw Removal The *Shaw Removal Operator* removes customers from the solution that are similar to each other. The first customer is removed with the *Random Removal Operator*. The following requests are chosen from C_R^C in increasing order of relatedness. The relatedness $Rel(c_1, c_2)$ of two customers c_1 and c_2 is measured by the distance of both customers and the similarity of their earliest completion times $E(j)$ in s. The lower the $Rel(c_1, c_2)$, the more related are customers c_1 and c_2.

$$Rel(c_1, c_2) = \frac{c_{c_1,c_2}}{\max\limits_{i,j \in N} c_{ij}} + \frac{|E(c_1) - E(c_2)|}{l_0} \tag{1.18}$$

Customers with a lower $Rel(c_1, c_2)$ are removed first.

1.5.2.4 Determination of Insertion Order

At the beginning of the insertion process, the set of removed customers, C_R is sorted in descending order of average historic costs $\frac{C_j^{\text{hist}}}{n_j}$. Customers who are hard to insert without incurring high penalized costs are inserted earlier and therefore have a greater chance of being inserted at an appropriate position. The later a customer is inserted into a tour, the more difficult it is to find a position that will increase the total costs at a moderate level only. Customers with low average historic costs should therefore be inserted at a later stage.

1.5.2.5 Insertion Operators

Two decisions have to be made when inserting a customer into a route: the position within the sequence of already assigned customers and the TW when the customer should be served. The selection of an appropriate TW can however influence the sequence and the TWs of already assigned customers. Within the ALNS, we therefore restrict the insertion operators to only allow the insertion of customers if TWs and sequences of customers on the route are not shifted. Based on this general strategy we define two variants of the *Best Insertion Operator*. Both variants iterate across all routes to find the insertion position that increases the cost least. Note that we again avoid the use of computationally intensive operators such as k-regret that are otherwise widely used in the ALNS literature (e.g., Larsen and Pacino 2019).

Best Insertion: First Feasible TW. For a given insertion position in a route, the *Best Insertion: First Feasible TW Operator* uses the earliest feasible TW. The feasibility of a selected TW can easily be checked in constant time by using the concept of reoptimization by concatenation of sequences (Vidal et al. 2013b). In the worst case, each and every TW has to be checked for every insertion position on a route, resulting in a worst-case complexity for the insertion of one customer of $\mathcal{O}(n|TW|)$. Here, n denotes the current number of customers on a route and $|TW|$ indicates the number of TWs.

Best Insertion: Best Feasible TW. The *Best Insertion: Best Feasible TW Operator* uses the TW that leads to the lowest penalized costs (compared to the first feasible TW) without shifting the sequence or the TW assignment of all other customers on that route. This operator has an average and worst-case complexity of $\mathcal{O}(n|TW|)$.

1.5.2.6 Simulated Annealing

The initial temperature is determined for every instance with $\gamma = -\frac{\overline{\Delta E}}{ln(\chi_0)}$ using the formula from Johnson et al. (1989) as cited in Ben-Ameur (2004). $\overline{\Delta E}$ estimates strictly positive cost increases and χ_0 expresses the probability of accepting a worse solution. We execute n_0 iterations of the ALNS to generate the transitions. The temperature γ thus determined is reduced by β after each iteration.

1.5.2.7 Adaptivity

The score of an operator is increased by either σ_1, σ_2 or σ_3. It is increased by σ_1 if a new best solution is found, by σ_2 if a previously unknown solution with lower costs than the current solution is found, or by σ_3 if the solution has higher costs but is accepted through the simulated annealing procedure. The probability of choosing removal and insertion operators depends on the scores and is calculated as in Ropke and Pisinger (2006).

1.5.3 Crossover Phase

The crossover phase uses the ALNS described in the previous section. In each generation, the ALNSs use the individuals in the current population one by one as a starting solution. The determination of the removal candidates is influenced by the global best solution (see Section 1.5.2.2). In a genetic algorithm, the idea of a crossover is to combine two individuals hoping to maintain good segments of a solution. In contrast, we try to achieve the same effect by reducing the probability of replacing well-placed customers. This procedure is based on the consideration that the optimal solution is likely to have some similarities with the solutions generated by ALNS and even more similarities with the best solution of the population.

1.5.4 Selection of Survivors

Surviving individuals are determined on the basis of total costs and the contribution of each individual to the diversity of the population similar to the diversity management of Vidal et al. (2012). All individuals of population P are placed in increasing order of total costs and a rank is assigned to the individual i, $Rank_i^{\text{Costs}}$, e.g., the individual with the lowest cost gets $Rank_i^{\text{Costs}} = 0$, the individual with the highest cost $Rank_i^{\text{Costs}} = |P| - 1$.

The diversity of an individual i is calculated by the hamming distance to all other individuals in the population based on the successor of nodes, $Succ^{\text{i}}$.

$$Hamming^{\text{ind}} = \sum_{i \in Population} \sum_{j \in C} Succ_j^{\text{ind}} \neq Succ_j^{\text{i}} \tag{1.19}$$

The population is placed in decreasing order of the hamming distance and a diversity rank is assigned to the individual $Rank_i^{\text{Diversity}}$. The overall rank is calculated by summing up the cost and diversity rank, $Rank_i^{\text{Total}} = Rank_i^{\text{Costs}} + Rank_i^{\text{Diversity}}$. Individuals with the n^{P} lowest total ranks are chosen for the next generation. Individuals with the $4 \cdot n^{\text{P}}$ lowest total ranks survive, meaning that an individual can be used in later generations, even if not used during previous generations. Lastly, after each fifth generation new individuals are generated in the same manner as the first generation to further enhance diversity.

1.6 Numerical Experiments

This section presents our numerical experiments based on simulated and empirical data. The HALNS is implemented in *C++* and run on an AMD Ryzen 9 3900X with 32GB RAM. Sec-

tion 1.6.1 details how we generated VRPAP instances. Section 1.6.2 evaluates the performance of the HALNS on related time-constrained VRP instances from literature and VRPAP instances generated, and additionally analyzes HALNS components. Section 1.6.3 generates managerial insights. Section 1.6.4 presents a case study with real APs. Table 1.2 gives an overview of experiments and data sets.

Table 1.2: Experiments and data sets

Section	Experiments and purpose	Data set used
1.6.1	VRPAP Instance Generation	Solomon (1987), Florio et al. (2018)
1.6.2	Performance Evaluation	
1.6.2.1	Benchmark for Time-Constrained VRPs	Solomon (1987), Belhaiza et al. (2014)
1.6.2.2	Benchmark for VRPAP Instances	Synthetic VRPAP instances generated in 1.6.1
1.6.2.3	Analysis of Algorithmic Components	Synthetic VRPAP instances generated in 1.6.1
1.6.3	Sensitivity Analysis and Managerial Insights	Synthetic VRPAP instances generated in 1.6.1
1.6.4	Case Study with Empirical Availability Profiles	Solomon (1987), Time Use Survey Data (Gershuny and Sullivan 2017)

1.6.1 VRPAP Instance Generation

We use instances for the VRPTW by Solomon (1987) and replace the TWs by different APs derived from Florio et al. (2018). Every profile consists of 10 TWs and has an average availability of 50% (see Table 1.3). The A-profile has a peak during midday. The V-profile has two peaks, one in the morning and one in the evening. The W-profile has an additional peak during midday and can therefore be considered a combination of A and V profiles. The M-profile is the opposite of the W-profile, i.e., when the availability of the W-profile is high, the availability of the M-profile is low. We generate instances based on these profiles where all customers of one instance have the same profile (A, V, W or M), and instances with a combination of profiles (AV, WM, and AVWM). In the latter case, customers of one instance are randomly assigned one of the respective profiles in equal proportions. We combine the seven different pure and mixed APs (A, V, W, M, AW, WM and AVWM) with six geographical distributions (R1, R2, C1, C2, RC1 and RC2) of customers from Solomon (1987), resulting in 42 instances.

1.6.2 Performance Evaluation

Section 1.6.2.1 evaluates the performance of the HALNS on special cases of the VRPAP, namely the VRPTW and the VRPMTW, as well as the more complex VRPSTW. Section 1.6.2.2 compares

Table 1.3: Availability profiles

	1	2	3	4	5	6	7	8	9	10
A	0.1	0.3	0.5	0.7	0.9	0.9	0.7	0.5	0.3	0.1
V	0.9	0.7	0.5	0.3	0.1	0.1	0.3	0.5	0.7	0.9
W	0.8	0.4	0.1	0.4	0.8	0.8	0.4	0.1	0.4	0.8
M	0.2	0.6	0.9	0.6	0.2	0.2	0.6	0.9	0.6	0.2

the HALNS solution for VRPAP instances to a lower and upper bound. Section 1.6.2.3 evaluates the importance of specific algorithmic components of the HALNS. We use the identical set of algorithmic parameters (see Appendix A.1) that has been selected after preliminary testing for all experiments.

1.6.2.1 Benchmark for Time-Constrained VRPs

Tables 1.4 to 1.7 summarize the results. Column *BKS* shows the (previous) best-known solution averaged over distribution types, column *Best X* shows the averaged best results obtained during 5 or 10 runs, and column *Avg X* demonstrates the arithmetic mean over 5 or 10 runs (combined by *Best/Avg* if only one run has been executed). Row Σ presents the cumulated objective values over the whole instance set, row *Avg gap* represents the average gap compared to the (previous) BKS, and row *#BKS* the number of BKS found. Furthermore, row *Avg T* shows the arithmetic mean of the runtime over 5 or 10 runs. We also specify the processors used and their passmark single thread rating (`https://www.cpubenchmark.net/`) in order to make the runtimes comparable.

VRPTW Benchmark Table 1.4 shows that the performance of the HALNS is comparable to well-performing algorithms for the VRPTW from Nagata et al. (2010) and Vidal et al. (2013b). Figure A.1 in Appendix A.2.1 additionally demonstrates that the HALNS also clearly outperforms the sole implementation of the ALNS by Pisinger and Ropke (2007) for smaller runtimes. This indicates the value of the hybridization of the ALNS.

Table 1.4: Summarized results for Solomon (1987) VRPTW instances

	BKS	Pisinger	Nagata	Vidal		HALNS	
		Best 10	Best 5	Best 5	Avg 5	Best 5	Avg 5
R1	1210.34	1212.39	1210.34	1210.69	1211.49	1210.35	1210.93
R2	951.03	957.72	951.03	951.51	952.05	951.03	951.77
C1	828.38	828.38	828.38	828.38	828.38	828.38	828.38
C2	589.86	589.86	589.86	589.86	589.86	589.86	589.86
RC1	1384.17	1385.78	1384.17	1384.17	1384.81	1384.17	1384.18
RC2	1119.24	1123.49	1119.24	1119.24	1119.4	1119.34	1119.43
Σ	57187	57332	57187	57196	57218	57188	57204
	Avg T	150s	300s	161s		155s	
	CPU	Pentium 4 3GHz	Opteron 2.4GHz	Xeon 2.93GHz		Ryzen 9 3900X	
	Passmark	561	430	1418		2731	

VRPMTW Benchmark Table 1.5 shows that the HALNS achieves a similar best gap to that of the ALNS of Larsen and Pacino (2019), which is the state-of-the-art approach for the VRPMTW. However, the HALNS outperforms the ALNS in terms of average performance. The HALNS produces 28 BKS out of 48 instances, including 12 newly found BKS. Note that the HALNS already produces reasonably good solutions at much shorter runtimes, as shown in Figure A.2 in Appendix A.2.2.

Table 1.5: Summarized results for Belhaiza et al. (2014) VRPMTW instances

BKS		Belhaiza 2014		Belhaiza 2017	Larsen		Schaap		Hoogeboom	HALNS	
		Best 10	Avg 10	Best/Avg	Best 10	Avg 10	Best 10	Avg 10	Best/Avg	Best 10	Avg 10
R1	2727.8	2740.8	2755.1	2732.3	2728.9	2731.1	2738.6	2773.3	2738.1	2729.5	2731.3
R2	2674.3	2810.7	2826.4	2793.9	2684.7	2687.3	2691	2702.1	2679.4	2683.5	2683.9
C1	3217.6	3297.3	3315.4	3278.2	3227.6	3256.7	3243.4	3266.3	3254.5	3231.1	3257.3
C2	4156.2	4192.7	4216.5	4169	4156.2	4166.7	4163.9	4182	4177.8	4158	4162.9
RC1	3218.4	3244.4	3258	3229.3	3218.4	3220.7	3222.6	3241	3235.1	3218.2	3219.3
RC2	2730	2899.7	2919.2	2879	2731.8	2789.1	2732.6	2845.4	2875.5	2727.7	2728.4
Σ	149795	153484	154324	152653	149981	150812	150337	152082	151683	149983	150265
Avg gap	0.00%	2.60%	3.16%	2.04%	0.13%	0.70%	0.37%	1.60%	1.33%	0.13%	0.31%
# BKS	36/48	2		8	27		5		5	28 (12 new)	
	Avg T	64s		81s	600s		185s		113s	629s	
	CPU	i5 3.3GHz		i5 3.3GHz	i7-4790K		i7 3.7GHz		i7 4GHz	Ryzen 9 3900X	
	Passmark	1704		1704	2469		2776		2469	2731	

VRPSTW Benchmark For the VRPSTW, we focus on the type 1 (only lateness considered) and type 2 (earliness and lateness considered) variants of the VRPSTW with $\alpha = 1$ (see, Fu et al. 2008). The parameter α sets the amount of linear penalty depending on the earliness/lateness. Setting $\alpha = 1$ means that one time unit of earliness/lateness equals one unit of transportation costs. While the VRPTW and the VRPMTW are special cases of the VRPAP and can be directly solved by the HALNS, the VRPSTW constitutes an extension to the VRPAP. The HALNS therefore needs some adaptations for solving VRPSTW variants. For the case of type 1 we need to adapt the objective function and allow solutions where customers are delivered after their desired TW. For type 2 we additionally need to adapt how operators determine the start of service for customers on a route. We replace every TW by two artificial TWs (with the same earliest and latest start times as the original single one), from which the HALNS may choose one. The first TW only allows earliness, the second TW only allows lateness. If the first TW is chosen, the customer will be served as early as possible even if this results in an earliness penalty. In contrast, if the second TW is chosen, the service does not start before the earliest start time of the TW, which means the vehicle has to wait if it arrives too early. This allows the use of the same operators as before to approximate the earliness/lateness. Note that penalties for earliness/lateness during the insertion phase are only valid for the customer, not for the overall route. The overall route is only evaluated after every customer is inserted.

Tables 1.6 and 1.7 show the summarized results. For type 1 instances, the HALNS achieves results similar to the currently best approach from Vidal et al. (2014b). For type 2 instances, the HALNS reveals a better average gap and finds 8 new BKS. This demonstrates the flexibility of the HALNS. Detailed results may be found in Appendix A.2.3.

1.6.2.2 Benchmark for VRPAP Instances

Table 1.8 shows summarized results of the HALNS algorithm when solving the 42 VRPAP instances generated (see Section 1.6.1). As a comparison we use the lower and upper bound described in Section 1.4 as benchmark. Column *Instance Group* aggregates the results by APs, as there are no significant differences between geographical distributions (see Table A.6 in Appendix A.2.4). The next three columns show the lowest possible distances costs, achieved by solving a VRP denoted by $C^{\text{trans}}_{\text{VRP}}$, the lowest possible failed-delivery costs $C^{\text{failed}}_{\text{VRPTW}}$, and the lower bound, derived by the summation of both terms before, denoted by $C^{\text{lower}}_{\text{VRPAP}}$. Column *Best 10* and *Avg 10*, show the best

Table 1.6: Summarized results for VRPSTW instances type 1 (only lateness), $\alpha = 1$

	BKS	Kritzinger		Vidal		HALNS	
		Best 10	Avg 10	Best 10	Avg 10	Best 10	Avg 10
C1	828.38	828.38	828.38	828.38	828.38	828.38	828.38
C2	589.86	589.86	589.86	589.86	589.86	589.86	589.86
R1	1170.11	1171.37	1174.9	1170.16	1171.11	1170.35	1170.73
R2	946.17	952.39	972.44	946.17	947.35	946.68	946.82
RC1	1313.9	1313.9	1315.34	1313.9	1314.04	1314.26	1314.94
RC2	1106.61	1108.26	1114.89	1106.61	1107.37	1106.61	1106.8
Σ	55987.56	56084.33	56411.68	55988.16	56019.79	55999.01	56012.04
Avg gap	0.00%	0.19%	0.81%	0.00%	0.06%	0.02%	0.04%
# BKS	56/56	36		55		50	
	Avg T	600s		349s		156s	
	CPU	Xeon E7-8837		Opteron 2.2GHz		Ryzen 9 3900X	
	Passmark	1124		445		2731	

Table 1.7: Summarized results for VRPSTW instances type 2 (earliness and lateness), $\alpha = 1$

	BKS	Vidal		HALNS	
		Best 10	Avg 10	Best 10	Avg 10
C1	828.38	828.38	828.38	828.38	828.38
C2	589.86	589.86	589.86	589.86	589.86
R1	1164.86	1164.86	1167.73	1167.38	1169.66
R2	949.87	949.87	957.44	945.99	946.78
RC1	1304.23	1304.23	1304.41	1309.14	1310.21
RC2	1106.43	1106.43	1108.42	1105.98	1106.09
Σ	55886.4	55886.4	56021.42	55909.8	55955.17
Avg gap	0.00%	0.00%	0.26%	0.00%	0.07%
# BKS	48/56	48		31 (8 new)	
	Avg T	1797s		159s	
	CPU	Opteron 2.2GHz		Ryzen 9 3900X	
	Passmark	445		2731	

and average result of ten HALNS runs. Column $C^{\text{upper}}_{\text{VRPAP}}$ signifies the upper bound. The next two columns Δ^{lb} and Δ^{ub} represent the percentage gaps to the lower and upper bound, respectively. Lastly, column *Avg T [s]* shows the average runtime of ten runs in seconds.

Compared to the upper bound procedure, the HALNS systematically makes use of APs and therefore finds a balance between transportation and expected failed-delivery costs. This means the HALNS is able to reduce expected costs if information on the AP is at hand. The lower bound is quite weak if the APs are homogenous and only few peaks exist (A, V, M), but becomes slightly better if the APs are more diverse or more peaks exist, i.e., the gap between the objective value obtained by the HALNS and the lower bound becomes smaller. This result is intuitive, because diverse APs (AV, WM, AVWM) and profiles with more peaks (W) offer more flexibility for altering the service time and thereby reducing expected failed-delivery costs without increasing the distance costs.

Table 1.8: Summarized results for VRPAP instances

Instance Group (R101, ..., RC201)	$C^{\text{trans}}_{\text{VRP}}$	$C^{\text{failed}}_{\text{VRPTW}}$	$C^{\text{lower}}_{\text{VRPAP}}$	Best 10	Avg 10	$C^{\text{upper}}_{\text{VRPAP}}$	Δ^{lb}	Δ^{ub}	Avg T [s]
A	762.88	76.29	839.16	1070.67	1074.72	1094.74	28%	-2%	955
V	762.88	76.29	839.16	1115.71	1118.6	1192.98	33%	-7%	908
W	762.88	152.58	915.45	1090.45	1094.82	1157.51	19%	-6%	973
M	762.88	76.29	839.16	1068.39	1071.04	1130.45	28%	-5%	891
AV	762.88	76.29	839.16	1064.36	1067.56	1134.55	27%	-6%	982
WM	762.88	114.43	877.31	1028.71	1031.23	1140.28	18%	-10%	947
AVWM	762.88	95.36	858.23	1049.22	1052.68	1140.41	23%	-8%	943
Avg	762.88	95.36	858.23	1069.64	1072.95	1141.56	25%	-6%	943

1.6.2.3 Analysis of Algorithmic Components

In order to gain insights into the importance of algorithmic components, we analyze the performance change of the HALNS when selected components are deactivated (−) or activated (+). The following configurations are tested.

HALNS: The baseline HALNS as described in Section 1.5.

(−) Simulated annealing: No simulated annealing within the ALNS (see Section 1.5.2).

(−) Crossover: No crossover (see Section 1.5.3).

(−) Removal candidates: No preselection of customers to be removed (see Section 1.5.2.2).

(−) Insertion order: No ordering of customers before inserting (see Section 1.5.2.4).

(+) Local improvement: Additional improvement procedure executed on the TW assignment of every new best solution found during the search based on the dynamic programming approach of Ibaraki et al. (2005) for solving the OSTP. The approach allows the determination of the optimal (i.e., penalty- and failed-delivery cost-minimal) service start times and respective TWs for a given sequence of customers on a route. Note that this sequence is retained throughout the algorithm and service start times are only altered by making use of the available waiting time in the tour.

We use the VRPAP instances from Table 1.8 and run each configuration ten times. Table 1.9 shows the average of the best of ten runs in Column *Best 10*, the average in Column *Avg 10* and the average runtime in seconds in Column *Avg T [s]*. All components, except *insertion order* and *local improvement* improve the best gap significantly. Also, all components (again, except for *local improvement*) further stabilize the average gap, resulting in a robust solution approach and additionally reduce the average runtime significantly. The findings are consistent over runtime (see Figure A.3 in Appendix A.3). To summarize, only the component *local improvement* is not beneficial for solving VRPAP instances, which means that the HALNS finds the optimal TW assignment without a dedicated algorithmic component. The slightly worse average performance indicates that results can even get worse when the TW assignment is optimized prematurely and the search gets stuck in a local optimum.

Table 1.9: Influence of specific components on HALNS performance

Algorithm	Best 10	Avg 10	Avg T [s]
HALNS	1069.64	1072.95	943
(−) Simulated annealing	1083.94	1101.33	1520
(−) Crossover	1073.73	1080.06	1227
(−) Removal candidates	1072.34	1079.04	1107
(−) Insertion order	1069.05	1073.53	1008
(+) Local Improvement	1069.41	1073.63	972

1.6.3 Sensitivity Analysis and Managerial Insights

In the following we generate insights into the benefits and implications of the VRPAP. Section 1.6.3.1 evaluates the cost savings potential of the VRPAP against current approaches. Section 1.6.3.2 analyzes which APs promise the highest cost reductions. Section 1.6.3.3 demonstrates the influence of c^{failed} on the solution structure. Section 1.6.3.4 analyzes the tradeoff between distance costs and failed-delivery rate. Section 1.6.3.5 examines the effect of applying a back-up policy when a delivery fails.

1.6.3.1 Cost Savings Potential

This section determines the cost savings achievable when using the VRPAP compared to applying myopic VRPMTWs with varying numbers of the most likely TWs. The VRPMTW includes the VRP (all TWs allowed) and the VRPTW (one most likely TW allowed) as special cases. This analysis quantifies the failed-delivery cost parameter c^{failed} as a multiple of $\overline{C^{\text{VRP}}}$. The cost value $\overline{C^{\text{VRP}}}$ denotes the average transportation costs per customer neglecting TWs, i.e., the cost resulting from solving the VRP divided by the number of customers served. This assumes that the CEP follows a policy in which customers whose delivery has failed are served at average delivery costs the next day.

We recalculate a reduced set of 32 instances from Table 1.8 for this experiment, as the approach with the most likely TW (i.e., the VRPTW) has no feasible solution for ten instances because this specific TW cannot be reached in time when originating from the depot. The upper section of Table 1.10 shows the average results when solved as VRPAP and as VRPMTW with 1, 3, 5, or 10 hard TWs that are chosen based on the highest customer availabilities. For $c^{\text{failed}} = 1 \cdot \overline{C^{\text{VRP}}}$, the results show that the VRPMTW approaches with the most likely TWs need considerably more vehicles compared to the VRPAP. Only the VRP solution uses the same number of vehicles as the VRPAP solution. The total costs of the VRP solution can be reduced by about 6% when solved as VRPAP. In this case, a slight increase of 2% of the transportation costs C^{trans} (from 710.41 to 725.32) is compensated by a reduction of failed delivery costs C^{failed} of 23% (from 347.26 to 267.00). This suggests that introducing a single or few multiple hard TWs in parcel delivery may only be worthwhile if the costs of failed deliveries are very high (e.g., for perishable goods). To confirm this conclusion, we reproduce the same experiment with more emphasis on failed delivery costs, i.e., $c^{\text{failed}} = 3 \cdot \overline{C^{\text{VRP}}}$ in the bottom section of Table 1.10. In terms of total costs C^{total}, the

Table 1.10: VRPAP vs. VRPMTW

c^{failed}	Approach	K	C^{trans}	C^{failed}	C^{total}
$1 \cdot \overline{C^{\text{VRP}}}$	VRPAP	4.84	725.32	267	992.32
	VRPMTW (1 most likely TW) $\widehat{=}$ VRPTW	27.38	2009.39	85.89	2095.28
	VRPMTW (3 most likely TWs)	10	1045.58	129.59	1175.17
	VRPMTW (5 most likely TWs)	7.06	853.4	196.44	1049.84
	VRPMTW (10 most likely TWs) $\widehat{=}$ VRP	4.84	710.41	347.26	1057.67
$3 \cdot \overline{C^{\text{VRP}}}$	VRPAP ($K = K^{\text{VRP}}$)	4.84	763.96	733.46	1497.42
	VRPAP ($K \leq K^{\text{VRPMTW 5TWs}}$)	6.78	819.06	501.81	1320.88
	VRPMTW (1 most likely TW) $\widehat{=}$ VRPTW	27.38	2009.39	257.68	2267.07
	VRPMTW (3 most likely TWs)	10	1045.58	388.78	1434.36
	VRPMTW (5 most likely TWs)	7.06	853.4	589.33	1442.73
	VRPMTW (10 most likely TWs) $\widehat{=}$ VRP	4.84	710.41	1041.79	1752.2

results look more favorable for the VRPMTW approach with the 3 or 5 most likely TWs, but at the cost of significantly more vehicles needed. When allowing the VRPAP to use the same number of vehicles as in the VRPMTW scenario with 5 TWs, the total costs are considerably lower and the VRPAP does not even use the maximum number of vehicles allowed ($K = 6.78$ against $K = 7.06$).

To further investigate the influence of the number of vehicles on the cost savings potential, we compare the VRPAP with the VRP approach as it is the only one to achieve the same number of vehicles. Figure 1.2 shows three different vehicle scenarios: (0) $K^{\text{VRPAP}} = K^{\text{VRP}}$, i.e., the VRPAP limits the number of available vehicles to the minimum number of vehicles required in the VRP cases, (1) one additional vehicle is available, and (2) two additional vehicles are available. Naturally, the VRPAP modeling approach becomes more important than the VRP approach when

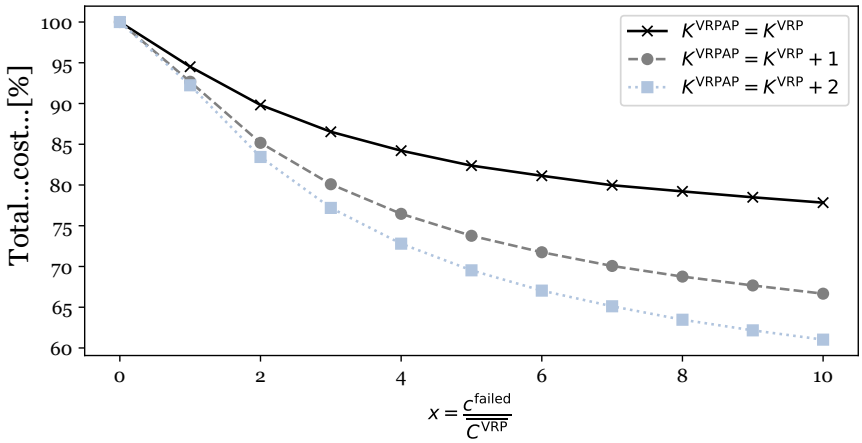


Figure 1.2: Total costs depending on c^{failed} and the number of vehicles used

the cost of failed-delivery increases. The cost differences are low assuming moderate failed-delivery costs ($c^{\text{failed}} \leq \overline{C^{\text{VRP}}}$, i.e., $x \in [0, 1]$), but become significant if a failed delivery becomes more costly than the average transportation costs per customer in a VRP solution ($c^{\text{failed}} > \overline{C^{\text{VRP}}}$, i.e., $x \in [2, ..., 10]$). Considering the case with an identical number of vehicles ($K^{\text{VRPAP}} = K^{\text{VRP}}$,

solid line), then at a certain point an increasing failed-delivery cost rate induces limited effects on the solution structure. At this point it becomes much harder to reach TWs with higher customer availability without inducing infeasibility. Thus, if CEPs want to increase their delivery fulfilment even more they have to increase the number of vehicles used (see dashed and dotted lines). The previous experiments have also shown, that in most cases it is not possible to reach all TWs with high or even highest customer availability without increasing the number of vehicles used. Only in an extreme case (i.e., with unlimited vehicles and high c^{failed}) the results achieved by the VRPAP modeling approach become equivalent to those achieved by an approach that first minimizes failed-delivery rates and then solves the resulting VRPTW.

1.6.3.2 Analysis of Availability Profiles

APs may be diverse in reality. It is therefore of interest how the different APs influence the potential cost savings achievable when these are explicitly considered in delivery planning. Figure 1.3 shows the cost savings and the cost distributions of the HALNS results from Table 1.8 compared to the respective upper bounds depending on the type of APs assumed. The lowest cost savings are possible for the A-profiles and the highest for the combination of W- and M-profiles. The share of distance costs increases when the APs are more diverse (AV, AVWM, WM) because in these cases it is worthwhile accepting additional distance in order to reach TWs with higher customer availability, which reduces expected costs for failed delivery. Interestingly, the V-profile also considerably saves costs but does not increase the share of distance costs significantly. This is a promising result since it can be assumed that many customers in reality show a V-profile.

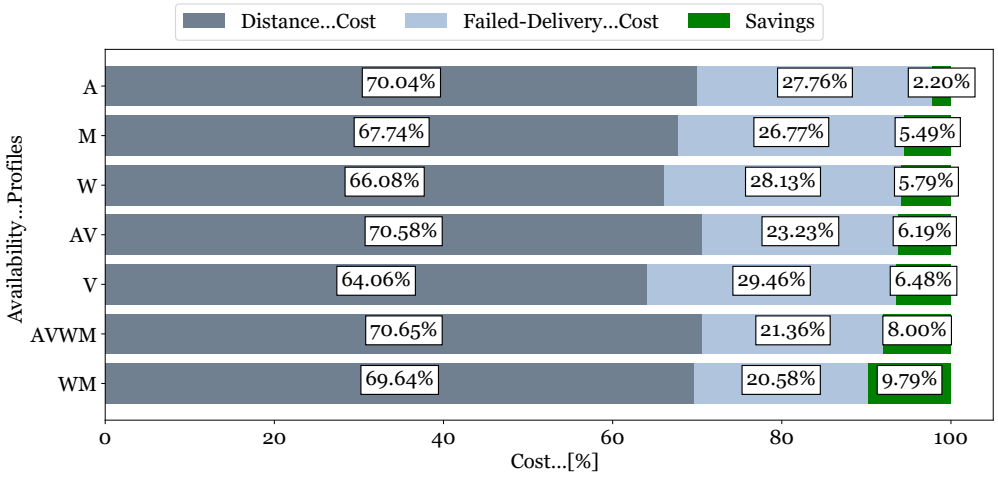
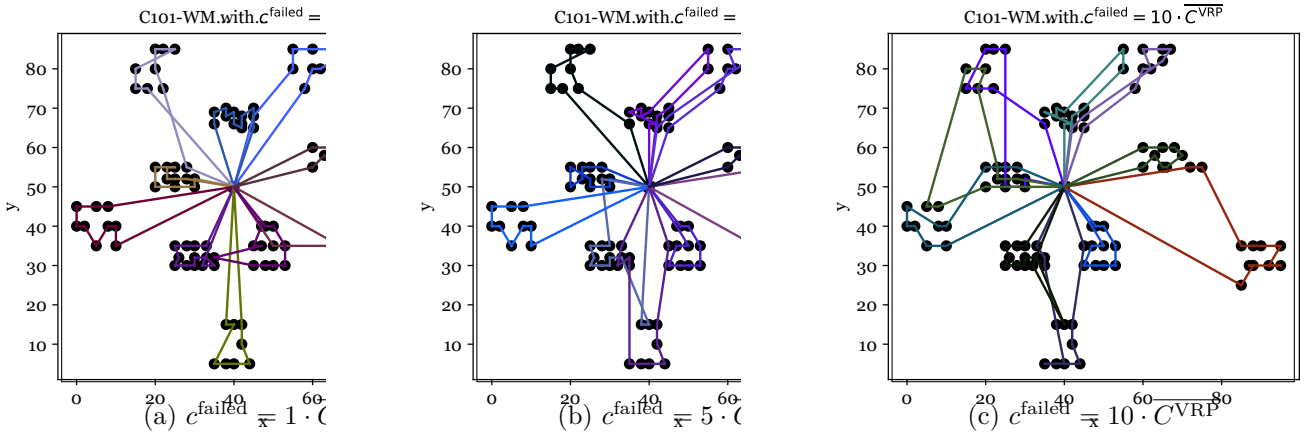


Figure 1.3: Cost savings and cost distributions in delivery planning depending on APs

1.6.3.3 Analysis of Solution Structure

In practice, drivers may change predefined routes on their own initiative if routes seem unreasonable to them. The question therefore arises when tours become apparently too long and drivers have to be motivated to follow the routes suggested even if these look unreasonable at first glance. To obtain an answer to this question, we analyze how the level of the failed-delivery cost parameter

c^{failed} affects the solution structure (see Figure 1.4). Equivalent to the above (see Section 1.6.3.1),



(a) $c^{\text{failed}} = 1 \cdot \overline{C}$ (b) $c^{\text{failed}} = 5 \cdot \overline{C}$ (c) $c^{\text{failed}} = 10 \cdot \overline{C^{\text{VRP}}}$

Figure 1.4: Solution structure depending on failed-delivery cost parameter c^{failed} for C101-WM

The routes appear quite reasonable if c^{failed} is low (see Figure 1.4a). The routes only show few crossings and customer clusters are mostly served by the same vehicle. The tours however become less intuitive as c^{failed} increases (see Figures 1.4b and 1.4c)). In these cases it is more important to reach a TW with higher customer availability than to save transportation costs. This also affects how customers are combined to tours. A single customer cluster is then even served by several vehicles.

1.6.3.4 Distance Costs vs. Failed-Delivery Rate

In the following we analyze the tradeoff between distance costs and failed-delivery rate. CEPs may be interested in the question regarding to what extent failed-delivery rates can be reduced in exchange for accepting higher transportation costs. To answer this question, we use all instances from Table 1.8. As in Section 1.6.3.1 we apply increasing failed-delivery cost rates ($c^{\text{failed}} = 0 \cdot \overline{C^{\text{VRP}}}, 1 \cdot \overline{C^{\text{VRP}}}, \cdots, 10 \cdot \overline{C^{\text{VRP}}}$). In addition, we limit the number of available vehicles to the minimum number of vehicles required in the corresponding VRP cases ($K^{\text{VRPAP}} = K^{\text{VRP}}$).

The lowest possible distance costs are achieved when we neglect failed-delivery costs, i.e., $c^{\text{failed}} = 0$. The average failed-delivery rate then results in a figure above 48%. This case equals the VRP modeling assumptions. Increasing the failed-delivery cost rate c^{failed} to $1 \cdot \overline{C^{\text{VRP}}}$, distance costs rise by about 3%, while the failed-delivery rate is brought down to 38%. The failed-delivery rate can be reduced to a minimum of approx. 34% if we put more emphasis on avoiding failed deliveries by increasing c^{failed}. Distance costs increase simultaneously. The failed-delivery rate improvement however becomes marginal, as it becomes increasingly difficult to reach TWs with higher customer availability without invoking infeasibility. More vehicles would be necessary, as mentioned in Section 1.6.3.1, to further improve the failed-delivery rate.

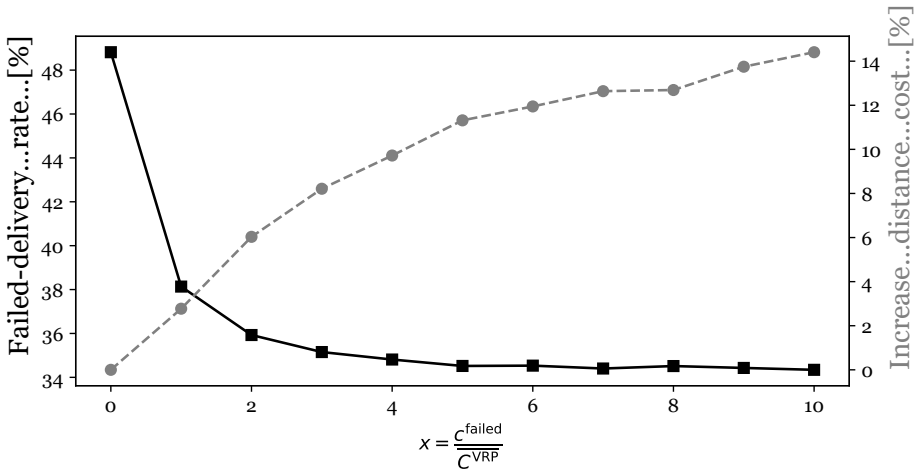


Figure 1.5: Failed-delivery rate vs. increase distance cost

1.6.3.5 Effect of Policies for Failed Deliveries

In this section, we analyze the effect on the subsequent day when the CEP applies the policy to deliver to a neighbor (if available) in the case of a first delivery attempt failing. We randomly decide for each customer if a neighbor is available with 50% probability. We assume that delivery to the neighbor is always successful and therefore no subsequent delivery attempt takes place after the first has failed. To reflect this we assume lower failed-delivery costs of $c^{\text{failed}} = 1 \cdot \overline{C^{\text{VRP}}}$ for customers with a neighbor compared to customers without a neighbor, where we assume $c^{\text{failed}} = 3 \cdot \overline{C^{\text{VRP}}}$ in a first, and $c^{\text{failed}} = 10 \cdot \overline{C^{\text{VRP}}}$ in a second experiment. This results in a ratio of failed-delivery costs for customers with and without a neighbor of either 1:3 or 1:10. We adapt the 42 instances from Table 1.8 and solve them either as VRP or VRPAP with the same maximum number of vehicles. Table 1.11 shows the average results. Columns C^{trans}, C^{failed} and C^{total} represent the transportation, failed and total costs. Column *Failed [%]* shows the failed-delivery rate, i.e., the share of first delivery attempts failing before the neighbor backup option is considered. Column *Neighbor[%]* shows the share of all parcels delivered to a neighbor and Column *Remaining [%]* the deliveries that remain for the delivery on the next day because no neighbor was available. The rows Δ *[%]* calculate the change in costs or shares, respectively, when comparing the VRP solution with the VRPAP solution.

Table 1.11: Results on VRPAP instances with neighbor delivery for failed deliveries

	C^{trans}	C^{failed}	C^{total}	Failed [%]	Neighbor [%]	Remaining [%]
Neighbor 1:3						
VRP	762.87	765.73	1528.61	49.97	24.82	25.15
VRPAP	808.90	511.16	1320.06	36.91	21.82	15.09
Δ [%]	6.03	-33.25	-13.64	-26.14	-12.09	-40.00
Neighbor 1:10						
VRP	762.87	2111.64	2874.51	49.97	24.82	25.15
VRPAP	870.51	1132.18	2002.69	37.27	25.12	12.15
Δ [%]	14.11	-46.38	-30.33	-25.42	1.21	-51.69

While the transportation costs C^{trans} increase by around 6% (14%), the failed-delivery costs C^{failed} are brought down by 33% (46%), which leads to a reduction of total costs by around 13% (30%) for a failed-delivery cost ratio of 1:3 (1:10) when using the VRPAP approach. Obviously, the reduction of total costs becomes higher if the cost for a failed-delivery attempt becomes higher. The share of failing deliveries is reduced by around 25%, independent of the failed-delivery cost ratio. While the VRP approach only makes arbitrary use of the neighbor option (for about 50% of failed deliveries), the VRPAP successfully identifies which customers have a neighbor and which do not. In the case of a cost ratio of 1:10, it allows more deliveries to be handed over to a neighbor in order to reach good TWs at customers without a neighbor and achieves more than 50% fewer remaining deliveries, compared to 40% fewer with the cost ratio 1:3. Only 12.15% compared to 15.09% of deliveries remain for the next day. However at higher cost ratios, customers who allow the delivery to a neighbor are increasingly discriminated against. This can be seen in the increasing number of deliveries taken to a neighbor (21% to 25%). This discrimination could dissatisfy these customers in the long run. To prevent this, c_j^{failed} for the customers whose previous delivery has been taken to a neighbor should be adapted for the following periods.

This experiment demonstrates that CEPs benefit in the long run (i.e., for subsequent days) if they use the VRPAP because remaining deliveries can be reduced significantly. Additionally, with the help of the VRPAP and appropriately set c_j^{failed}, CEPs can benefit even more from backup policies. Also, if the maximum number of failed parcels that could be delivered a second time on the next day due to capacity restrictions is known, the VRPAP can be used to respect this limit by setting c_j^{failed} accordingly on each day. We found similar results when considering a parcel shop as backup option instead of a neighbor where failed deliveries are taken to the parcel shop if available.

1.6.4 Case Study with Empirical Availability Profiles

We use the United Kingdom Time Use Survey (UKTUS) 2014-2015 (Gershuny and Sullivan (2017)) to generate realistic APs. The UKTUS is a large-scale survey that provides data on how people in the UK spend their time. Participants of the survey usually record events for a single day. Over a period of 24 hours on a weekday and a day at the weekend, participants indicate their main activity every 10 minutes and specify secondary activities as well as the place where the activity took place and with whom the respondent was together. The location and activity details are used to derive the daily routines of the residents and to identify whether a participant is at home. UKTUS data also makes it possible to discover correlations between this behavior and socio-economic characteristics that are representative for a large population. The UKTUS data set contains information on a total of 4,733 households. This corresponds to 11,422 individuals. We only consider weekdays from Monday to Friday and focus on times from 8-18 o'clock as these represent the main delivery days and periods of parcel service providers. In order to investigate attendance behavior, all persons are removed from the data record who have not provided information on location and activity and are therefore not useful for the creation of APs. The remaining data set contains 7,986 persons aged between 8 and 99 years. It should be noted here that some households may no longer be fully represented with all persons surveyed. In the following, we primarily consider people older than 15,

since it is assumed that younger children generally do not stay at home alone without older siblings or parents or are not allowed to accept parcels. A random sample of 7,221 persons remains. We distinguish between employed (employees and self-employed) and unemployed (including parental leave and the long-term sick) and different age classes. We choose these features as they seem to have the most effect on APs. Appendix A.4 shows ten different APs depending on age and employment status that we derived from UKTUS data and used to generate APs for Solomon instances (R1, R2, C1, C2, RC1 and RC2). We randomly sample from these APs and allocate each customer one of these ten APs in the Solomon instances. The probability of assigning a specific AP to a customer depends on the relative frequency of this profile, e.g., the relative frequency is calculated by $\frac{675}{7221} = 0.0935$ for employed people between 15 and below 30. The sampling is repeated 30 times, resulting in 30 AP distributions for each of the six geographical distributions of customers from Solomon, leading to 180 instances overall.

We solve all instances ten times with HALNS with $c^{\text{failed}} = 3 \cdot \overline{C^{\text{VRP}}}$ and present the average (over 30 instances per geographical distribution) of the best result encountered during these 10 runs in Table 1.12. On average the distance cost increases from 762.88 to 805.45 whereas the rate of failed deliveries decreases from 0.585 to 0.5156. This corresponds to 5.58% rise in distance costs in order to decrease the failed-delivery rate by 11.86%. Total expected costs decrease by 4.82% from 2103.56 to 2002.15. A Wilcoxon test confirmed that all differences in total costs and in the share of failed deliveries are highly significant ($p < 0.001$). To summarize, costs are reduced by about 5% by only using information on the age and employment status. This confirms the value of the VRPAP even without using sophisticated methods to determine customer individual APs.

Table 1.12: Average Results on Solomon Instances with UK TUS APs

Instance	$C^{\text{Dist}}_{\text{VRP}}$	$P^{\text{Failed}}_{\text{VRP}}$	$C^{\text{Failed}}_{\text{VRP}}$	$C^{\text{Total}}_{\text{VRP}}$	$C^{\text{Dist}}_{\text{VRPAP}}$	$P^{\text{Failed}}_{\text{VRPAP}}$	$C^{\text{Failed}}_{\text{VRPAP}}$	$C^{\text{Total}}_{\text{VRPAP}}$
R101-TUS	865.95	0.5873	1525.72	2391.67	872.49	0.5597	1453.92	2326.41
R201-TUS	651.3	0.5873	1147.53	1798.83	725.03	0.4519	882.89	1607.92
C101-TUS	824.78	0.5796	1434.13	2258.91	830.49	0.5384	1332.24	2162.73
C201-TUS	584.28	0.5745	1007.01	1591.29	667.11	0.5049	885.08	1552.19
RC101-TUS	995.59	0.5949	1776.83	2772.42	1008.37	0.5719	1708.04	2716.41
RC201-TUS	655.35	0.5864	1152.89	1808.24	729.19	0.467	918.06	1647.24
Average	762.88	0.585	1340.69	2103.56	805.45	0.5156	1196.71	2002.15

1.7 Conclusions and Future Areas of Research

Conclusions In this paper we present a time-constrained vehicle routing problem, the vehicle routing problem with availability profiles that addresses the tradeoff between routing and expected failed-delivery costs. We use so-called availability profiles that indicate the probability of successful delivery for a set of potential delivery TWs. We show how these APs can be integrated into an optimization model for delivery route planning such that routing and expected failed-delivery costs are considered simultaneously. We propose a novel hybrid adaptive large neighborhood search solution framework for the VRPAP and related time-constrained VRPs. The HALNS embeds an

ALNS into a population-based metaheuristic. The HALNS shows results on a par with the best-performing algorithms from the literature. For the VRPMTW it finds 12 and for the VRPSTW 8 new BKS. The hybrid approach shows superiority to sole ALNS implementations. We also undertake diverse numerical studies on newly generated VRPAP instances based on simulated and real-world data that confirm the value of integrated planning of delivery tours and delivery TWs. These studies additionally reveal several managerial insights:

(a) The VRPAP improves the total cost by 6% on average compared to the best-performing benchmark approaches that are potentially applied in practice. The number of vehicles may have to be increased to exhaust all cost benefits when tours and TWs are planned simultaneously.

(b) More peaks within APs lead to higher cost savings. An A-profile thus offers fewer opportunities for cost improvement than W- and M-profiles or a combination of these. V-profiles that can frequently be observed in reality considerably save total costs without increasing distance costs significantly.

(c) In cases where delivery fulfilment has a high priority, overall transportation costs can substantially increase. Customers in the same region could even be served by several vehicles in such instances. Drivers must then be motivated to follow their routes, even if the tours created seem unreasonable at first glance.

(d) A small increase in transportation costs may noticeably increase delivery fulfilment. Above a certain level, however, a further increase in delivery fulfilment leads to disproportionately high additional transportation costs.

(e) CEPs may benefit from using the VRPAP on subsequent days because the number of remaining deliveries can be reduced significantly, especially if there are additional backup options available.

(f) Generally available socio-economic data are already sufficient to define APs of customers. In an experiment close to reality the failed-delivery rate is reduced by about 12% while overall costs decrease by about 5%.

Outlook on Future Research There are several opportunities for improvement and extension based on the proposed modeling and solution approach. The determination of APs could be further developed, e.g., by focusing more intensively on individual customer behaviors. Also, customer presence is becoming a critical success factor for new delivery technologies, such as drone parcel delivery or mobile parcel lockers (see, e.g., Schwerdfeger and Boysen 2020). The VRPAP modeling and solution approach could be extended in these directions. Finally, as the HALNS framework mostly uses simple operators, it could be easily adapted to solve related routing problems, while the benefits from hybridization would be retained.

Contribution 2 - Hybrid Adaptive Large Neighborhood Search for Vehicle Routing Problems with Depot Location Decisions

2 Hybrid Adaptive Large Neighborhood Search for Vehicle Routing Problems with Depot Location Decisions

Stefan Voigt, Markus Frank, Pirmin Fontaine, Heinrich Kuhn

Abstract This article considers three variants of the vehicle routing problem (VRP). These variants determine the respective depot locations from which customers are supplied, i.e., the two-echelon VRP (2E-VRP), the location routing problem (LRP), and the multi-depot VRP (MD-VRP). Both the LRP and the MDVRP can be formulated as special cases of the 2E-VRP, so that all three problem classes can be readily solved via a single solution approach. We develop such a unified solution approach for all three problem classes based on the recently proposed hybrid adaptive large neighborhood search (HALNS). The HALNS uses a population of solutions generated by an efficient ALNS. Individuals of this population are subject to a crossover and selection phase, using elements of genetic algorithms resulting in a hybrid heuristic. Computational experiments on several sets of instances from literature demonstrate the competitive performance of the HALNS. The HALNS outperforms all approaches for solving the 2E-VRP and is on par with heuristics that are dedicated either to the LRP or the MDVRP. Furthermore, the HALNS shows superior robustness, i.e., the variance of results from several runs is comparatively low. The HALNS especially outperforms all existing pure ALNS implementations on these problem classes, demonstrating the value of hybridization. Additionally, the HALNS finds three new best-known solutions for LRP instances.

Published: Computers & Operations Research
URL: `https://doi.org/10.1016/j.cor.2022.105856`

2.1 Introduction

The multi-depot vehicle routing problem (MDVRP) can be formulated as a special case of the location routing problem (LRP), and the LRP in turn as a two-echelon vehicle routing problem (2E-VRP) (Hemmelmayr et al. 2012). We classify these problems as *Vehicle Routing Problems with Depot Location Decisions*. Besides the clustering and sequencing as essential decisions in any vehicle routing problem (VRP), problem settings in this class are characterized by decisions about which of multiple depots to open and which customers to serve from each depot opened.

The most generic problem in this class is the 2E-VRP. The 2E-VRP arises for example in city logistics where goods are transported from a main depot outside the city to several satellites within the city, from where the goods are finally distributed to customers (e.g., Crainic et al. 2011). Two interdependent decisions have to be made in order to solve the 2E-VRP. First, we have to cluster customers to satellites and decide about the routing from the main depot to satellites (first level). Not all potential satellites necessarily have to be used. Second, we need to decide about the routing from satellites to customers (second level). We call the first decision *Depot Location Decision*, which includes the first-level routing, and the second decision *Routing Decision*. The MDVRP and LRP share these decisions with the 2E-VRP and can be interpreted as special cases of the 2E-VRP. The *Routing Decision* covers a classical VRP that is already $\mathcal{NP}$-hard (e.g., Vidal et al. 2013a). The 2E-VRP, LRP, and MDVRP themselves are therefore $\mathcal{NP}$-hard.

This paper, develops a unified heuristic solution approach to solve all three problems. Based on recent work by Voigt et al. (2021), we adapt and enhance the concept of a hybrid adaptive large neighborhood search (HALNS) that uses multiple solutions generated by an efficient ALNS as a population of individuals within a genetic algorithm framework. Numerical experiments based on established benchmark data sets for the 2E-VRP, LRP, and MDVRP show the robustness and good performance of the HALNS.

Our research contributes to the existing literature as follows. (1) We propose a novel solution approach that hybridizes an ALNS with elements of genetic algorithms for solving these problems. (2) We present a performance ranking of all heuristics reviewed from the literature for the problems named and compare their solution quality and scaled computation times. (3) We show that the HALNS outperforms approaches for the 2E-VRP and is on par with problem-specifically designed solution approaches for the LRP and MDVRP. Furthermore, the HALNS outperforms sole implementations of state-of-the-art ALNSs, making hybridization an interesting option to improve existing ALNS implementations. (4) We present insights on the components of the HALNS, showing the importance of using these components.

The remainder of this paper is structured as follows: Section 2.2 describes the vehicle routing problems with depot location decisions that we address with our heuristic and their relationships. In Section 2.3, we cluster previous heuristic solution approaches for these problems and analyze the frequency of heuristic components. We present the HALNS developed in Section 2.4 and conduct computational experiments in Section 2.5, including the evaluation of previous approaches and their comparison with the HALNS based on established benchmark sets. Finally, we conclude our work and indicate further directions for research in Section 2.6.

2.2 Vehicle Routing with Depot Location Decisions

We consider three VRP classes that determine the depot locations, namely the 2E-VRP, as well as the LRP and MDVRP as special cases of the 2E-VRP. These problems have in common the decision from which location, i.e., satellite, facility or depot, customers are served. We will henceforth use the term depot except when referring to the specific problem class, i.e., we use satellite when referring to 2E-VRP, facility for LRP, and depot for MDVRP. We first explain the 2E-VRP in detail in Section 2.2.1. Afterwards, we show how the LRP and MDVRP can be interpreted as special cases of the 2E-VRP in Section 2.2.2.

2.2.1 2-Echelon VRP

The 2E-VRP is a two-level routing problem with two fleets of vehicles, which minimizes costs arising from servicing customers via given satellites from a single main depot. First-level vehicles start from the main depot to satellite locations. Freight is then transferred from first-level vehicles at the satellite to second-level vehicles. The first-level routing problem constitutes a VRP with split deliveries, as it is possible that more than one vehicle delivers to a single satellite because of capacity restrictions. Second-level vehicles start from satellites to customers and execute the final delivery. Note that split deliveries are not allowed on the second level. The second level therefore represents a MDVRP, which is explained in more detail in Section 2.2.2.

The 2E-VRP is defined on a directed graph $G = (V, A)$. V is the set of nodes, consisting of the main depot subset $V_0, V_0 = \{v_0\}$, the satellite subset V_S with n_s satellites s, and the customer subset V_C with n_c customers i. Each customer node has an associated demand d_i. The arc set A consists of one set $A_1 = \{(i, j) : i, j \in V_0, V_S\}$ with arcs connecting the depot node with satellite nodes and the satellite nodes amongst each other, and a second set $A_2 = \{(i, j) : i, j \in V_S, V_C\}$ with arcs connecting satellite nodes with customer nodes and customer nodes amongst each other. This means that direct deliveries from the depot to customers are not possible. We assume a positive weight of arc (i, j), c_{ij}, that reflects the distance, traveling times or transportation costs between nodes i and j. Vehicles are limited in their number and have limited capacity on both levels. m_1 identical vehicles are available at the main depot for the first level and have a capacity of K_1 each. m_2 identical vehicles with capacity K_2 each are available in total for the second level. The satellites are implicitly capacitated by constraining the number of second-level vehicles m_s allowed to start from the satellite s, or explicitly by a capacity limit w_s. The flow of demand to and from each satellite must be balanced. Figure 2.1 illustrates an example for a 2E-VRP instance, with customers receiving their deliveries via satellites s_1, s_2, s_3 out of four possible satellites, which are in turn supplied by depot v_0, applying weights c_{ij} on the first level.

The objective in the 2E-VRP is to minimize the routing costs arising from the vehicle fleets traveling on the first and the second level by 1) assigning customers to satellites, 2) deciding on the clustering of customers to tours and their sequence on each tour for each satellite, and 3) the routing from the main depot to satellites, respecting the flow of demand. Perboli et al. (2011) present a corresponding MIP model.

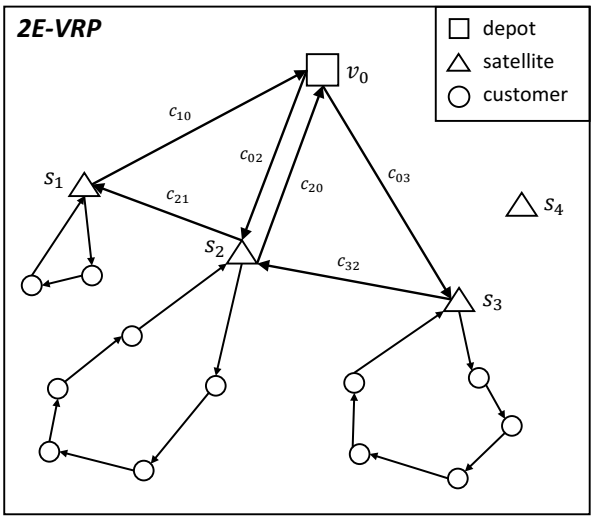


Figure 2.1: Exemplary 2E-VRP instance

2.2.2 Location Routing Problem and Multi-Depot VRP as Special Cases of the 2E-VRP

Location Routing Problem The LRP can be described as a strategic problem that consists of the facility location problem and the (multi-depot) vehicle routing problem. The LRP aims for the cost optimal locations of multiple facilities while considering the resulting routing of vehicles. Obviously these decisions interact with each other. In the following we consider the standard LRP, defined as a deterministic, static, discrete, single-echelon problem that minimizes costs arising from fixed costs f_s for opening (capacitated) facilities and travel costs in order to serve customers exactly once via one vehicle (Schneider and Drexl 2017). Figure 2.2 shows how an LRP instance can be formulated as a special case of the 2E-VRP presented above. The following adaptations must be made (see, Hemmelmayr et al. 2012):

- Facilities in the LRP correspond to satellites V_s in the 2E-VRP.
- A dummy main depot v_0 is created. Edges from the main depot to facilities V_S are respectively valued with fixed costs $c_{0s} = f_s$ for opening the facility $s \in V_S$. Edges from one facility to another are valued with a sufficiently large number. There is no cost on the return trip to the main depot v_0, i.e., $c_{s0} = 0$. A facility is thus opened when a first level vehicle visits that facility.
- m_1 is set to the number of facilities available.
- K_1 is set to the maximum capacity of all facilities to choose from. This ensures that split deliveries are not necessary and every facility can be served by exactly one vehicle, so the opening costs are only charged once.

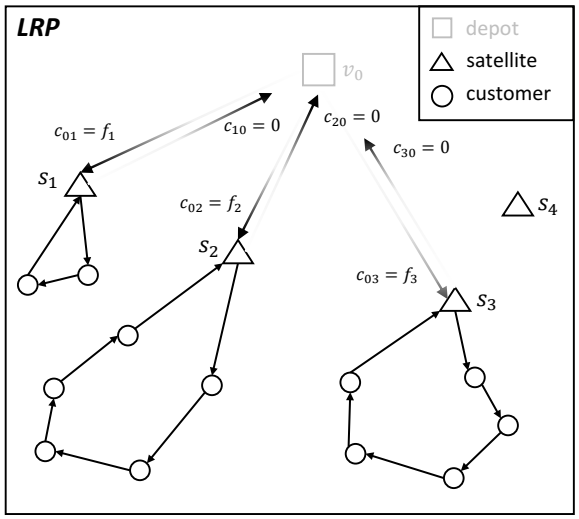


Figure 2.2: Exemplary LRP instance as special case of the 2E-VRP

Multi-Depot VRP In the MDVRP each customer is served exactly once from one of several available depots. Vehicles are assigned to a depot and must therefore start and end at the same depot. Depots do not have opening costs. Renaud et al. (1996), for example, present a mathematical model of the MDVRP. The MDVRP can be formulated as 2E-VRP by introducing a dummy main depot and treating all other depots as satellites. All travel costs on the first level are set to zero, i.e., $c_{s,0} = c_{0,s} = c_{st} = 0 \; \forall s, t \in V_s$, see also Figure 2.3. As mentioned in Section 2.2.1, the second level of the 2E-VRP is an MDVRP, making it obvious that the MDVRP is a special case of the 2E-VRP without considering the first-level routing.

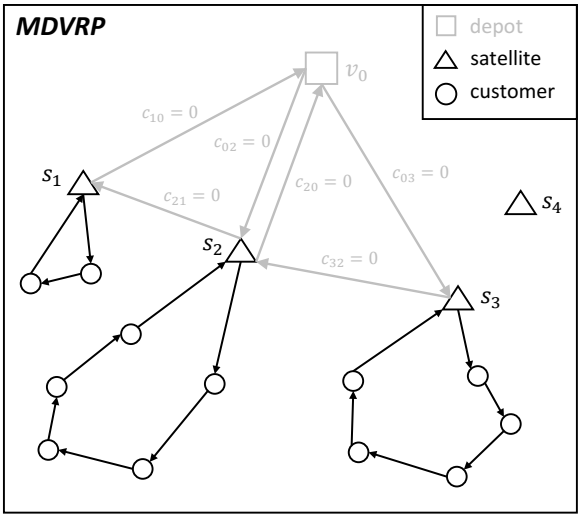


Figure 2.3: Exemplary MDVRP instance as special case of the 2E-VRP

2.3 Overview of Related Heuristics

All three problem classes considered in this study, the 2E-VRP, LRP and MDVRP, are known to be $\mathcal{NP}$-hard problems. The literature therefore suggests a diverse set of heuristic solution approaches. In this section we categorize and discuss the available approaches, focusing on those that have been applied to at least one of the commonly known benchmark instances (see Section 2.5.1). The review of solution approaches for each problem class is mainly structured according to the dominant solution approach, i.e., matheuristic or metaheuristic. Furthermore, we distinguish metaheuristic approaches with respect to their search space, i.e., neighborhood-based, population-based, or hybrid search space. In the following, we briefly clarify our understanding of these distinctions.

- **Heuristic approach**
  - *Matheuristic:* Combines exact and (meta)heuristic components.
  - *Metaheuristic:* Uses strategies to guide other heuristic components.
- **Search space of metaheuristics**
  - *Neighborhood-based:* Focuses on one incumbent solution.
  - *Population-based:* Uses a population of several solutions, so-called individuals.
  - *Hybrid:* Traverses the search space by using neighborhood-based and population-based means. Please note that merely combining local search with other approaches does not qualify for our classification as hybrid.

We focus on heuristic approaches for the standard versions of the respective problems. Exact approaches are out of scope for this review. We refer to the contributions of Marques et al. (2020) for the 2E-VRP, to Contardo et al. (2014) for the LRP and to Sadykov et al. (2021) for the MDVRP. For additional in-depth investigations and complete literature reviews we refer to Sluijk et al. (2022) for the 2E-VRP, to Schneider and Drexl (2017) for the LRP, to Drexl and Schneider (2015) and Prodhon and Prins (2014) for LRP variants and to Montoya-Torres et al. (2015) for the MDVRP.

2.3.1 2-Echelon VRP

Matheuristics Perboli et al. (2011) introduce a mathematical model for the 2E-VRP and strengthen their formulation with valid inequalities. They also develop two matheuristics based on the mathematical model and the observation that the 2E-VRP can be split into $n_k + 1$ VRPs, which in turn can be solved by any exact method or heuristic for the VRP. The authors therefore focus on finding near-optimal assignments for customers to satellites. Wang et al. (2017) propose a variable neighborhood search (VNS) and a post-optimization step based on integer programming to solve the 2E-VRP with environmental considerations. Amarouche et al. (2018) combine two components to solve the classical 2E-VRP. The first component generates a set of routes by a large neighborhood search (LNS). The second component uses integer programming to recombine the routes. Jie et al. (2019) decompose the problem by solving the first echelon with an exact column generation approach and the computationally more demanding second echelon with an ALNS largely based on Hemmelmayr et al. (2012) and Ropke and Pisinger (2006).

Neighborhood-based Metaheuristics Neighborhood-based approaches for the 2E-VRP usually tackle the first- and second-level routing separately. Crainic et al. (2011) present a multi-start heuristic based on this two-level approach. The approach iteratively solves the two subproblems by applying a local search, perturbing the solution and applying a feasibility search, if necessary. Hemmelmayr et al. (2012) develop an ALNS with specially designed operators for the 2E-VRP and the LRP. The heuristic works in a sequential manner like the approach presented previously. The authors distinguish between operators with large and small impact on the solution structure. Operators with a large impact open or close satellites, while operators with a small impact only execute minor changes. Solutions destroyed by operators with a large impact are accepted as a new incumbent solution, even if the solution is worse. In contrast, solutions destroyed by operators with a small impact must be better in order to be accepted. Hence, the solution space is explored by the large-impact operators, and the search intensifies by applying small-impact operators. Enthoven et al. (2020) and Yu et al. (2021) present similar ALNSs for variants of the 2E-VRP with a large set of small- and large impact operators. Compared to the rather sophisticated ALNSs, Breunig et al. (2016) implement a simplistic LNS with only few operators. They also provide consistent benchmark instances, resolving some confusion with previously existing instances.

Hybrid Metaheuristics The parallelized LNS (PLNS) of Mühlbauer and Fontaine (2021) is currently the only approach for the 2E-VRP hybridizing an LNS with a population of solutions. The authors develop the PLNS for solving a problem arising in city logistics when using cargo bicycles with swap containers, which can be modeled as a 2E-VRP. The authors suggest a new first-level heuristics and show the value of incorporating first-level costs when making decisions at the second level.

2.3.2 Location Routing Poblem

A large variety of heuristics for the LRP exist. Many of these approaches rely on a sophisticated combination of solution approaches; only few approaches like the tabu search of Tuzun and Burke (1999) or the simulated annealing (SA) of Yu et al. (2010) are straightforward implementations. Combinations of approaches are often based on a greedy randomized adaptive search procedure (GRASP) (Prins et al. (2006b), Duhamel et al. (2010), Contardo et al. (2013)) or a granular tabu search (GTS) (Prins et al. (2007), Escobar et al. (2013), Escobar et al. (2014a) and Schneider and Löffler (2019)).

Matheuristics Prins et al. (2007) implement a heuristic based on a GTS and a lagrangean relaxation to solve the LRP. Pirkwieser and Raidl (2010) tackle the LRP and periodic LRP (PLRP) with a VNS combined with three very large neighborhood searches based on integer linear programming (ILP) similar to Prins et al. (2007). Contardo et al. (2013) present a three-phase heuristic that (1) applies a GRASP based on Prins et al. (2006b) to construct initial solutions, (2) solves an ILP in order to recombine routes from the set of initial solutions similar to Pirkwieser and Raidl (2010), and (3) solves the same ILP to improve the solution.

Neighborhood-based Metaheuristics The tabu search of Tuzun and Burke (1999) works in two phases. First the location phase tries to find a good location configuration, and second the routing phase minimizes routing costs. The authors propose a set of benchmark instances that laid the foundation for comparing heuristics for the LRP. Yu et al. (2010) use a simplistic SA metaheuristic based on a special solution representation, encoding the solution in only one string. The problem is then solved by common neighborhood operators in an integrated manner compared to previous publications that tackled the problem by separating the location and routing decisions.

Escobar et al. (2013) propose a two-phase heuristic using a construction phase and as second phase a GTS with different diversification strategies. The heuristic uses a software library containing fast local search heuristics for the VRP, and works especially well on larger instances. A similar team of authors present a granular variable tabu neighborhood search, that combines a GTS with a VNS (Escobar et al. 2014a). Schneider and Löffler (2019) present a tree-based search algorithm. The algorithm uses a location phase that searches for a good set of facilities in a tree-like fashion and a routing phase that solves the resulting MDVRP by a GTS. The GTS is composed of a large set of 14 neighborhood operators. In contrast to that, the approach of Arnold and Sörensen (2021) uses only few operators based on the knowledge-guided local search (Arnold and Sörensen 2019). In addition, a progressive filtering technique significantly reduces the computational effort wasted on calculating routing solutions on unpromising depot configurations by estimating an upper bound of the number of open depots.

Population-based Metaheuristics Population-based approaches are again rarely used for the LRP, although the approaches of Ting and Chen (2013) and Lopes et al. (2016) show good results on benchmark instances. Prodhon and Prins (2008) implement a memetic algorithm, i.e., a genetic algorithm with local search, with population management for the PLRP and for the LRP (Prins et al. 2006a). Ting and Chen (2013) implement a multiple ant colony optimization algorithm (ACO) with two hierarchical ant colonies. The first colony solves the facility location problem, the second colony the MDVRP. Both colonies exchange information through a global pheromone updating rule. Lopes et al. (2016) propose a genetic algorithm with local search procedures as mutation operators and a so-called route copy crossover operator, which always maintains feasibility.

Hybrid Metaheuristics Prins et al. (2006b) propose a GRASP combined with a learning process, and a post-optimization step using path relinking. The authors were to the best of our knowledge the first to test the performance on three sets of instances from Tuzun and Burke (1999), Prins et al. (2004) and Barreto et al. (2007). Duhamel et al. (2010) propose a GRASP that uses an evolutionary local search within two solution spaces.

2.3.3 Multi-Depot VRPs

Matheuristics Subramanian et al. (2013) develop a matheuristic based on a set partitioning formulation and an iterated local search (ILS) for a large class of VRPs, including the MDVRP. The MIP solver and the ILS cooperate while solving the set partitioning problem.

Neighborhood-based Metaheuristics The concept of tabu search has also been frequently applied to the MDVRP. Early approaches are from Renaud et al. (1996) and Cordeau et al. (1997). Renaud et al. (1996) first construct an initial solution similar to the approach of Chao et al. (1993), except that they use a petal heuristic for solving the VRPs instead of the savings algorithm. Second, the initial solution is improved by applying a tabu search. The tabu search of Cordeau et al. (1997) is developed for solving the PVRP. They show that the periodic traveling salesman problem and the MDVRP can be seen as special cases of the PVRP and therefore readily solved with the same tabu search. More recently Escobar et al. (2014b) adapt the hybrid granular tabu search from Escobar et al. (2013) to solve the MDVRP. Cordeau and Maischberger (2012) implement a parallel iterated tabu search for solving the VRP, the PVRP, MDVRP, and the site-dependent VRP. The heuristic combines ILS with tabu search. The authors show that the implementation can be readily implemented in parallel. Sadati et al. (2021) implement a granular tabu search combined with a VNS for solving a class of MDVRPs. The approach uses a shaking phase with a large set of neighborhood structures. As another neighborhood-based approach, Pisinger and Ropke (2007) use the ALNS for solving five different VRP variants, including the MDVRP. All problems are transformed to a pickup and delivery problem and solved with the ALNS presented in Ropke and Pisinger (2006). The ALNS adaptively chooses removal and insertion operators to build new solutions. In contrast to the rather sophisticated metaheuristic approaches described above, Arnold and Sörensen (2019) propose a knowledge-guided local search with few efficiently pruned local search operators. The local search uses only a simple perturbation of the cost matrix to escape local optima.

Population-based Metaheuristics The ACO of Yu et al. (2011) uses an ant-weight strategy and mutation operators. The approach adds a virtual central depot as the origin to make the MDVRP similar to a classical VRP. Luo and Chen (2014) present a nature-inspired algorithm, which can be classified as a particle swarm optimization algorithm (PSO). The HGSADC from Vidal et al. (2012) is also able to solve the multi-depot periodic VRP, which includes the MDVRP and PVRP as special cases. The metaheuristic uses two populations of individuals, one for feasible and one for infeasible solutions. Individuals are educated, i.e., a local search is applied on offsprings. The diversity of the population is maintained by evaluating the fitness from a cost perspective and a distance measure to every other individual. Vidal et al. (2014a) combine sequence-based moves with an optimal choice of vehicle, depot and of the first customer to be visited in the route. They present a dynamic programming approach for the evaluation of these neighborhoods. The authors integrate this approach into an ILS and into the HGSADC based on Vidal et al. (2012), demonstrating the value of the proposed concept.

2.3.4 Summary

Numerous heuristic approaches can be found in the literature for the interrelated problem settings of 2E-VRP, LRP and MDVRP. Mostly, they are specifically developed for a single problem class without considering the related problem settings. We give an overview of solution approaches suggested and their respective components applied in Table 2.1. As already mentioned, we classify

the approaches as matheuristic or metaheuristic and denote the respective search space used in a metaheuristic, either neighborhood-based, population-based or both, i.e., hybrid search space. In addition, Table 2.1 denotes various heuristic components that are used in the respective solution approaches. The latter classification scheme is similar to Vidal et al. (2013a).

- **Local improvement:** Solutions are improved by local search procedures, e.g., 2-opt.
- **SA:** Simulated annealing accepts deteriorating solutions with a probability controlled by a gradually decreasing temperature parameter.
- **Tabu search:** Some elements/moves are set tabu for a certain period.
- **Granularity:** The arc set is reduced, e.g., a certain percentage of the longest arcs are excluded.
- **Indirect representations:** The solution is not directly represented, but must be extracted from the indirect representation by applying a decoding procedure, e.g., by applying a split procedure on a giant tour representation.
- **Infeasible solutions:** During the search, infeasible solutions may occur (e.g., if the capacity of vehicles is violated) and be accepted as an incumbent solution. Infeasible solutions are usually penalized by a cost factor or kept in a different population.
- **Diversity management:** The diversity of a population is managed by introducing a distance measure between individuals and using this measure for selection purposes.
- **Parameter adaptation:** Parameters are adapted during the search, e.g., weights for operators (ALNS) or the penalty factor in the presence of infeasible solutions.

Table 2.1 shows that hybrid approaches and also approaches that include multiple components were rarely studied. Almost all publications present results and runtimes on established benchmark instances that we detail in Section 2.5.1. This facilitates a detailed comparison of the performance of these approaches in Section 2.5.3, where we determine state-of-the-art heuristics for benchmarking the HALNS. The HALNS uses a unique mix of components that has not been considered before. We evaluate the impact of single HALNS components in Section 2.5.4.

Table 2.1: Classification of heuristics solving 2E-VRP, LRP, or MDVRP

Approach/Authors	Classification														Components							
	Class		Metaheuristic			Neighborhood-based					Population-based											
	Matheuristic	Metaheuristic	Neighborhood-based	Population-based	Hybrid	GRASP	VNS	LNS	ALNS	ILS	GA	Path Relinking	ACO, PSO	Local Improvement	SA	Tabu Search	Granularity	Infeasible solutions	Indirect representations	Diversity management	Parameter adaptation
2E-VRP																						
Yu et al. (2021)		x	x						x					x			x				x
Mühlbauer and Fontaine (2021)		x			x			x						x	x					x	
Enthoven et al. (2020)		x	x						x					x	x						x
Jie et al. (2019)	x		x						x						x			x			x
Amarouche et al. (2018)	x		x					x						x							
Wang et al. (2017)	x		x				x							x					x		
Breunig et al. (2016)		x	x					x						x			x				
Hemmelmayr et al. (2012)		x	x						x					x				x			x
LRP																						
Arnold and Sörensen (2021)		x	x											x			x				
Schneider and Löffler (2019)		x	x											x		x	x	x		x	x
Lopes et al. (2016)		x		x							x			x							
Escobar et al. (2014a)		x	x				x							x		x	x	x			x
Ting and Chen (2013)		x		x									x	x							
Escobar et al. (2013)		x	x											x	x	x	x	x			x
Contardo et al. (2013)	x		x			x								x							
Hemmelmayr et al. (2012)		x	x						x					x				x			x
Yu et al. (2010)		x	x											x	x			x	x		
Duhamel et al. (2010)		x			x	x				x				x		x			x		
Prins et al. (2007)	x		x											x		x	x	x			x
Prins et al. (2006a)		x		x							x			x					x	x	
Prins et al. (2006b)		x			x	x						x		x						x	
MDVRP																						
Sadati et al. (2021)		x	x				x							x		x	x	x			
Arnold and Sörensen (2019)		x	x											x			x				
Luo and Chen (2014)		x		x									x	x					x	x	
Vidal et al. (2014a)		x		x							x			x			x	x	x	x	x
Escobar et al. (2014b)		x	x											x	x	x	x	x			x
Subramanian et al. (2013)	x		x				x			x				x							
Vidal et al. (2012)		x		x							x			x			x	x	x	x	x
Cordeau and Maischberger (2012)		x	x							x				x		x		x			x
Yu et al. (2011)		x		x									x	x					x		
Pisinger and Ropke (2007)		x	x						x						x						
Sum	6	25	21	7	3	3	4	3	6	3	4	1	3	29	7	8	12	13	8	7	13
This article		x			x				x		x			x	x	x		x		x	x

2.4 Hybrid Adaptive Large Neighborhood Search for VRPs with Depot Location Decisions

Despite the existence of numerous specialized approaches for solving the 2E-VRP, LRP and MD-VRP, there is no method proven to be capable of solving all three problem classes equally efficiently and effectively. We develop a new solution approach, the hybrid adaptive large neighborhood search (HALNS) based on the solution framework proposed by Voigt et al. (2021) that can be used for all three problem classes. Algorithm 3 depicts the general scheme of the HALNS. The HALNS uses multiple solutions, i.e., individuals, generated by an ALNS, called population P, and executes crossover steps within that population. The initial population consists of n^{P} individuals, each generated by executing an ALNS run (lines 1-3 of Alg. 3). In further generations, the ALNS is used to crossover and educate individuals (line 6-9 of Alg. 3). Surviving individuals are chosen according to solution quality and contribution to the diversity of the population (line 10 of Alg. 3). Depot locations can be set tabu for the next generation if they are not used within at least one of the individuals or selected by chance (line 11 of Alg. 3). Figure 2.4 additionally illustrates such a single HALNS generation. The procedure ends when gen^{stop} generations without finding a new best solution have been executed (line 4 of Alg. 3) or the maximum number of generations $gen^{\max}$ is reached. The following sections explain the components of the HALNS in more detail and show the problem-specific adaptations.

Algorithm 3: Hybrid adaptive large neighborhood search

```
1  while |P| < n^P do                                          // Initial population (2.4.1)
2  |   s ← ALNS()
3  |   P ← P ∪ {s}
4  while gens without Improvement < gen^stop ∧ gens < gen^max do   // GA generations
5  |   ŝ ← DetermineBestSolution(P)
6  |   while i < n^P do                                        // Crossover and education phase (2.4.2)
7  |   |   s ← P[i]
8  |   |   s ← ALNS(s, ŝ)
9  |   |   P ← P ∪ {s}
10 |   P ← DiversityManagement(P)                              // Select survivors and manage diversity (2.4.3)
11 |   SetDepotLocationsTabu(P)                                // Set depot locations tabu for next generation (2.4.4)
```

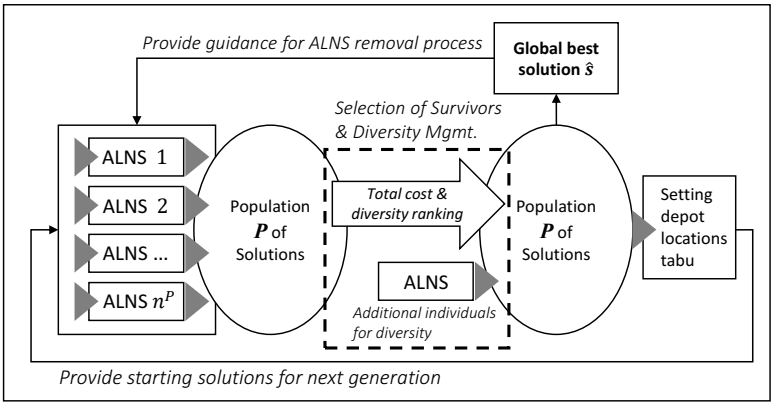


Figure 2.4: Schematic overview of a single HALNS iteration

2.4.1 ALNS Algorithm

Algorithm 4: ALNS algorithm in simulated annealing framework

Input : Starting solution s, global best solution $\hat{s}$
Output: best solution s^*

1 $s^* \leftarrow s$
2 **while** *Iterations without improvement* $< it^{\text{stop}}$ **do**
3 ChooseOperators()
4 $C_R^C \leftarrow \text{getRemovalCandidates}(s, \hat{s})$ // Removal candidates (2.4.1.2)
5 $(s^{\text{new}}, C_R) \leftarrow \text{Remove}(s, C_R^C)$ // Removal operators (2.4.1.3)
6 $C_R \leftarrow \text{sort}(C_R)$ // Insertion order (2.4.1.4)
7 $s^{\text{new}} \leftarrow \text{Insert}(s^{\text{new}}, C_R)$ // Insertion operators (2.4.1.5)
8 $s^{\text{new}} \leftarrow \text{FirstLevelHeuristic}(s^{\text{new}})$ // Determine first-level solution (2.4.1.6)
9 **if** $f(s^{\text{new}}) < f(s^*)$ **then**
10 $s^{\text{new}} \leftarrow \text{LocalSearch}(s^{\text{new}})$ // Local search (2.4.1.7)
11 $(s, s^*) \leftarrow \text{Symmetries}(s^{\text{new}})$ // Account for symmetries (2.4.1.8)
12 **if** $f(s^*) < f(\hat{s})$ **then**
13 $\hat{s} \leftarrow s^*$
14 **else if** $accept(f(s^{\text{new}}), f(s^*), \tau)$ **then**
15 $s \leftarrow \text{Symmetries}(s^{\text{new}})$ // Simulated annealing (2.4.1.9)
16 $\tau \leftarrow \tau \cdot \alpha$
17 UpdateParameters() // Adaptive parameters (2.4.1.10)

Algorithm 4 describes the ALNS algorithm used in the HALNS. First, the ALNS requires a starting solution s. s is either obtained by randomly applying insertion operators until every customer is served (first generation) or simply obtained from the population (further generations). The local best solution s^* is set to s (line 1 of Alg. 4). The ALNS makes use of the global best solution $\hat{s}$. Note that there is no global best solution in the first generation. This means that all solutions are equivalent at the beginning of the first generation, i.e., $\hat{s} = s^* = s$.

The ALNS generates a new solution s^{new} by iteratively removing and then inserting customers. The search proceeds while the number of iterations without finding a new best solution is lower than it^{stop} (line 2 of Alg. 4). At the beginning of every iteration, removal and insertion operators are chosen (line 3 of Alg. 4). The probability of choosing removal and insertion operators depends on the historic performance of operators. Next, a set of customers who are candidates for removal, C_R^C is generated by comparing the current solution s and the global best solution $\hat{s}$ (line 4 of Alg. 4). The basic idea results from the observation that a certain number of edges in a good solution is similar to an even better solution. Customers are therefore to be added to C_R^C if they seem oddly placed in s compared to $\hat{s}$. The removal operator chosen then removes some or all customers from the solution who are included in set C_R^C (line 5 of Alg. 4). Data collected during the search determines the order in which the removed customers C_R are inserted (line 6 of Alg. 4). Next, the chosen insertion operator inserts all customers from the sorted set of previously removed customers C_R into the solution (line 7 of Alg. 4). After inserting all customers, the first-level costs have to be determined via a first-level heuristic (line 8 of Alg. 4).

Whenever a new best solution is found, a local search procedure is used to improve the routes (line 10 of Alg. 4). Before a solution is accepted, it must be checked for symmetries (lines 11 and 15 of Alg. 4)). The ALNS uses simulated annealing to escape local optima (lines 9-15 of Alg. 4). Please note that determination of the initial temperature is instance-specific. At the end of every

iteration the temperature τ is multiplied by the cool rate α (line 16 of Alg. 4) and the parameters are updated (line 17 of Alg. 4).

Before we detail the parts of the ALNS in Subsections 2.4.1.2 - 2.4.1.10, we introduce the information collection via insertion operators.

2.4.1.1 Information Collection

The insertion operators collect data when inserting customers which is then used during the subsequent removal and insertion operations. These are

- the historic penalized costs C_j^{hist},
- the minimal encountered costs $C_j^{\min}$,
- and the number of times customer j has been reinserted, n_j.

C_j^{hist} is increased on every insertion of customer j by $\Delta_j = f(s_1) - f(s_0)$, where s_1 represents the solution after inserting customer j and s_0 the solution before inserting customer j. Thus, C_j^{hist} measures the accumulated increase in costs when inserting customer j. Similarly, $C_j^{\min}$ quantifies the minimum insertion costs of customer j found so far. n_j is simply increased by one if customer j has been reinserted.

2.4.1.2 Determination of Removal Candidates

Before applying one of the removal operators the set of removal candidates is determined by comparing the current solution s with the global best solution $\hat{s}$. We implement two variants for selecting which customers are added to set C_R^C corresponding to the location and the routing decisions considered. The first variant uses the routing decision, more specifically a customer is added to C_R^C if its successor in s differs from its successor in $\hat{s}$. Customers who do not fulfill this criterion may still be added to the set with probability p^{binom}. The second variant compares the location decision of customers. A customer who is served from a different location in s compared to $\hat{s}$ will be added to C_R^C. The same probability-based approach as for the first variant is used. Furthermore, we use a third alternative to foster diversification, where $C_R^C = C$, i.e., there exists no pre-selection of customers. One of the three alternatives is randomly chosen in every iteration. The probability depends on the performance and is adapted during the search (see Section 2.4.1.10).

2.4.1.3 Removal Operators

The maximum number of customers to be removed in one iteration q^{binom} is sampled from a binomial distribution with sample size $|C|$ and probability p^{binom}. The number of customers that are actually removed by one of the removal operators below is expressed by $q^{\text{r}} = \min\{q^{\text{binom}}, |C_R^C|\}$. Thus, removal operators select q^{r} customers from set C_R^C according to the criterion specific to the removal operator used (e.g., randomly or according to costs), and then remove these customers from the current solution. The customers removed are added to set C_R.

Random Removal The operator selects and removes customers randomly.

Historic Cost Removal The operator selects customers in decreasing order of average historic penalized costs, i.e., in decreasing order of $\frac{C_j^{\text{hist}}}{n_j}$.

Worst Cost Removal The operator calculates the change in costs Δ_j if customer j is removed from the current solution. Customers are selected in descending order of $\Delta_j - C_j^{\min}$.

Shaw Removal The operator selects customers similar to each other. The first customer is removed with the *Random Removal Operator*. The following requests are chosen from C_R^C in increasing order of relatedness. The relatedness $Rel(c_1, c_2)$ of two customers c_1 and c_2 is measured by the distance and demand of both customers. Customers with a lower $Rel(c_1, c_2)$ are more related and removed first.

$$Rel(c_1, c_2) = \frac{c_{c_1,c_2}}{\max\limits_{i,j \in N} c_{ij}} + \frac{|d_{c_1} - d_{c_2}|}{\max\limits_{j \in C} d_j} \tag{2.1}$$

Depot Removal The operator randomly selects a depot and removes either all customers who are served via this depot or randomly up to q^{binom} customers. Note that for this and the following operator, we set $C_R^C = C$, i.e., all customers are available for removal. This operator helps to close a depot if only few customers are served via this depot.

Least Efficient Vehicle Removal This operator uses the whole customer set for removal similar to the *Depot Removal Operator*. The operator identifies the least efficient vehicles. The least efficient vehicle has the highest ratio of tour costs compared to the number of customers on the tour, i.e., the highest cost per customer. The operator then removes customers who are served via this vehicle. The operator repeats the process until q^{binom} customers have been removed. If q^{binom} exceeds the number of customers served via the least efficient vehicle, for example, customers from the second least efficient vehicle are removed, and so on, until the total number of customers removed exceeds q^{binom}.

2.4.1.4 Determination of Insertion Order

The set of removed customers, C_R is sorted in descending order of average historic costs $\frac{C_j^{\text{hist}}}{n_j}$. This means that customers with high historic costs are inserted earlier, and there are therefore more options to insert them at a low-cost position. Customers with low average historic costs can be inserted later as they supposedly have more options to be inserted at an appropriate position with low costs.

2.4.1.5 Insertion Operators

After sorting C_R, the customers are inserted using one of the operators described in the following. The operators have to decide which depot the customer is to be served from and at which position in a route the customer is placed. Obviously both decisions influence not only each other but also the decisions taken for customers inserted later. For example, if a depot is opened because a customer is inserted, the customers to be inserted in the following will more likely be assigned to

that depot, instead of a new depot being opened. We therefore use variants of the *Best Insertion Operator* that neglect specific cost components. The total costs consist of costs when opening a depot (called depot costs), vehicle costs when using a previously unused vehicle, and distance costs when traveling from one node to the next. All operators include distance costs. Please note that the depot costs can be determined accurately for the LRP. In contrast, for the 2E-VRP we can only approximate depot costs as it would be extremely time-consuming to solve the first level-routing every time a customer is inserted. The depot (satellite) costs are approximated for the 2E-VRP using the approach of Mühlbauer and Fontaine (2021).

Best Insertion Operator The operator iterates across all depots and vehicles to find the position where the customer can be inserted with lowest total cost, i.e., the sum of depot costs, vehicle costs and distance costs. This operator favors positions in depots and vehicles that have already been opened.

Best Insertion Operator - without Vehicle Costs This operator neglects vehicle costs, so new vehicles may be used when the distance costs are lower in a new vehicle compared to vehicles already in use. When respecting vehicle costs, the difference in distance cost must be at least as high as the cost of using a vehicle, otherwise we have to accept high distance costs in favor of not using a new vehicle. This operator still favors depots that have already been opened.

Best Insertion Operator - without Depot Costs This operator neglects depot costs. A depot may be opened if the distance costs plus vehicle costs at the unopened depot are lower compared to a position within a used/unused vehicle at a depot that is already open. This operator favors the opening of additional depots and therefore helps to investigate further depot configurations.

2.4.1.6 First-Level Heuristic

The first-level problem is solved from scratch after all customers have been inserted in the second level, i.e., once the assignment of customers to satellites is known. The first-level heuristic is inspired by Hemmelmayr et al. (2012) and Mühlbauer and Fontaine (2021). The heuristic works in three steps. 1) If the demand of a satellite exceeds the capacity of the first-level vehicle, the heuristic creates back-and-forth trips until the remaining demand fits into one truck. 2) The heuristic generates an initial solution by inserting all open satellites (with their respective remaining demands) at its best position in a giant tour and then splitting the giant tour. We follow the giant tour as long as the capacity of the vehicle suffices. As soon as the capacity limit is reached we randomly decide with probability p^{split} whether we either split the demand (generating a split satellite) and return to the main depot or return directly to the main depot without the demand of the satellite that caused the capacity to be exceeded. 3) An improvement phase searches for better solutions, stopping when no more improvements can be found. Let r_i and r_j be a pair of first-level routes. The improvement phase uses a relocate and a swap operator to find improvements for all pairs of first-level routes. The relocate operator relocates a satellite in r_i to a different position in the same route. The swap operator swaps two satellites s_1 in r_i and s_2 in r_j. The swap operator

checks whether the capacity of the first-level vehicles suffices. If the capacity suffices on both first-level vehicles, both satellites are swapped. If the capacity only suffices for s_1 on r_j, s_1 is moved to r_j and s_2 is moved to an empty route, and vice versa. Steps 2) and 3) are repeated $10 \cdot n_s^{\text{open}}$, whereas n_s^{open} is the number of satellites in use. Please note that the heuristic is not needed if $n_s^{\text{open}} \leq 2$, as the optimal solution can easily be obtained. We randomly set $p^{\text{split}} = unif(0, 1)$ for each iteration. To speedup the procedure, a hash table keeps track of satellite configurations already examined. The first-level problem consists of only few satellites, so that this simple heuristic finds good solutions within reasonable runtimes.

2.4.1.7 Local Improvement of Routes

The local search procedure is used every time a new best solution is found to further enhance the quality of that solution. The local search procedure consists of simple insertion and swap operators working on the established routes of a solution. The local search maintains the location decision and route assignment of customers, resulting in pure intra-route improvement. We use the following operators, whereas c_1 denotes a customer, c_2 its successor and c a customer on the same route. The local search iterates in that order of operators across each customer c_1 and c in that route until no more improvement can be found. The first improving move is accepted.

- Insertion 1: Remove c_1, then insert it after c.
- Insertion 2a: Remove c_1 and c_2, then insert c_1 and c_2 after c.
- Insertion 2b: Remove c_1 and c_2, then insert c_2 and c_1 after c.
- Swap 1: Swap c_1 with c.
- Swap 2: Swap c_1 and c_2 with c.

2.4.1.8 Accounting for Symmetries

After inserting every customer and executing the local search in case of a new best solution, the routes have to be aligned to account for symmetrical solutions. This is important as the determination of removal candidates (Section 2.4.1.2) and the diversity measure (Section 2.4.3) both rely on the successor relation of nodes. The direction of the route does not alter the cost, but a different direction is represented by a completely different successor vector. Same solutions would be identified as completely different. Consider an example with depot 0, customers 1 and 2 and two solutions A and B, solution A with route (0-1-2-0) and solution B with route (0-2-1-0). We can simply change the direction of the solution B and see that it is the same as solution A. The successor vector of the two customers 1 and 2 of solution A, however, is (2,0), and for solution B (0,1). A simple way to remove this kind of symmetrical solutions is to allow only routes where the first customer has a lower index than the last customer on that route (or the same index if there is only one customer on the route). This means the route of solution A is aligned correctly, whereas the route of solution B has to be inverted.

2.4.1.9 Simulated Annealing

Simulated annealing is used as an acceptance criterion. Deteriorating solutions are accepted with a probability depending on the difference in costs of the candidate solution $f(s^{\text{new}})$, the cost of the best solution obtained during the run of the ALNS $f(s^*)$ and the current temperature τ. A worse solution is accepted if $e^{\frac{-(f(s^{\text{new}})-f(s^*))}{\tau}} > unif(0,1)$.

The temperature τ is determined for every instance with $\tau = -\frac{\overline{\Delta E}}{ln(\chi_0)}$ using the formula from Johnson et al. (1989) at the beginning of each ALNS run. $\overline{\Delta E}$ estimates the cost increase of strictly positive transitions and χ_0 expresses the probability of accepting a deteriorating solution. We execute n_0 iterations of the ALNS in order to generate the transitions. The temperature is reduced by α after each iteration.

2.4.1.10 Adaptive Parameters

The HALNS adaptively changes three kinds of parameters, which are (1) the probability of choosing operators (determining the removal candidates, removal and insertion operators), (2) the probability of the binomial distribution p^{binom} determining the number of requests to be removed, and (3) the weight ω for penalties.

Operators The probability of an operator depends on the historic performance, expressed by a score that is increased by either σ_1, σ_2 or σ_3 (Ropke and Pisinger 2006). If a new best solution is found, the score is increased by σ_1. If a previously unknown solution with lower costs than the current solution is found, the score is increased by σ_2. If the solution has higher costs but is accepted through the simulated annealing procedure, the score is increased by σ_3.

Probability of Binomial Distribution The probability of binomial distribution is adapted in a similar manner with $p^{\text{binom}} = \gamma \cdot \overline{p^{\text{binom}}} + (1-\gamma) \cdot p^{\text{binom}}$. Parameter γ denotes the reaction factor and $\overline{p^{\text{binom}}}$ the average of the share of actual removed customers weighted by σ_1, σ_2 or σ_3. If in iteration A, for example, 20% of customers were removed and a new best solution is generated, and in iteration B 40% of customer were removed and the solution was accepted via simulated annealing, then $\overline{p^{\text{binom}}} = \frac{\sigma_1 \cdot 0.2 + \sigma_3 \cdot 0.4}{\sigma_1 + \sigma_3}$.

Weight for Penalties The weight for penalties is adapted after each generation depending on the number of infeasible solutions in the population. If the number of infeasible solutions within the generation is smaller than $\frac{n_p}{3}$ the weight is divided by five or if the number of infeasible solutions exceeds $\frac{2n_p}{3}$ the weight is multiplied by five.

2.4.2 Crossover and Education Phase

The crossover and education phase tries to combine two individuals by using the ALNS described previously instead of using an explicit crossover operator, which usually tries to combine individuals by maintaining good parts of these solutions. The ALNS uses solutions of the population one by one as a starting solution and the global best solution to guide the removal process. Customers are

removed if they differ in both solutions with respect to location assignment or successor assignment, as described in Section 2.4.1.2. This means the probability of removing customers who are already well placed decreases.

2.4.3 Selection of Survivors and Diversity Management

The selection of individuals is based on total costs and a diversity measure. We assign a cost rank $Rank_i^{\text{Costs}}$ to each individual i. Similarly, a diversity rank is assigned to each individual $Rank_i^{\text{Diversity}}$. The cost rank is determined from sorting the population in increasing order of total costs. The diversity rank is determined by sorting the population based on the hamming distance to all other individuals. The hamming distance is calculated by comparing the successor of nodes, $Succ^{\text{ind}}$ against the $Succ^{\text{i}}$ of all other individuals $i \in P$.

$$Hamming^{\text{ind}} = \sum_{i \in P} \sum_{j \in C} Succ_j^{\text{ind}} \neq Succ_j^{\text{i}} \tag{2.2}$$

Finally, the population is ordered in increasing order of the overall rank ($Rank_i^{\text{Overall}} = Rank_i^{\text{Costs}} + Rank_i^{\text{Diversity}}$).

Individuals with the n^{P} lowest overall ranks are used in the next generation. The population may grow until $|P| = 4n^{\text{P}}$. After reaching this point, excess individuals with the highest overall rank are removed from the population. New individuals are generated to foster diversification after each fifth generation.

2.4.4 Setting Depot Locations Tabu

After every generation, depot locations may be set tabu for the following generation. The procedure sets all depot locations tabu and then iterates across all individuals to check which depot locations are in use for that specific individual. As soon as a depot location is used for at least one individual, its use is permitted in the next generation. To further explore depot location configurations, depot locations may be set as non-tabu with a 30% probability. Similarly, depot locations may be set tabu with a 10% probability as long as the remaining total capacity suffices to accommodate the demand. This search strategy significantly reduces runtime and intensifies the search around promising depot location configurations.

2.5 Numerical Experiments

In this section we introduce the instances used and our experimental setting (Section 2.5.1), state the parameter tuning (Section 2.5.2), evaluate the performance of the HALNS on benchmark instances against approaches from literature (Section 2.5.3), and lastly evaluate the various components of the HALNS (Section 2.5.4).

2.5.1 Instances and Experimental Setting

Table 2.2 shows the instances used for the problem classes 2E-VRP, LRP and MDVRP and the respective sources for obtaining the instances. Please note that we use the MDVRP instances with a tight fleet size limit as given in `https://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/`. As pointed out by Sadati et al. (2021) there also exists a variant with a higher fleet size limit corresponding to the original values given by Cordeau et al. (1997).

Table 2.2: Instances

Problem	Instances	Exemplary Source
2E-VRP ($n = 207$)	Set 2a, 2b, 2c	Breunig et al. (2016) `http://www.univie.ac.at/prolog/research/TwoEVRP`
	Set 3a, 3b, 3c	
	Set 4a, 4b	
	Set 5	
	Set 6a	
LRP ($n = 79$)	Prodhon	Prins et al. (2004) `http://prodhonc.free.fr/Instances/instances_us.htm`
	Tuzun	Tuzun and Burke (1999) `http://prodhonc.free.fr/Instances/instances_us.htm`
	Barreto	Barreto et al. (2007) `http://sweet.ua.pt/sbarreto/`
MDVRP ($n = 33$)	Cordeau	Cordeau et al. (1997) `https://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/`

The HALNS is coded in C++. Experiments in Section 2.5.2 and 2.5.4 are conducted on an AMD Ryzen 7 2700X CPU with eight cores and 16 GB of RAM and in Section 2.5.3 on an AMD Ryzen 9 3900X CPU with twelve cores and 32 GB of RAM. All experiments use one thread. We use the same parameter setting for all experiments and problem classes after extensive preliminary parameter testing (see Section 2.5.2). We only vary the maximum number of generations, $gen^{\max}$, which is increased from $gen^{\max} = 10$ in Section 2.5.2 to $gen^{\max} = 100$ in Section 2.5.3 and $gen^{\max} = 50$ in Section 2.5.4. All parameters can be found in B.1.

To make the runtimes comparable we standardize runtimes by the passmark single thread rating of the CPUs used. The AMD Ryzen 9 3900X has a passmark single thread rating of 2731, while the AMD Ryzen 7 2700X has a rating of 2439. We can thus expect a runtime of the HALNS of 89.3% when running on 3900X, instead of running on 2700X. This approximation is reasonably accurate, as we show in B.2. In the following, all runtimes are standardized as if run on the AMD Ryzen 9 3900X. The respective passmark single thread ratings for all CPUs used in the relevant benchmark literature can be found in Table B.3.

2.5.2 Parameter Tuning

We tune a set of parameters that showed a relevant impact in the numerical study of Voigt et al. (2021) (see Table 2.3). These parameters are quantified as follows. First, we randomly select 30 instances from the total set of 2E-VRP instances. Starting from the initial parameter setting as given in Voigt et al. (2021), we alter the parameters one by one in the sequence and the range listed in Table 2.3. The initial values are marked with a superscript i.

Table 2.3: Parameter tuning for HALNS on 2E-VRP instances

Parameter		Values	Chosen
n^{P}	size of the initial population	4, 12, 24, 36, 48^i	12
it^{stop}	number of ALNS iterations without improvement	1000, 10000, 20000, 30000^i, 40000	10,000
p^{binom}	probability for binomial distribution drawn at the beginning of every ALNS run	$unif(0.12, 0.24)^i$, $unif(0.10, 0.35)$, $unif(0.24, 0.48)$	$unif(0.10, 0.35)$
α	cool rate in SA	0.9991, 0.9993, 0.9995, 0.9997, 0.9999^i	0.9997
χ_0	acceptance probability in SA	0.01, 0.10, 0.15, 0.25^i, 0.35	0.10

We run the HALNS five times for every parameter setting on the subset selected and set the number of generations to $gen^{\max} = 10$ to reduce the computational effort. We plot the average search trajectory, i.e., the average gap achieved after each generation (with its corresponding average runtime) to assess the trade-off between runtime and solution quality. Lastly, upon visually inspection, we chose the value with the lowest curve, where the HALNS produces the same results within shorter runtime or better results within the same runtime. In other words, a parameter setting dominates, if the area under the respective search trajectory is the smallest.

As an example, Figure 2.5 illustrates the procedure for parameter α. The figure shows the search trajectories for the five tested values. Setting $\alpha = 0.9997$ yields the best result for longer runtimes. For shorter runtimes there is no clear best parameter. Similar results are found for the other parameters. The HALNS is therefore robust and not very dependent on parameters as long as the chosen values are within a reasonable range.

2.5.3 Benchmarks

We evaluate the performance of approaches on benchmark instances and average the results over all benchmark instances of one problem class. In the following tables, the first column shows the *Authors*, while the second column shows the *Approach* used. The *Best Gap* column represents the gap of the best run compared to the (previous) BKS, whereas column *Avg. Gap* represents the average gap over n runs. Most authors use five runs for 2E-VRP and LRP ($n = 5$), and ten ($n = 10$) for MDVRP. As a result, the HALNS is run five and ten times. The column $\bar{T}_{\mathrm{scaled}}$ *[s]* represents the average standardized runtimes across n runs in seconds, as if run on the AMD Ryzen 9 3900X using the passmark single thread ratings, as described previously. The last column n represents the number of runs. The approaches are sorted according to their *Best Gap* performance. Furthermore, Figure 2.6-2.8 illustrate the average gaps for standardized runtimes to facilitate an easier comparison of approaches. The graphs of the HALNS are generated by varying the stopping

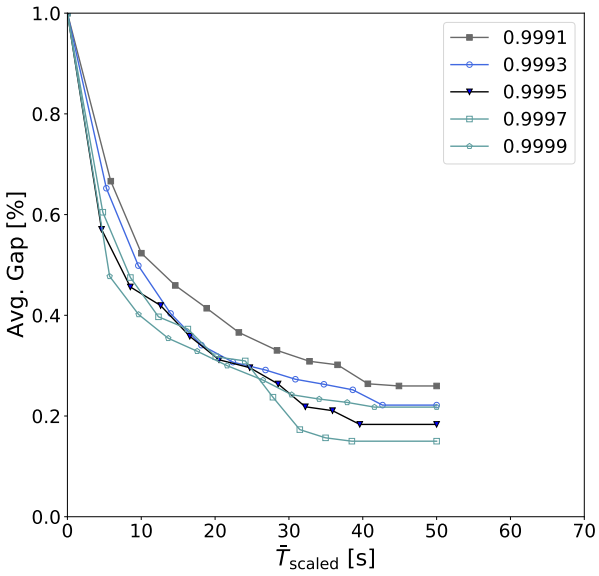


Figure 2.5: Search trajectories for different α on subset of 2E-VRP instances

condition, $gen^{\text{stop}} = [1, \ldots, 50]$. In contrast to these figures, Table 2.4-2.6 show only the results when $gen^{\text{stop}} = 2,\ 10,$ or 50. For example, *HALNS 2* means that the HALNS stops after two generations without improvement, i.e., $gen^{\text{stop}} = 2$. Please note that the figures show only approaches with an average gap lower than 0.5%.

2.5.3.1 2E-VRP Benchmark

Table 2.4 shows the results for the 2E-VRP. Figure 2.6 additionally illustrates the trade-off between solution quality and computation time. The HALNS forms the efficient boundary, i.e., the HALNS dominates all other approaches both in terms of solution quality and computation time. *The HALNS 2* already generates good results with a best gap of 0.05% and an average gap of 0.16% within a short runtime of 27s. Only the approaches of Wang et al. (2017) and Amarouche et al. (2018) achieve a similar best gap, and while their average gap is lower, $\bar{T}_{\text{scaled}}$ is considerably higher. The HALNS's best gap can be further reduced to 0.03% by increasing gen^{stop} to 10. Additional generations do not further improve the best gap. However, the average gap decreases to 0.04% for the *HALNS 50* and approaches the best gap, showing that additional generations stabilize results. The same pattern can also be observed for the LRP and MDVRP. The HALNS outperforms all pure ALNS implementations, showing the value of hybridizing the ALNS. Note that several approaches omit some of the benchmark sets. We therefore additionally give the HALNS's average gap for the considered subsets. The detailed results for the *HALNS 50* can be found in B.3, Table B.4, where it is shown that the *HALNS 50* finds 190 of 207 BKS.

Table 2.4: Heuristics for 2E-VRP

Authors	Approach	Best Gap	Avg. Gap	$\bar{T}_{\text{scaled}}$ [s]	n
Voigt et al.	**HALNS 50**	**0.03%**	**0.04%**	**286**	**5**
Voigt et al.	**HALNS 10**	**0.03%**	**0.07%**	**83**	**5**
Wang et al. (2017)	VNS + Math	0.04%	0.08%	126	5
Amarouche et al. (2018)	LNS + Math	0.05%	0.10%	65	5
Voigt et al.	**HALNS 2**	**0.05%**	**0.16%**	**27**	**5**
Mühlbauer and Fontaine (2021)	Hybrid LNS	0.07%	0.19%	49	5
Breunig et al. (2016)	LNS	0.09%	0.17%	118	5
Yu et al. (2021)[1]	ALNS	0.16%	0.32%	156	5
Hemmelmayr et al. (2012)[2]	ALNS	0.24%	0.48%	34	5
Jie et al. (2019)[3]	ALNS + Math	0.60%	0.82%	552	5
Enthoven et al. (2020)[4]	ALNS	0.61%	0.97%	63	5

[1] Set 6 not included. Avg. gap of *HALNS 50* on the subset: 0.04%.
[2] Sets 2c, 3c, 4a and 6 not included. Avg. gap of *HALNS 50* on the subset: 0.06%.
[3] No CPU indicated. We assume a passmark score of 1500, similar to publications from the same year.
[4] Sets 4a, 4b and 6 not included. Avg. gap of *HALNS 50* on the subset: 0.08%.

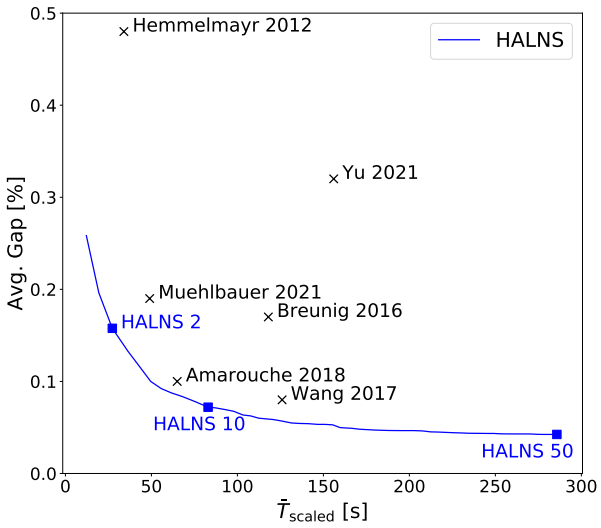


Figure 2.6: Performance chart 2E-VRP

2.5.3.2 LRP Benchmark

Table 2.5 and Figure 2.7 show the results for the LRP. Again, the *HALNS 2* already yields a good best gap of 0.16% at a low $\bar{T}_{\text{scaled}}$ of 40s. Both best and average gap can be significantly improved by additional generations, up to 0.02% and 0.07%, respectively, which are the best results of all approaches considered. Again, additional generations reduce the difference between best and average gap and make the approach more robust. In total, we find 62 of 79 BKS (see B.3, Table B.5), including three newly found BKS, which are shown in detail in B.3, Tables B.7 - B.9. However, as easily seen in Figure 2.7, the approach of Schneider and Löffler (2019) with its different configurations achieves similar results as the HALNS. It dominates the HALNS for shorter runtimes. It is reasonable to assume that the approach of Schneider and Löffler (2019) shows a similar behavior as HALNS in the brackets between the given values, but for longer runtimes, the HALNS yields slightly better results.

Table 2.5: Heuristics for LRP

Authors	Approach	Best Gap	Avg. Gap	$\bar{T}_{\text{scaled}}$ [s]	n
Voigt et al.	**HALNS 50**	**0.02%**	**0.07%**	**389**	**5**
Schneider and Löffler (2019)	Tree + GTS (Quality)	0.02%	0.08%	387	5
Schneider and Löffler (2019)	Tree + GTS (Basic)	0.04%	0.16%	109	5
Voigt et al.	**HALNS 10**	**0.05%**	**0.15%**	**121**	**5**
Voigt et al.	**HALNS 2**	**0.16%**	**0.37%**	**40**	**5**
Schneider and Löffler (2019)	Tree + GTS (Speed)	0.18%	0.35%	23	5
Arnold and Sörensen (2021)	Progressive Filtering	0.21%	0.21%	97	1
Contardo et al. (2013)	GRASP + Math	0.22%	0.53%	741	10
Hemmelmayr et al. (2012)	ALNS	0.45%	0.79%	94	5
Lopes et al. (2016)	GA	0.50%	0.77%	209	10
Escobar et al. (2014a)	GTS + VNS	0.66%	0.66%	40	1
Ting and Chen (2013)	ACO	0.79%	0.79%	22	10
Escobar et al. (2013)	GTS	0.93%	0.93%	77	1
Yu et al. (2010)	SA	0.96%	0.96%	237	1
Duhamel et al. (2010)	GRASP + ELS	1.09%	1.09%	182	5
Prins et al. (2007)	GTS + Math	1.47%	1.47%	3	1
Prins et al. (2006a)	GA	1.79%	1.79%	18	1
Prins et al. (2006b)	GRASP + Path Relinking	3.31%	3.31%	15	1

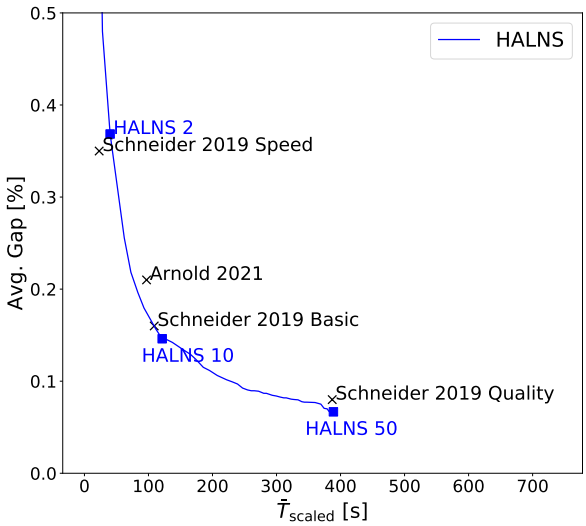


Figure 2.7: Performance chart LRP

2.5.3.3 MDVRP Benchmark

Table 2.6 and Figure 2.8 show the results for the MDVRP, detailed results are to be found in B.3, Table B.6. Overall, we find 27 of 33 BKS. Several approaches from literature deliver similar best gaps as the HALNS. The lowest best gap is achieved by the genetic algorithm of Vidal et al. (2012). The HALNS needs considerably more runtime to reach similar results compared to their approach and still produces a slightly worse best gap. However, the average gap is better. In contrast to that, Luo and Chen (2014) produce a slightly worse average gap than the HALNS, but a similar best gap at even lower runtimes than Vidal et al. (2012). Furthermore, the approach of Subramanian et al. (2013) delivers results similar to those of the HALNS, but within shorter runtimes. This slightly worse performance of the HALNS compared to the previous results on 2E-VRP and LRP

instances can be expected since the MDVRP differs most from the 2E-VRP. There is effectively only one insertion operator for the MDVRP. Furthermore, we expect that a stronger local search may be beneficial for solving the MDVRP, as used by Vidal et al. (2012).

Table 2.6: Heuristics for MDVRP

Authors	**Approach**	**Best Gap**	**Avg. Gap**	$\bar{T}_{\text{scaled}}$ [s]	n
Vidal et al. (2012)	GA	0.00%	0.09%	52	10
Voigt et al.	**HALNS 50**	**0.01%**	**0.05%**	**695**	**10**
Luo and Chen (2014)	PSO	0.01%	0.3%	17	10
Subramanian et al. (2013)	ILS + Math	0.02%	0.08%	364	10
Voigt et al.	**HALNS 10**	**0.03%**	**0.12%**	**187**	**10**
Voigt et al.	**HALNS 2**	**0.06%**	**0.22%**	**63**	**10**
Arnold and Sörensen (2019)[1]	Guided Local Search	0.09%	0.09%	34	1
Pisinger and Ropke (2007)	ALNS	0.11%	0.44%	49	10
Sadati et al. (2021)	TS + VNS	0.2%	0.3%	297	25
Cordeau and Maischberger (2012)	TS + ILS	0.11%	0.29%	476	10
Escobar et al. (2014b)	GTS	0.19%	0.19%	41	1
Renaud et al. (1996)	TS	0.92%	0.92%	42	1
Cordeau et al. (1997)	TS	1.06%	1.06%	41	1
Chao et al. (1993)[1]	no classification	2.74%	2.74%	37	1

[1] *pr* instances from Cordeau et al. (1997) not included. Avg. gap of *HALNS 50* on the subset: 0.03%.

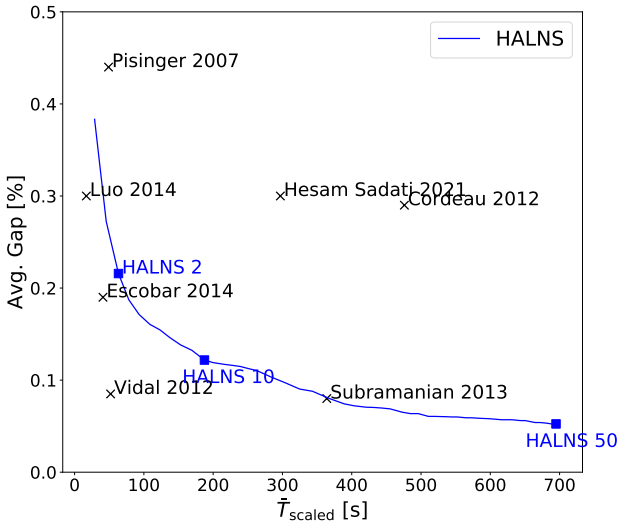


Figure 2.8: Performance chart MDVRP

2.5.4 Analysis of Algorithm Components

The HALNS combines a number of different heuristic components. The question therefore arises what influence each of these components has on the overall performance of the HALNS. We individually and independently disable the features and components listed below (see also Table 2.1) and compare the resulting performance to the performance of the full version of HALNS.

Hybridization: We disable the hybrid approach, i.e., neglecting the extended search of a population-based approach. In this case, the HALNS is reduced to a multi-start ALNS. The ALNS repeats $n_{\mathrm{P}} \cdot gen^{\max}$ times without using information from previous generations (line 8 of Alg. 3). The procedure disables the diversity management (line 10 of Alg. 3). The weights for penalties for infeasible solutions are not adapted. The depot locations are set tabu without considering information on depots that have already been used (line 11 of Alg. 3), which means the depot locations are just randomly set tabu.

Local improvement: We disable the additional local search procedure in case a new best solution is found during an ALNS run (line 10 of Alg. 4).

Simulated annealing: We exclude the diversification function within the ALNS. The ALNS then accepts only improved solutions (line 15 of Alg. 4).

Tabu search: We set depot locations within the search space never tabu (line 11 of Alg. 3).

Infeasible solutions: We increase the weight for penalties to a sufficiently high number, such that infeasible solutions are not accepted if the ALNS has already found a feasible solution.

Diversity management: We limit the diversity of the search space so that the surviving individuals are only selected according to their cost rank, $Rank_i^{\mathrm{Costs}}$. Additionally, new individuals are not generated after each fifth generation (line 10 of Alg. 3).

Parameter adaptation: We disable the dynamic adaptation process of search space parameters. The probabilities for the choice of operators, the probability for the binomial distribution, p^{binom}, and the weight of penalties, ω, remain unchanged during the search (line 17 of Alg. 4).

Table 2.7 presents the results for all 2E-VRP instances when applying the HALNS algorithm with a certain configuration and when applying the HALNS disabling one mentioned component at a time.

Column *Best Gap* respective *Avg. Gap* shows the best and average gap from five runs. Column $\bar{T}_{\mathrm{scaled}}$ *[s]* shows the average standardized runtime as if run on the AMD Ryzen 9 3900X. The first section of the table (HALNS) shows the results of the HALNS with $gen^{\max} = 50$, $n^{\mathrm{P}} = 12$ (full), $gen^{\max} = 1$, $n^{\mathrm{P}} = 12$ (one generation) and $gen^{\max} = 1$, $n^{\mathrm{P}} = 1$ (one individual). The experiment with only one individual demonstrates that the ALNS on its own already generates reasonably good results with an average gap of 1.27% in under 1s of runtime. Executing just one generation with 12 individuals improves the average gap to 0.44%. The execution of the complete HALNS with

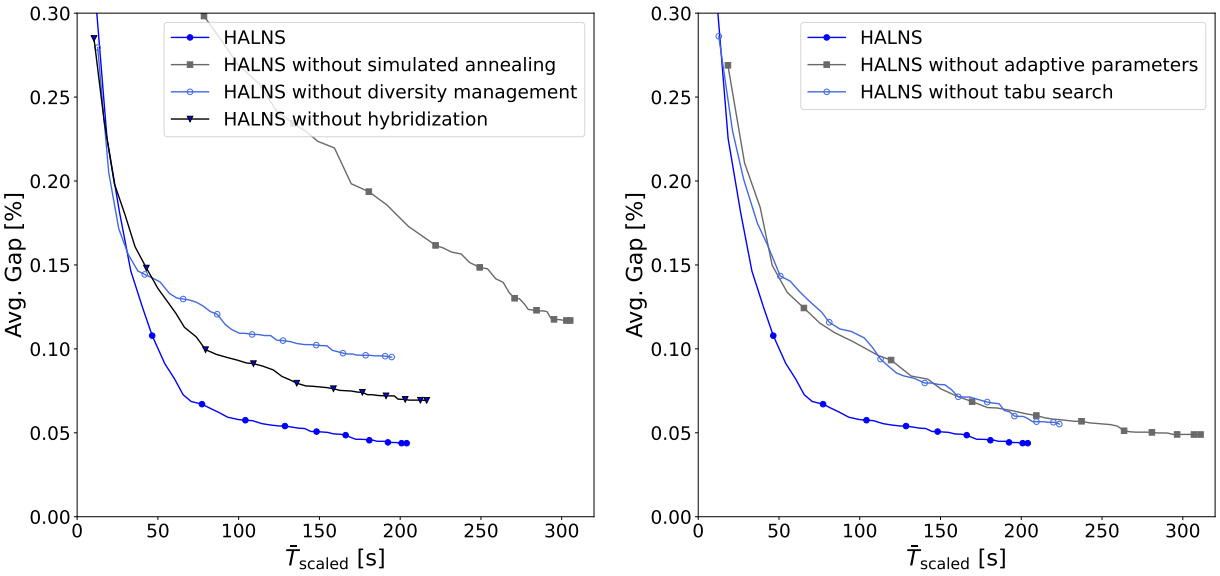
Table 2.7: Analysis of HALNS solutions of 2E-VRP when disabling one component at a time

Configuration	Best Gap	Avg. Gap	$\bar{T}_{\text{scaled}}$ [s]
HALNS			
$gen^{\max} = 50$, $n^{\text{P}} = 12$ (full)	0.03%	0.04%	204
$gen^{\max} = 1$, $n^{\text{P}} = 12$ (one generation)	0.21%	0.44%	5
$gen^{\max} = 1$, $n^{\text{P}} = 1$ (one individual)	0.47%	1.27%	1
High-impact components			
Simulated annealing	0.06%	0.12%	305
Diversity management	0.04%	0.10%	195
Hybridization	0.04%	0.07%	216
Medium-impact components			
Tabu search	0.03%	0.06%	224
Parameter adaptation	0.03%	0.05%	311
Low-impact components			
Local improvement	0.02%	0.04%	204
Infeasible solutions	0.02%	0.04%	205

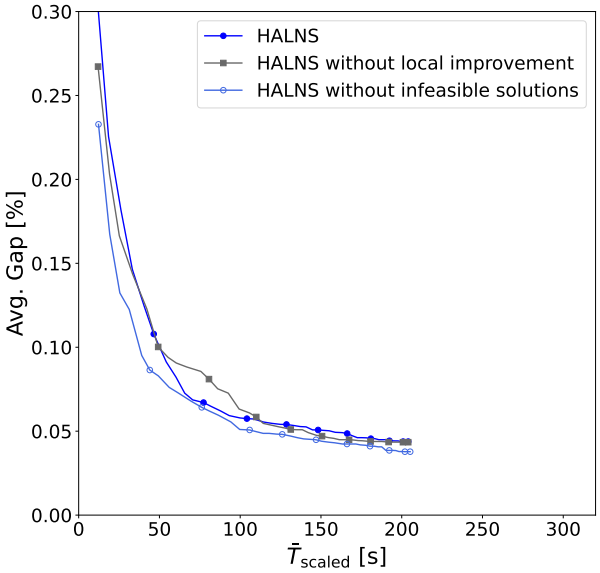
50 generations further stabilizes the results to an average gap of 0.04%, but increases the average runtime to 204s.

The following experiments apply the HALNS in the full configuration ($gen^{\max} = 50$, $n^{\text{P}} = 12$), and individually and independently disable one component. The second, third and fourth section of Table 2.7 (high-/medium-/low-impact components) show the respective results. The best and average gap deteriorates when high impact components are disabled compared to the full configuration. *Simulated annealing*, *diversity management* and *hybridization* highly impact the solution quality. Medium-impact components do not deteriorate the best gap when disabled, but deteriorate the average gap. *Tabu search* and *parameter adaptation* have a medium impact on the solution quality. Lastly, disabling *local improvement* and the absence of *infeasible solutions* do not affect the average gap and even slightly improve the best gap, while the effect on runtime is negligible. Note that multiple components not only improve solution quality, but also reduce runtime, e.g., *simulated annealing* and *parameter adaptation*.

Figures 2.9a, 2.9b and 2.9c show the solution gap as a function of scaled runtime for high-, medium-, and low-impact components, respectively. The graphs are generated by varying the stopping condition, just as in the previous section. The graphs are largely consistent across runtimes, i.e., a disabled component deteriorates the solution quality for high- and medium-impact components at all runtimes tested. The component *local improvement* (see Figure 2.9c), however, has a slightly different pattern. It seems to improve the performance for medium runtimes, but has no effect at long runtimes. The local search only improves routes (see Section 2.4.1.7). Its contribution to solution improvements is therefore rather limited at long runtimes.



(a) High-impact components - 2E-VRP



(c) Low-impact components - 2E-VRP

Figure 2.9: Analysis of components of HALNS for 2E-VRP

In light of the even slightly worse results when including *local improvement* and *infeasible solutions*, we reconduct the experiment with MDVRP instances. Contrary to the results when solving the 2E-VRP, we find that the two components significantly impact the solution quality when solving the MDVRP (see Figure 2.10). The presence of infeasible solutions seems to be especially important when solving MDVRP instances, which include additional restrictions on the route length. This may be due to a well-fitting penalty weight ω for MDVRP instances or to characteristics of the instances, so that good solutions are close to infeasible solutions.

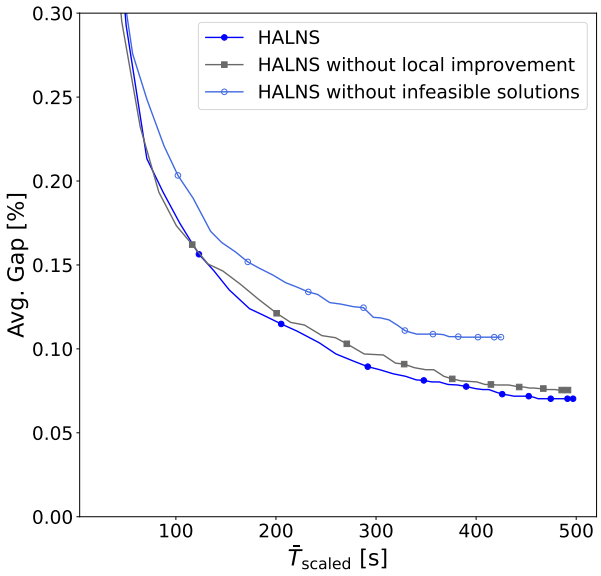


Figure 2.10: 2E-VRP low-impact components are high-impact components for MDVRP

In summary, the *simulated annealing*, *diversity management* and *hybridization* components strongly influence the solution quality when solving 2E-VRP instances. *Tabu search* and *parameter adaptation*, on the other hand, only have a medium influence on solution quality. However, our objective is a single solution approach for all the three problem classes and even components that have little effect on solution quality when solving 2E-VRP instances have a positive effect when solving other problem classes, i.e., LRP or MDVRP. All in all, we can conclude that a sophisticated combination of components is needed to improve solution quality at competitive runtimes for several problem classes of the 2E-VRP.

2.6 Summary and Further Areas of Research

Summary The design and operation of delivery systems also requires the determination of depots from which to deliver to customers. This general problem can be referred to as VRP with depot location decisions. We distinguish three variants of this problem class, the two-echelon VRP (2E-VRP), the location routing problem (LRP), and the multi-depot VRP (MDVRP). The literature offers a huge variety of quite specialized approaches for solving each of these problems.

We reviewed and classified these approaches and present a performance ranking. All three problems, however, can be solved with a single solution approach, since both the LRP and the MDVRP are special cases of the 2E-VRP. We developed a new metaheuristic method, the hybrid adaptive large neighborhood search (HALNS) based on the solution framework proposed by Voigt et al. (2021) that can solve all three problem classes. Computational experiments on benchmark sets from the literature demonstrate that the HALNS outperforms all approaches for the 2E-VRP and performs equally well with heuristics that are dedicated either to the LRP or the MDVRP. Furthermore, the HALNS shows robust results for all benchmarks, i.e., the differences between best and average runs are remarkably small. Additionally, the HALNS achieves three new best solutions

for LRP instances. Further, we have identified that different heuristic components of the HALNS with low performance impact when solving the 2E-VRP become relevant when solving LRP and MDVRP.

Further Areas of Research One main difficulty when inserting or removing customers during the search procedure is to approximate the impact of opening or closing a depot. Machine learning approaches could be used to approximate the resulting cost changes. Furthermore, the HALNS uses multiple ALNS runs to generate the population-based solution space. This structure offers the possibility of easy parallelization. The HALNS especially outperforms all pure ALNS implementations on these problem classes, demonstrating the value of an enlarged search space by using neighborhood- and population-based approaches. An interesting avenue for future research could therefore be the hybridization of existing and forthcoming ALNS approaches for other variants.

Contribution 3 - Hybrid Adaptive Large Neighborhood Search for the Traveling Salesman Problem with Time Windows and Adjusted Costs

3 Hybrid Adaptive Large Neighborhood Search for the Traveling Salesman Problem with Time Windows and Adjusted Costs

Stefan Voigt

Abstract This article models the underlying problem of the Amazon/MIT last mile routing research challenge as Traveling Salesman Problem with Time Windows (TSPTW). It uses an efficient heuristic based on the recently proposed Hybrid Adaptive Large Neighborhood Search (HALNS) to solve the TSPTW. Furthermore, the approach leverages information on high-quality routes to adjust the traveling costs and therefore guide the HALNS to mimic the behavior of drivers. Thus, the overall approach follows a classical Operations Research paradigm without using sophisticated Machine Learning (ML) techniques. Nevertheless, the HALNS generates a population of individuals, which may be used as additional features in an ML approach. We achieve an average score of 0.0678 on test data consisting of all 2718 high-quality routes.

Published: Last Mile Routing Research Challenge Proceedings
URL: `https://hdl.handle.net/1721.1/131235`

3.1 Introduction

The Amazon/MIT last mile routing research challenge (*Amazon Challenge*[1]) covers a routing problem, where routes with few hundreds of customers have to be constructed route by route, i.e., independently. Some customers have time windows during which the delivery must be completed. Additionally, servicing a customer takes time. The problem is similar to a Traveling Salesman Problem with Time Windows (TSPTW). The Traveling Salesman Problem (TSP) is a routing problem where one salesman (delivery vehicle) is required to visit each location (customer) once, starting at and returning to the same location (depot). However, the goal of the *Amazon Challenge* differs from the goal of the classical TSP, as not only the traveling costs are to be minimized, but the route should be similar to a high-quality route realized by an actual driver. The driver may visit customers in a different sequence as in the route suggested, for a couple of reasons, like avoiding traffic jams, lack of parking spaces, more convenient roads, avoiding left-turns, and so on. The data provided by the *Amazon Challenge* can be leveraged to identify and include such strategies into constructing solutions. Considering the size of the problem with a few hundred customers and the structure of the problem, both Machine Learning (ML) and Operations Research (OR) approaches seem to be well-suited to the problem. This article however, follows a classical OR process, consisting of the phases problem definition, formulation of a mathematical model, development of a solution procedure, testing and refinement (see e.g., Hillier and Lieberman (1995)). It is the author's belief that the performance of any ML approach used in the field of optimization must be compared to a state-of-the-art OR approach. The lack of an OR based benchmark is a shortcoming often found in ML literature applied to classical optimization problems, like the TSP (see Section 3.2).

The problem is modeled as TSPTW and solved with a metaheuristic solution approach based on the recently proposed Hybrid Adaptive Large Neighborhood Search (HALNS) by Voigt et al. (2021). The HALNS generates a population of solutions with an efficient ALNS, which are then improved by a crossover phase. Before the HALNS is applied, the cost matrix (costs for traveling from node to node) is adjusted to capture some of the behavior of drivers. Additionally, we tested different strategies, e.g., violating time windows, and tried to predict which cost adjustment/strategy should be applied depending on instance features. Unfortunately, the prediction accuracy was not good enough. Henceforth, we simply use the cost adjustment/strategy with the lowest average score.

This article makes several contributions: (1) The approach suggested can serve as a baseline for comparing ML approaches. (2) Information gathered within the population may be included as additional features in ML approaches. (3) An ML model (with sufficient accuracy) may leverage the proposed strategies.

The remainder of this paper is structured as follows: Section 3.2 shows related literature. Section 3.3 models the problem as TSPTW, introduces the cost adjustments, and presents the HALNS. Section 3.4 experiments with different adjustments of the cost matrix and strategies drivers may follow. Finally, Section 3.5 concludes the work and indicates avenues for further research.

[1] Please refer to `https://github.com/MIT-CAVE/rc-cli` for details on the scoring process and to access the data.

3.2 Literature Review

The TSP is one of the most studied $\mathcal{NP}$-hard optimization problems in the field of OR with many scientific papers and books published (e.g., Applegate et al. (2007), Cook (2012)). There exist many approaches to solve the TSP heuristically. A very well-performing heuristic is the implementation of the Lin-Kernighan heuristic by Helsgaun (2000) (LKH)[2]. Approaches for solving large-scale instances with millions of cities include POPMUSIC (Taillard and Helsgaun 2019). The method of choice for exactly solving TSPs is the Concorde solver, which is able to solve instances with thousands of cities (Applegate et al. 2006)[3]. Despite the $\mathcal{NP}$-hardness, both heuristic and exact approaches can solve instances with few hundreds of customers within reasonable runtime, as demanded in the *Amazon Challenge*. This review neglects the time window component since it can efficiently be integrated into heuristic approaches (Vidal et al. 2015).

Mainly in the last decade, researchers developed ML approaches for graph problems and use the TSP as testing ground for their models. ML approaches either construct the solution directly or use ML together with classical optimization techniques.

ML combined with optimization techniques Arnold and Sörensen (2021) use ML within a metaheuristic solution approach to identify high-order moves and improve the performance of state-of-the-art heuristics for the Vehicle Routing Problem. For a general overview of ML at the service of metaheuristics please refer to Karimi-Mamaghan et al. (2021). Instead of integrating ML within the metaheuristic, Hutter et al. (2014) use ML to predict the performance of algorithms depending on instance characteristics. This approach is similar to our idea of solving the TSPTW with different cost adjustments/strategies and trying to predict which strategy should be applied for solving the instance. Closely related is ML-based hyperparameter tuning, where an ML model tunes the parameters of heuristics depending on instance characteristics (e.g., *irace package* by López-Ibáñez et al. (2016)).

ML models constructing solutions directly There exists a variety of ML models that try to solve the TSP directly. Bello et al. (2017) use neural networks with reinforcement learning and negative tour lengths as reward signal. They solve instances with up to 100 cities and use LKH, Google OR-Tools and Concorde as benchmark. They achieve near-optimal results but are outperformed by LKH, which finds optimal results in similar runtime. Dai et al. (2018) propose a combination of reinforcement learning and deep graph embedding. The authors train their model with instances ranging from 50-100 cities and apply the trained model on instances with up to 1,200 cities achieving a rather high optimality gap. The approach is compared to simple heuristics. However, a benchmark against state-of-the-art heuristics is missing. Kool et al. (2019) present a model with attention layers, solving instances with up to 100 cities and outperform simple heuristics. Dwivedi et al. (2020) present a framework for benchmarking graph neural nets, showing that Gated-Graph Convolutional Networks achieve promising results for solving the TSP.

[2] `http://akira.ruc.dk/~keld/research/LKH-3/`
[3] `https://www.math.uwaterloo.ca/tsp/concorde.html`

To summarize, models that directly construct solutions fall short compared to the performance reached by state-of-the-art heuristics/solvers and work only for instances with few hundreds of customers. Furthermore, most approaches generalize poorly on larger instances (Joshi et al. 2020) and have issues with additional constraints, e.g., time windows. It remains to be seen if these issues can be solved in the future - maybe the *Amazon Challenge* stimulates research in this direction. Right now, classical optimization techniques together with ML seem to be more suitable than ML models that directly construct solutions.

3.3 Methodology

Section 3.3.1 models the problem as TSPTW. Section 3.3.2 details strategies and cost adjustments. Section 3.3.3 presents the HALNS to solve the TSPTW with then adjusted costs. Lastly, Section 3.3.4 introduces the data-driven approach for leveraging the given data.

3.3.1 Model for the traveling salesman problem with time windows

The TSPTW can be defined on a directed graph $G(N, A)$ with node set N, consisting of the depot 0 and customers $j \in C$, and arc set A. The arc set is defined as $A = \{(i, j) : i \neq j,\ i, j \in N\}$. c_{ij} denotes the associated transportation costs. The vehicle starts and ends at the depot. Customers must be visited within their time window defined by lower limit e_j and upper limit l_j. The duration of a tour is determined by the traveling times denoted by t_{ij}, by the service time S_j, and eventually waiting times between customers. The duration may not exceed the delivery period's length D. Table 3.1 summarizes the notation.

Table 3.1: Notation

Sets	
C	Set of customers, $C = \{1, ..., \|C\|\}$
N	Set of nodes, $N = \{0\} \cup C = \{0, ..., \|C\|\}$
Parameters	
c_{ij}	Traveling cost from i to j, $i, j \in N$
t_{ij}	Traveling time from i to j, $i, j \in N$
S_j	Service duration at customer j, $j \in C$
e_j	Earliest start of service for customer j, $j \in C$
l_j	Latest start of service for customer j, $j \in C$
D	Length of delivery period
Decision Variables	
x_{ij}	Binary variable indicating whether arc (i, j) is used, $(i, j) \in A$
s_i	Start time of service at node i, $i \in N$

Model TSPTW

$$\text{Minimize } C_{\text{TSPTW}} = \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{3.1}$$

s.t.

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in C \tag{3.2}$$

$$\sum_{j \in N} x_{ij} = \sum_{j \in N} x_{ji} \qquad \forall i \in N \tag{3.3}$$

$$s_j + S_j + t_{j0} \leq D \qquad \forall j \in C \tag{3.4}$$

$$s_j - s_i \geq (t_{ij} + S_i)x_{ij} - D(1 - x_{ij}) \qquad \forall i, j \in C,\ i \neq j \tag{3.5}$$

$$e_j \leq s_j \leq l_j - S_j \qquad \forall j \in C \tag{3.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in N \tag{3.7}$$

$$s_i \in \mathbb{R}_0^+ \qquad \forall i \in N \tag{3.8}$$

The objective function (3.1) minimizes traveling costs. Constraints (3.2) ensure that each customer is visited exactly once. Constraints (3.3) conserve flow. Constraints (3.4) ensure that the vehicle returns in time. Constraints (3.5) guarantee that the service starts after traveling time plus service time starting from its predecessor. Constraints (3.6) ensure that service starts and ends within the time window. Please note, that this restriction is more restrictive than usual assumed, as the service must generally only start before the upper limit of the TW. Constraints (3.7) and (3.8) define the domains of the variables.

3.3.2 Strategies and adjustments of cost matrix

Before applying the HALNS for solving the TSPTW, the given traveling cost matrix is adjusted to mimic the behavior of the driver. This subsection details the strategies and cost adjustments. Please note, that traveling costs are originally equivalent to traveling times, i.e., $c_{ij} = t_{ij} \quad \forall i, j \in N$. However, after adjusting the cost matrix, traveling costs will in general differ from traveling times.

Triangle inequality This adjustment simply checks if the triangle inequality holds for all arcs. The triangle inequality holds, if it is impossible to achieve lower costs by traveling via another node, k, instead of traveling directly from node i to j. The traveling costs of arcs are adjusted until the triangle inequality holds for all arcs. Costs are adjusted by finding the shortest paths for all nodes as follows.

$$c_{ij} = \begin{cases} c_{ij}, & \text{if } c_{ij} \leq c_{ik} + c_{kj} \quad (3.9) \\ c_{ik} + c_{kj}, & \text{else} \quad (3.10) \end{cases}$$

Unreasonable expensive arcs This adjustment checks if the costs given for an arc are unreasonably high compared to the great-circle distance (gcd) of node i and node j with corresponding coordinates $coord_i$ and $coord_j$. Costs are adjusted as follows. The factor 1.3 is chosen after preliminary testing.

$$c_{ij} = \begin{cases} \min(c_{ij}, c_{ji}), & \text{if } c_{ij} > 1.3 \cdot \text{gcd}(coord_i, coord_j) \quad (3.11) \\ c_{ij}, & \text{else} \quad (3.12) \end{cases}$$

The reasoning behind this adjustment is that drivers may use the arc if the arc has a small gcd and the difference between the way from i to j and back (j to i) are different, as this may indicate

an error in the data. The minimum of c_{ij} and c_{ji} can be interpreted as the best guess for correcting the data.

Depot arcs The depot is usually located outside the city and therefore connected by a highway with the delivery area. Drivers may accept higher traveling costs (longer traveling time) on highways, if this means, that they can in turn reduce the distance traveled within the delivery area - assuming that the highway is more convenient to drive. Costs are adjusted as follows. The factor 0.1 is chosen after preliminary testing.

$$c_{0j} = 0.1 \cdot c_{0j} \quad \forall j \in N \tag{3.13}$$

$$c_{j0} = 0.1 \cdot c_{j0} \quad \forall j \in N \tag{3.14}$$

Time windows The last strategy disregards all time windows, resulting in solving a classical TSP. Drivers may disregard time windows of some or of all customers, because they are for example short on time.

3.3.3 Hybrid adaptive large neighborhood search for solving TSPTW

This subsection briefly introduces the Hybrid Adaptive Large Neighborhood Search (HALNS) which is used to solve the TSPTW. The metaheuristic is derived from the recently proposed HALNS for the Vehicle Routing Problem with Availability Profiles. For a detailed explanation, please refer to Voigt et al. (2021). Algorithm 5 describes the HALNS. The HALNS generates an initial population, P of n^{P} solutions by iteratively applying an efficient ALNS (lines 1-3). Afterward, the HALNS executes crossovers within that population (line 6-9) During crossover, two individuals - one solution from the population, s and the global best solution, $\hat{s}$ - are combined by using the ALNS, instead of using an explicit crossover operator. The global best solution guides the removal procedure. Customers are removed if their successors differ in both solutions, i.e., if they are placed differently in both solutions. After crossover, the procedure selects surviving individuals (line 10) according to objective value and diversity. The procedure ends after $gen^{\max}$ generations.

Algorithm 5: Hybrid Adaptive Large Neighborhood Search

```
1  while |P| < n^P do                                          // Initial Population
2  |   s ← ALNS()
3  |   P ← P ∪ {s}
4  while gens < gen^max do                                     // Generations
5  |   ŝ ← DetermineBestSolution(P)
6  |   while i < n^P do                                        // Crossover and Education Phase
7  |   |   s ← P[i]
8  |   |   s ← ALNS(s, ŝ)
9  |   |   P ← P ∪ {s}
10 |   P ← DiversityManagement(P)                              // Select survivors and manage diversity
```

Algorithm 6 describes the ALNS algorithm. The ALNS was proposed by Ropke and Pisinger (2006) and has been applied to a variety of routing problems (Pisinger and Ropke 2007, Hemmelmayr et al. 2012, Voigt and Kuhn 2022). The algorithm begins with setting the local best solution

Algorithm 6: Adaptive Large Neighborhood Search used within HALNS

Input : Starting solution s, global best solution $\hat{s}$
Output: best solution s^*

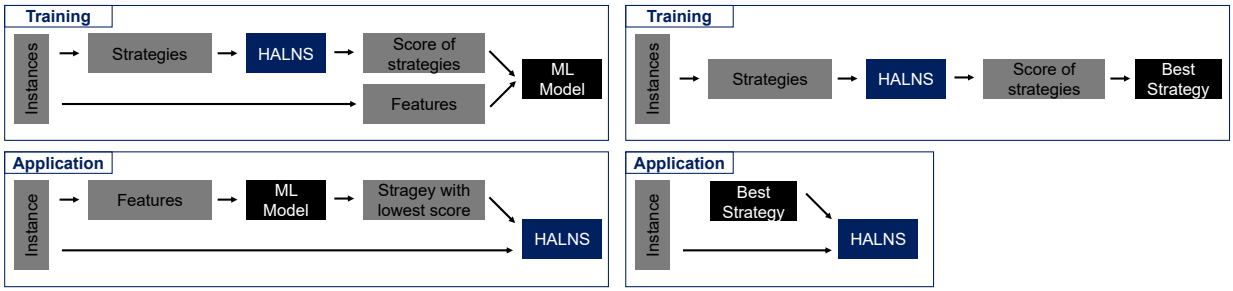
1 $s^* \leftarrow s$
2 **while** *Iterations without improvement* $< it^{\text{stop}}$ **do**
3 ChooseOperators()
4 $C_R^C \leftarrow$ getRemovalCandidates$(s, \hat{s})$ // Determine Removal Candidates
5 $(s^{\text{new}}, C_R) \leftarrow$ Remove(s, C_R^C) // Remove Customers
6 $C_R \leftarrow$ sort(C_R) // Determine Insertion Order
7 $s^{\text{new}} \leftarrow$ Insert(s^{new}, C_R) // Insert Customers
8 **if** $f(s^{\text{new}}) < f(s^*)$ **then**
9 **if** $f(s^*) < f(\hat{s})$ **then**
10 $\hat{s} \leftarrow s^*$
11 **else if** *accept*$(f(s^{\text{new}}), f(s^*), \tau)$ **then**
12 $s \leftarrow s^{\text{new}}$ // Simulated Annealing
13 $\tau \leftarrow \tau \cdot \alpha$
14 UpdateParameters()

s^* to the starting solution, s (line 1). The ALNS uses the global best solution $\hat{s}$, as soon as it exists, i.e., in all but the first generation. After initialization, the ALNS generates a new solution s^{new} by removing and inserting customers with randomly chosen removal operators and a deterministic insertion operator, as there is only one insertion operator (line 3). The probability of choosing removal operators depends on their performance during previous iterations. By comparing the current solution s and the global best solution $\hat{s}$, a set of customers which are candidates for removal, C_R^C is generated (line 4). The removal operator chosen then removes some or all customers from the solution which are included in set C_R^C (line 5). The maximum number of customers to be removed in one iteration, q^{binom}, is sampled from a binomial distribution with sample size $|C|$ and probability p^{binom}. The number of customers that are actually removed by one of the below removal operators is expressed by $q^{\text{r}} = \min\{q^{\text{binom}}, |C_R^C|\}$. The ALNS uses four removal operators: Random Removal, Historic Cost Removal, Worst Cost Removal, and Shaw Removal (for details, see Voigt et al. (2021)). The removed customers are added to set C_R. After sorting C_R according to data collected during the search (line 6), the customers are inserted using the Best Insertion Operator (line 7). This operator simply finds the position, where the customer can be inserted with the lowest total cost while respecting time windows. The ALNS uses simulated annealing to escape local optima (lines 8-12). Deteriorating solutions are accepted with a probability depending on the difference in costs of the candidate solution $f(s^{\text{new}})$, the cost of the best solution obtained during the run of the ALNS $f(s^*)$ and the current temperature τ. The initial temperature is determined instance-specific with $\tau = -\frac{\overline{\Delta E}}{ln(\chi_0)}$ using the formula from Johnson et al. (1989). At the end of every iteration the temperature τ is multiplied with the cool rate α (line 13) and the parameters are updated (line 14).

The determination of removal candidates, adaptivity of parameters, and diversity management are equivalent to the HALNS originally proposed (Voigt et al. 2021).

3.3.4 Data-driven approach

This subsection presents the planned and then realized approach to leverage the data. During training, the planned approach (top of Figure 3.1a) solves instances using different strategies with the HALNS. The solutions are evaluated, resulting in instance-specific scores of strategies. An ML model is then trained to predict which strategy should be applied depending on instance features to achieve the lowest score. Unfortunately, the prediction accuracy of several tested ML models was not good enough during application (bottom of Figure 3.1a) to justify the additional complexity. Therefore, the final approach simply uses the cost adjustment/strategy with the lowest average score (*Best Strategy*) identified during experiments in Section 3.4 independent of instance features (see Figure 3.1b).



(a) Planned approach with ML model

(b) Realized approach without ML model

Figure 3.1: Planned vs. realized approach

3.4 Experiments and Results

All experiments were conducted in parallel on an AMD Ryzen 7 2700X CPU with eight cores and 32 GB of RAM. Data and instances are prepared in Python, the HALNS is coded in C++. The test data set contains all 2718 high-quality routes given in the *Amazon Challenge*. The delivery period's length D is set to 9.5 hours for all experiments.

In addition to the strategies mentioned in Section 3.3.2, a combination of strategies and the influence of the number of generations were tested. All but the last two strategies use ten HALNS generations, $gen^{\max} = 10$.

- TSPTW: Solve TSPTW without adjustments.
- TSP: Solve TSP, i.e., disregard time windows.
- TSPTW-tri: Solve TSPTW with triangle inequality adjustment.
- TSPTW-gcd: Solve TSPTW with adjustment of unreasonable expensive arcs.
- TSPTW-depot: Solve TSPTW with adjustment of depot traveling costs.
- TSPTW-tri-gcd-depot: Combination of cost adjustments.
- TSPTW-tri-gcd-depot-1: Combination of cost adjustments, $gen^{\max} = 1$.
- TSPTW-tri-gcd-depot-50: Combination of cost adjustments, $gen^{\max} = 50$.

Table 3.2 shows the average score and standard deviation for the different strategies. Interestingly, the scores of *TSPTW* and *TSP* are similar. This may indicate, that time windows are often not restrictive. All other cost adjustments reduce the average score compared to strategy *TSPTW*. The combination of strategies further reduces the score. Rows *TSPTW-tri-gcd-depot-1* and *TSPTW-tri-gcd-depot-50* show, that the score is only slightly affected by the number of generations, indicating that already good solutions are found in just one generation. The second-last row shows the theoretical performance of combining all strategies, assuming a perfect prediction model. Unfortunately, the ML model constructed was not able to predict the strategies with sufficient accuracy. Henceforth, the final approach uses simply the best strategy, i.e., *TSPTW-tri-gcd-depot-12*. As mentioned before, increasing the number of generations does not necessarily improve the score. Therefore, the number of generations is only slightly increased to $gen^{\max} = 12$, also to remain well within the time limit of 240 minutes.

Table 3.2: Results

Strategy	Average Score	Standard Deviation	Total Runtime [min]
TSPTW	0.0754	0.0512	104
TSP	0.0758	0.0512	103
TSPTW-tri	0.0748	0.0508	104
TSPTW-gcd	0.0721	0.0499	117
TSPTW-depot	0.0708	0.0522	103
TSPTW-tri-gcd-depot	0.0683	0.0513	120
TSPTW-tri-gcd-depot-1	0.0698	0.0517	38
TSPTW-tri-gcd-depot-50	0.0679	0.0510	461
Result assuming perfect prediction	0.0467	0.040	NA
Instead, chosen strategy: TSPTW-tri-gcd-depot-12	0.0678	0.0520	138

Figure 3.2 shows the distribution of scores for the chosen strategy, *TSPTW-tri-gcd-depot-12* for all 2718 high-quality routes tested. The majority of routes has a score ranging from 0.0 to 0.25. Only few routes have scores above 0.30. For these high-scoring routes, drivers may have followed other strategies.

3.5 Conclusion

Summary This paper models the *Amazon Challenge* as Traveling Salesman Problem with Time Windows and uses the Hybrid Adaptive Large Neighborhood Search as solution approach. The cost matrix is adjusted to mimic the behavior of drivers. The results show, that there exist arcs with an unreasonably high difference in the way to and from a customer/depot, resulting in the tendency of drivers to still use these arcs even if the costs are supposedly high. Additionally, the costs resulting from traveling from the depot to the delivery area (and back) are less important, meaning that drivers tend to accept a long distance on their way to the delivery area (and back) if this results in short distances within the delivery area. This can be explained by the availability and convenience of highways to get from the depot to the delivery area (and back). Furthermore, the combination of different strategies results in a good score. A model that predicts which strategy should be used depending on the instance with sufficient accuracy has yet to be developed.

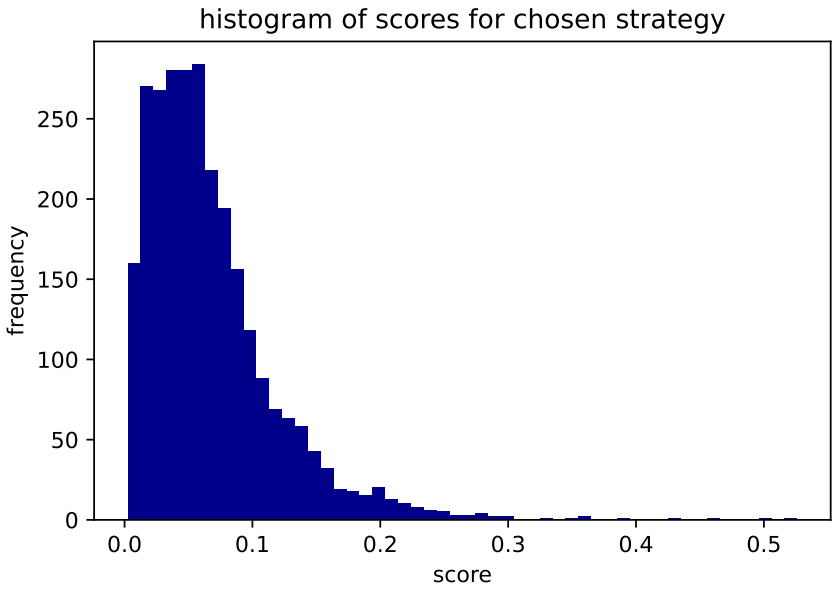


Figure 3.2: Histogram of scores for chosen strategy *TSPTW-tri-gcd-depot-12*

Avenues for further research There exist three avenues for further research, building on this work:

- **OR**: Interview drivers to get insights into strategies they may follow, e.g., avoiding left-turns, respecting breaks and customer-availability. Construct optimization models that include such strategies.
- **ML combined with optimization techniques**: Implement a model with higher prediction accuracy for predicting which strategy should be used depending on instance characteristics. Additionally, ML-based insertion operators could be integrated into the HALNS.
- **ML models directly constructing solutions**: Predict the sequence directly by using Gated-Graph Convolutional Neural Networks, complemented with features from the HALNS population.

Contribution 4 - Crowdsourced Logistics: The Pickup and Delivery Problem with Transshipments and Occasional Drivers

4 Crowdsourced Logistics: The Pickup and Delivery Problem with Transshipments and Occasional Drivers

Stefan Voigt, Heinrich Kuhn

Abstract This article considers a setting in which a courier, express, and parcel service provider operates a fleet of vehicles with regular drivers (RDs) to ship parcels from pickup to delivery points. Additionally, the company uses a platform where occasional drivers (ODs) offer their willingness to take on requests that are on or near the route they had originally planned. There exist transshipment points (TPs) to better integrate these ODs. ODs or RDs may transfer load at these predetermined TPs. The problem is modeled as a mixed-integer programming model and called pickup and delivery problem with transshipments and occasional drivers (PDPTOD). We develop a solution approach based on an adaptive large neighborhood search. The article provides insights on how the number and location of TPs impact the cost advantages achieved by integrating ODs. It also shows that the cost savings are highly sensitive to the assumed flexibility and compensation scheme.

Published: Networks
URL: https://doi.org/10.1002/net.22045

4.1 Introduction

The growth in online sales during recent decades combined with consumer expectations of a fast, cheap and on-time delivery service necessitates the development of new and more cost-efficient delivery concepts. The global share of e-commerce in retail sales is predicted to more than double from 7.4 % in 2015 to 15.5 % in 2021 (Sampaio Oliveira et al. 2019). At the same time, consumer expectations are rising. According to a study conducted by McKinsey on the future of last-mile delivery, same-day delivery will grow to 25 % of B2C and C2C (business-to-consumer and consumer-to-consumer) volume (Joerss et al. 2016). Nevertheless, they find that only a fraction of customers is willing to pay an appropriate premium for this kind of service. Traditional grocery retailers recognize the growing importance of online sales and have accepted the challenge of creating a fast and inexpensive delivery service for their customers by introducing crowdshipping (Hübner et al. 2019, Hübner et al. 2016), among other measures.

Crowdshipping applies the concept of crowdsourcing to logistics. Participants are people who become couriers, so-called occasional drivers (ODs). ODs can be commuters or travelers who are already traveling or dedicated non-professional or professional drivers (McKinnon 2016). If commuters or travelers offer their services on a crowdshipping platform, the marginal costs, in economic and environmental terms, will be very low. Crowdshipping therefore offers a chance of reducing costs, increasing the speed of delivery and at the same time reducing environmental impact. This makes it a promising concept for mastering some of the challenges that will arise in last-mile logistics.

Crowdsourcing is a participative online activity in which an individual, institution, non-profit organization, or company proposes to a group of individuals that they voluntarily undertake a task via a flexible, open call. The undertaking of the task should entail mutual benefit (Estellés-Arolas and de Guevara 2012). This definition of crowdsourcing implies several characteristics that apply to crowdshipping to the same extent. First, a crowdshipping platform operates in digital form and all participants need access to mobile technology. Furthermore, the undertaking of the task is voluntary and flexible, which leads to challenges for the platform provider in sustaining and guaranteeing a certain level of service quality. As ODs are not obliged to accept requests, solutions may not be robust if many ODs decline the fulfillment of requests. Mechanisms to minimize the number of declined requests are for example scores and reviews for ODs that exclude unreliable ODs from the platform, or bonuses that promote reliable ODs. The platform provider supposedly cuts costs, customers may benefit from increasing delivery service, and ODs from an additional source of income. In addition to monetary compensation for ODs, a variety of possible incentives exists, e.g., some ODs are motivated by social or moral considerations, some just want to pass the time (Devari et al. 2017). The crowd consists of a group of individuals and is therefore diverse. The platform needs to consider this diversity by offering a certain flexibility in respect of compensation, the number of pickups and dropoffs per trip, the time windows for service, and permitting detouring in order to reach a critical mass of ODs. If the platform wants to allow multiple pickups and dropoffs in a single trip, the routing has to be considered, which makes the problem even more challenging (Arslan et al. 2018). In addition, it is crucial to generate trust between the participants. Trust-

generating mechanisms include for example background checks, reviews and testimonials (Einav et al. 2016). Another challenge, yet unsolved, is the possible exploitation of ODs. ODs have the tendency to under-estimate the value of their time and the vehicle operating costs (McKinnon 2016). A possible solution is a platform that only allows cost-efficient matchings for both sender and courier by providing fair compensation.

We consider a setting in which a distributor (e.g., a courier, express and parcel service provider or an omnichannel retailer) operates a fleet of vehicles with regular drivers (RDs) to ship parcels from pickup to delivery points, e.g., from retailer to online-customers or from one individual to another. Additionally, the distributor uses a platform where ODs offer their willingness to fulfill pickup/delivery tasks. We assume that the platform uses trust-generating mechanisms, scoring and bonus systems to increase the reliability of ODs. ODs are therefore highly motivated to fulfill the pickup/delivery tasks that have been assigned to them. As a result, the solutions generated are stable, i.e., routes for RDs and ODs are likely to be realized. In order to increase flexibility we consider the possibility of transshipments between couriers at certain locations called transshipment points (TPs), multiple tasks in a single trip for ODs and flexible compensation. At these predetermined TPs, drivers (ODs or RDs) can hand over or take on their freight. Several planning problems arise in this context. On a strategic level the distributor, for example, has to determine the location and capacity of the TPs based on aggregated and estimated demand. On an operational planning level the distributor has to decide which tasks are assigned to RDs, which to ODs, in which sequence the drivers attend to the customers, and whether TPs are used. The present paper focuses on the daily operational planning of routes and assumes that TPs are used on a flexible basis depending on demand structure. The distributor aims to minimize the overall costs arising from the distance traveled by RDs and the compensation paid to ODs. ODs receive compensation or payment depending on the additional effort required to fulfill their task(s), i.e., their compensation or payment depends on the length of the detour and the number of additional stops. Furthermore, we assume a static setting where requests and offers from ODs are known in advance and are not time critical. This assumption renders the system impractical for same-day delivery, but is reasonable due to the increasing number of possible matches. As mentioned before, the platform needs to reach a critical mass to sustain a certain service level. Especially in the early phase of a platform where the number of couriers is insufficient, i.e., critical mass is not reached, lesser limitations regarding time restrictions will produce more matches and hence lead to a better service level and higher utilization of ODs. The lack of time windows for ODs is admittedly a simplifying assumption, but it is nevertheless reasonable to investigate the effects of TPs and the spatial flexibility of ODs on solutions.

This paper makes several contributions to the field of research. We introduce a new and challenging routing problem with regular drivers (RDs), occasional drivers (ODs) and transshipment points (TPs). We term this problem "Pickup and Delivery Problem with Transshipments and Occasional Drivers" (PDPTOD). The problem differs from existing modeling approaches in the following respects. (1) The PDPTOD focuses on ODs, i.e., drivers who already have an origin and destination. (2) ODs have realistic spatial limitations as their flexibility is expressed by the number of stops and their willingness to make a detour. (3) RDs are employed alongside ODs. (4) TPs are

integrated in the delivery process. Additionally, we develop a heuristic solution approach based on an adaptive large neighborhood search (ALNS) in a simulated annealing framework. Furthermore, we conduct extensive numerical experiments, providing insights on how the number and location of TPs impact the cost advantages achieved when ODs are integrated into the delivery process. We show that the cost advantages depend not only on the location and number of TPs but also on the flexibility of ODs, the compensation scheme and the interaction between the aforementioned aspects. On a more practical note, the model can be used for platforms that have not reached a critical mass of ODs and therefore cannot guarantee to serve all customers with ODs only.

The remainder of the paper is organized as follows. Section 4.2 describes the problem in detail. Section 4.3 discusses the related literature. Section 4.4 formulates the associated mixed-integer programming (MIP) model. Section 4.5 presents the solution approach suggested that is used in Section 4.6 for numerical experiments. Section 4.7 summarizes the fundamental findings of the paper and provides directions for future research.

4.2 Problem Description

We focus on a local distributor (e.g., a courier, express and parcel (CEP) service provider or an omnichannel retailer) operating its own fleet of vehicles with regular drivers (RDs) $\bar{k} \in \bar{K}$ to ship requests $r \in R$ from pickup $o(r)$ to delivery points $d(r)$, e.g., from retailer to online customers or from one individual to another. Every request has to be satisfied. The RDs can execute pickup and delivery tasks when there are not sufficient occasional drivers (ODs) available or if it is cheaper to use RDs, thus enabling the distributor to guarantee a certain service level. RDs start and end at the same depot and cause costs depending on the distance traveled. Besides RDs, the company uses an online crowdsourcing platform to obtain ODs for pickup and delivery tasks. Every OD $\tilde{k} \in \tilde{K}$ starts at its origin $o(\tilde{k})$ and end at its destination $d(\tilde{k})$. The distributor knows the pickup and delivery tasks in advance, e.g., one day before the start of the tour. The same holds for ODs, i.e., ODs express their willingness to participate the day before. This assumption is reasonable for trips well known in advance or regular trips, e.g., daily trips to work and back home. The platform mainly aims at commuters where the origin and destination are the same every working day. Professional drivers with spare capacity and fixed routes can also participate on the platform. ODs specify the maximum number of stops $\alpha_{\tilde{k}}$, maximum length of detours $\beta_{\tilde{k}}$, and the origins $o(\tilde{k})$ as well as the destinations $d(\tilde{k})$ of the respective trips originally planned. This gives ODs a reasonable amount of control over their tasks while not overtaxing them with too much information. In addition we assume that ODs have sufficient temporal flexibility, i.e., ODs are not obliged to indicate time windows, as we assume that all requests can be fulfilled in a reasonable amount of time. Of course, this is a simplifying assumption and may be critical if ODs have no temporal flexibility. However, hard time windows would decrease the number of possible matchings, which is especially unfavorable for platforms in their early stages. ODs and RDs are not obliged to deliver parcels directly, they can transfer parcels at predetermined transshipment points (TPs) $t \in T$ to another OD or RD. A TP can be any suitable location with sufficient capacity, for example shops or gas stations with a backroom, large parcel lockers or intermediate depots. We assume that these

facilities have a large enough capacity to handle all the parcels that will potentially need to be processed. As parcels may be stored at TPs for a limited time, drivers do not need to wait to hand over parcels to the following driver. The objective from the distributor's point of view is to minimize total costs while serving all customers. The costs are composed of routing costs for RDs c_{ij} and costs for ODs that are derived from fixed cost per stop and variable cost depending on the length of a detour compared to the length of their route originally planned $c_{o(\tilde{k})d(\tilde{k})}$. Note that if there are no ODs and no TPs the problem is reduced to the standard pickup and delivery problem (PDP) that typically arise in local area CEP services.

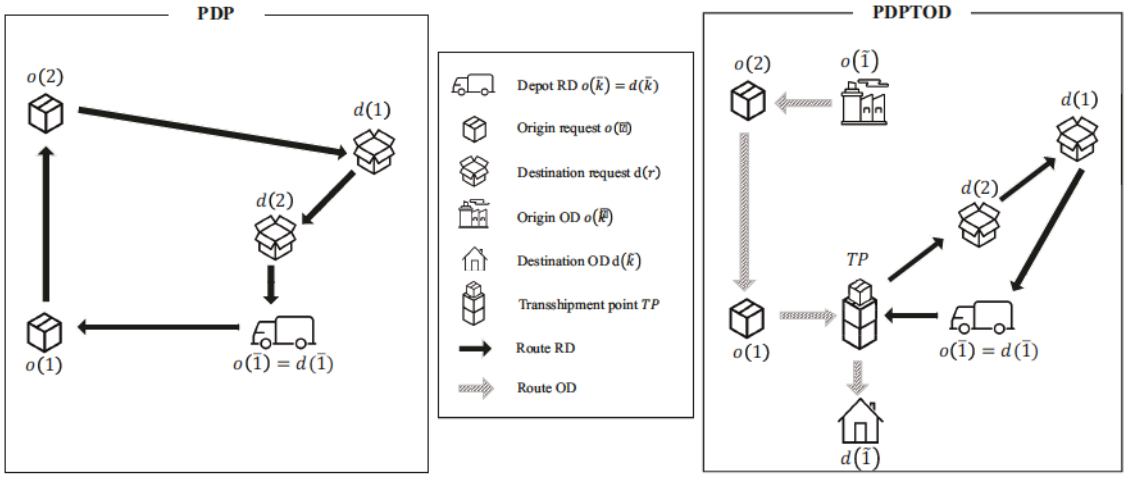


Figure 4.1: Illustrative example of PDP vs. PDPTOD solution

Figure 4.1 shows an illustrative example with two requests $r = 1$ and $r = 2$. Without ODs, the requests have to be served by one RD. The RD ($\bar{k} = \bar{1}$) starts at the depot $o(\bar{1})$, picks up the requests at $o(1)$ and $o(2)$, delivers the first request 1 to $d(1)$ and then request 2 to $d(2)$. Finally, the RD returns to the depot $d(\bar{1})$. The route of the RD is depicted with black arrows on the left of Figure 4.1. The solution changes if ODs and TPs are used as displayed on the right of Figure 4.1. An OD $\tilde{k} = \tilde{1}$ starts at her/his origin $o(\tilde{1})$ (for example her/his workplace), picks up both requests, drops off the load at the transshipment point TP and reaches her/his destination $d(\tilde{1})$ (grey route). The RD starts at the depot $o(\bar{1})$, picks up the dropped load from $\tilde{1}$ at TP, delivers request 2, then request 1, and returns to the depot $d(\bar{1})$.

4.3 Related Literature

The present section reviews relevant literature. First, Subsection 4.3.1 briefly presents literature that discusses general aspects of crowdshipping. Subsection 4.3.2 then gives an overview on modeling approaches that are closely related to our problem setting. Afterwards, Subsection 4.3.3 reviews exact and heuristic solution approaches for the pickup and delivery problem (PDP) since our modeling and solution approach are based on these approaches. Finally, Subsection 4.3.4 summarizes the findings from literature and identifies the specific contribution of our work.

4.3.1 Crowdshipping: Definition, challenges and opportunities

Many authors deal with fundamental questions of how crowdshipping can be used in practice, what challenges and opportunities arise, or how to define crowdshipping. Sampaio Oliveira et al. (2019) and Buldeo Rai et al. (2017) consider crowdshipping as an opportunity for city logistics, especially for the last mile. Doerrzapf et al. (2016) see crowdshipping as a chance to strengthen local shops in the competition with online retailers. McKinnon (2016) stresses the possible positive environmental impact by cutting down urban traffic levels and therefore emissions. In contrast, Qi et al. (2018) find that crowdshipping may increase emissions because occasional drivers (ODs) routes are prolonged and that most cost savings are due to the reducing of the fleet of regular drivers (RDs) rather than cutting operating costs directly. Le et al. (2019) present a review of current practice, research and case studies from a demand, supply and operations point of view, coming to the conclusion that a functioning crowdshipping system has to be fairly complex and needs features such as sophisticated pricing and matching procedures. Similarly, Rouges and Montreuil (2014) suggest using automated matching algorithms to increase the efficiency of crowdshipping platforms.

4.3.2 Crowdshipping problems

This subsection reviews publications that are directly related to our problem setting as all of them consider ODs within the delivery process of parcels and goods. These papers present modeling and solution approaches that match delivery requests and offers from ODs in a crowdshipping setting. Table 4.1 lists and classifies these approaches. Approximately half of these approaches consider transshipment points (TPs). Note that we omit literature related to other shared mobility concepts. Interested readers are referred to, e.g., Mourad et al. (2019).

Table 4.1: Literature overview of crowdshipping approaches

Author	**RDs**[1]	**R**[2]	**All**[3]	**TP**[4]	**C**[5]	**S**[6]
Soto Setzke et al. (2017)	n	1	n	n	v	s
Wang et al. (2016)	n	≥ 1	y	n	v	s
Archetti et al. (2016)	y	1	y	n	f	s
Macrina et al. (2017)	y	≥ 1	y	n	f	s
Dayarian and Savelsbergh (2020)	y	1	y	n	-	d
Arslan et al. (2018)	y	≥ 1	y	n	f, v	d
Kafle et al. (2017)	y	≥ 1	y	y	bids	s
Chen et al. (2017)	(y)	≥ 1	y	y	f, v	s
Raviv and Tenzer (2018)	n	≥ 1	y	y	v	d
Sampaio Oliveira et al. (2020)	n	≥ 1	y	y	v	s
Own Approach	y	≥ 1	y	y	f, v	s

[1] Regular drivers: y/n.
[2] Number of requests one OD can serve on a single trip: 1 or ≥ 1.
[3] All requests have to be served: y/n.
[4] Transshipment points: y/n.
[5] Compensation: Fixed per request (f), variable depending on detour (v), bids or none (-).
[6] Setting: static (s) or dynamic (d).

Crowdshipping without transshipments The following publications consider ODs fulfilling delivery requests but omit the option of using TPs. Soto Setzke et al. (2017) propose a matching algorithm based on routes and time feasibility modeled as min-cost max-flow model. Wang et al. (2016) provide a similar approach with the difference that all parcels have to be delivered and each OD is allowed to make multiple deliveries. Both models are suitable for large-scale datasets, but as a drawback require highly simplifying assumptions and lack the ability to integrate RDs in order to guarantee a certain service level. Archetti et al. (2016) develop the vehicle routing problem with occasional drivers (VRPOD) based on the classical capacitated vehicle routing problem (VRP), and propose a multi-start heuristic for the solution. In contrast to both modeling approaches mentioned previously, the VRPOD uses one RD as a backup option if no suitable OD is found. The OD can handle only one request per trip and in turn receives compensation depending on the distance between depot and customer, independent of the detour. The contribution of Archetti et al. (2016) shows the potential benefits of employing ODs and the importance of choosing an appropriate compensation scheme for ODs. Macrina et al. (2017) extend the VRPOD by allowing multiple deliveries, split deliveries and integrating time windows. Allowing multiple deliveries is one promising possibility to reduce the need for RDs and increase the use of ODs instead. Dayarian and Savelsbergh (2020) also set up on the VRPOD. They assume that demand and capacity, i.e., the number of ODs, changes over time. ODs are in-store customers willing to deliver at most one online order. They assume in contrast to Archetti et al. (2016) that ODs are always cheaper than RDs because their compensation is in the form of store credits rather than actual payments. They point out that a higher number of ODs with higher flexibility has the potential to decrease costs and at the same time improve service quality. Arslan et al. (2018) consider a peer-to-peer platform that uses foremost ODs and RDs only as a backup option, similar to Dayarian and Savelsbergh (2020). They assume that ODs can deliver more than one request per job. This means routing is necessary. They show that the use of ODs reduces costs and system-wide distance traveled.

Crowdshipping with transshipments All publications mentioned so far come without the possibility of transferring load at TPs and make simplifying assumptions in some cases, such as only one request per OD trip. The following publications are more closely related to our work as they integrate TPs and allow multiple deliveries per OD. Kafle et al. (2017) suggest using cyclists and pedestrians as ODs who are close to customers. In the first step, ODs submit bids to a truck carrier and relay parcels at TPs if they win the bid. The ODs can take on several requests and have to make sure that they are able to fulfill these requests on time and therefore organize their routes independently. This assumption leads to high transactional costs for ODs, reducing the attractiveness of offering their services on the platform. The approach reduces the distance the truck travels and cuts costs compared to pure truck delivery. Chen et al. (2017) develop the multi-hop driver-parcel matching problem. This problem is characterized by the possibility of transferring parcels multiple times between ODs. They assume that there is a shipping company with RDs that takes on tasks that cannot be served by ODs at additional cost. However, compared to our setting, they do not consider the routing of these RDs. Raviv and Tenzer (2018) develop an innovative approach based on a connected network of automatic service points similar to the approach of Chen et al.

(2017). Their work however considers the compensation scheme of ODs and incorporates service level considerations, i.e., parcels that are in the system longer are prioritized. ODs may pick up, drop off or transfer parcels at service points (SPs) that are along their original route. The ODs are compensated for stopping by these SPs. Similar to our work, ODs are compensated per stop and are allowed to make multiple stops at SPs, as long as they are on their route. In contrast to our work, there are no RDs as backup option, and the routes of ODs are predetermined.

A very closely related problem to our problem setting is developed by Sampaio Oliveira et al. (2020). Similar to our approach, they model the problem as a pickup and delivery problem (PDP) with TPs based on the model of Rais et al. (2014), and develop an ALNS to solve it with reasonable computation times. Our problem setting is different in the following respects. Sampaio Oliveira et al. (2020) assume that crowdshippers work for a given period of time and are paid on an hourly basis. Note that we particularly use the term "occasional drivers" to distinguish these types of drivers from crowdshippers who work part-time and would otherwise not be traveling. ODs, in contrast, are traveling anyway. In addition, the authors do not incorporate RDs as alternative option to serve requests. The crowdshippers are only constrained by the time window in which they want to work. Their model is therefore fitting for settings in which the crowdshipper works part-time for the service provider. This is the case for companies such as Amazon Flex or Uber-Freight, where crowdshippers indicate the period they are willing to work, but are not limited in any other way. We focus on a courier express and parcel provider that uses fairly constrained ODs (instead of part-time working crowdshippers) as an option, together with RDs. These RDs guarantee to serve all requests, as long as their capacity suffices.

4.3.3 Pickup and delivery problem

Pickup and delivery problems (PDPs) describe a class of VRPs, in which vehicles not only deliver goods, but also pick up goods. Some PDPs also permit transshipments within the delivery and/or pickup tours. We briefly review the available modeling and solution approaches for PDP with and without transshipments since the PDPTOD extends these problem classes.

A promising solution approach for the PDP with time windows (PDPTW) is the ALNS developed by Ropke and Pisinger (2006). They applied the heuristic to more than 350 benchmark instances and were able to provide new best solutions for more than 50% of them. More recently, the Lin-Kernighan-Helsgaun (LKH) heuristic that is well-known for its good performance on the traveling salesman problem (Helsgaun 2000) is extended to deal with many variants of VRPs by introducing penalties for constraints (Helsgaun 2017). The LKH is able to further improve best-known solutions of instances from the literature.

The literature on PDP with transshipments (PDPT) is rather limited. Drexl (2012, 2013) presents some practical applications where transshipments are involved. Cortés et al. (2010) give a new strict arc-based formulation for the PDPT and present a branch-and-cut solution approach that decreases computation time by approximately 90% compared to a standard branch-and-bound solver. Nevertheless, the instances for which they obtained solutions are remarkably small with six requests, two vehicles and just one TP. Rais et al. (2014) present a mixed integer formulation for the PDPT with a polynomial bounded number of constraints and variables. The MIP uses

two distinct flows, i.e., (1) the vehicle flow and (2) the request flow, which are matched by linking constraints. The proposed MIP formulation can be extended by simple modifications in order to represent several variants of the problem. The authors prove this capability by showing how to integrate time windows, split deliveries, heterogeneous vehicles with a flexible fleet size and flexible vehicle stops, i.e., a vehicle may end its route at a node differing from its original depot. However, their model does not cover the integration of constrained ODs, which can be used alongside RDs. Numerical results are based on small instances with up to 14 nodes and are therefore rather limited, as in Cortés et al. (2010).

The publications attempting to solve the PDPT exactly show that exact approaches are only able to solve small instances, so heuristic solution approaches are necessary. A promising heuristic solution approach for the PDPT developed by Masson et al. (2013) is an extension of the ALNS for the PDP by Ropke and Pisinger (2006). The publication of Masson et al. (2013) is motivated by an application where people with disabilities require transport, e.g., from their home to schools. Similar to our work, transfers are possible at designated transfer points, but only RDs starting from a set of depots are considered. They first implement an ALNS for the PDPTW without transfers based on four destroy and two repair operators. This method is then extended by three destroy operators, two repair operators and heuristics to choose appropriate transfer points to cope with the PDPT. The ALNS is tested on instances from the literature and also on a case study concerning the transport of disabled persons with a maximum of 193 requests and a maximum of 24 transfer points. The authors are able to show that the introduction of transshipments has the potential to reduce costs. However, the cost advantages greatly depend on the geographical characteristics of the instances.

4.3.4 Summary

Many publications focus on crowdshipping in general, but few papers deal with optimization models for matching ODs and demand. Early papers make highly simplifying assumptions, e.g., one request per trip (Soto Setzke et al. 2017, Wang et al. 2016). Archetti et al. (2016) were the first to model the problem as VRP, but still allow only one request per OD trip. The following publications extend the ideas of Archetti et al. (2016) by allowing multiple deliveries, split deliveries and imposing time windows (Macrina et al. 2017), covering the stochastic and dynamic aspects of the problem (Dayarian and Savelsbergh 2020) or turning the problem into a PDP (Arslan et al. 2018). These publications however neglect the possibility of using TPs to increase the flexibility of the system. Kafle et al. (2017) use TPs, but do not cover the routing aspect of the ODs. In contrast, Chen et al. (2017) neglect the routing of RDs. The setting of Sampaio Oliveira et al. (2020) is the most related to our work, but focuses on crowdshippers instead of ODs, as previously explained.

To sum up, the problem setting considered in our paper has not yet been addressed in the literature. We consider ODs who can serve multiple requests, alongside RDs and both type of drivers have the possibility to transfer loads at TPs. The problem enriches the literature on PDPs and crowdshipping by presenting a PDP variant with ODs and TPs. In the course of this we focus on drivers who already have an origin and destination, termed ODs. These ODs have realistic spatial limitations. Additionally, RDs are employed alongside ODs and TPs are integrated into

the delivery process to increase the use of ODs and further reduce costs. The combination of these aspects is unique in literature. From the methodological point of view, we present a MIP formulation and an extension of the ALNS, which underlines the wide applicability of the ALNS.

4.4 The Pickup and Delivery Problem with Transshipments and Occasional Drivers

The Pickup and Delivery Problem with Transshipments and Occasional Drivers (PDPTOD) is defined on a directed graph $G(N, A)$ with node set N and arc set A. The node set N consists of the origins $o(\tilde{k})$ and destinations $d(\tilde{k})$ of occasional drivers (ODs), $\tilde{k} \in \tilde{K}$, the starting $o(\bar{k})$ and ending depots $d(\bar{k})$ of regular drivers (RDs), $\bar{k} \in \bar{K}$, the locations of pickup $o(r)$ and delivery requests $d(r)$ and transshipment points (TPs) $t \in T$. For $i, j \in N$, we denote the arc from i to j as $(i, j) \in A$ and the associated costs using c_{ij}. The arc set consists only of reasonable arcs, for example there are no outgoing arcs from a destination depot. The model PDPTOD is based on the PDPT model suggested by Rais et al. (2014). Table 4.2 summarizes the sets, parameters, and decision variables defined. The PDPTOD is modeled afterwards.

Table 4.2: Notation used to formulate model PDPTOD

Sets	
K	Set of vehicles, $K = \bar{K} \cup \tilde{K}$, $K = \{1, \ldots, k, \ldots, \lvert\bar{K}\rvert + \lvert\tilde{K}\rvert\}$
$\bar{K}$	Set of regular drivers, start at origin $o(\bar{k})$ and end at destination $d(\bar{k})$, $\bar{K} = \{1, \ldots, \bar{k}, \ldots, \lvert\bar{K}\rvert\}$
$\tilde{K}$	Set of occasional drivers, start at origin $o(\tilde{k})$ and end at destination $d(\tilde{k})$, $\tilde{K} = \{\lvert\bar{K}\rvert + 1, \ldots, \tilde{k}, \ldots, \lvert\bar{K}\rvert + \lvert\tilde{K}\rvert\}$
R	Set of requests with origin $o(r)$ and destination $d(r)$, $R = \{1, \ldots, r, \ldots, \lvert R\rvert\}$
T	Set of transshipment points, $T = \{1, \ldots, t, \ldots, \lvert T\rvert\}$
Parameters	
$\alpha_{\tilde{k}}$	Maximum number of stops, $\tilde{k} \in \tilde{K}$
$\beta_{\tilde{k}}$	Factor limiting the length of tour, $\tilde{k} \in \tilde{K}$
$\gamma_{\bar{k}}$	Maximum duration of tour, $\bar{k} \in \bar{K}$
c_{ij}	Traveling costs from node i to j, $(i, j) \in A$
M	Sufficiently large number
p^{detour}	Compensation depending on length of detour for ODs
p^{stop}	Compensation per stop for ODs
τ_{ij}	Traveling time from node i to j, $(i, j) \in A$
Decision Variables	
s_{jrkl}	Binary variable indicating whether request r is transferred from vehicle k to vehicle l at node j, $s_{jrkl} = 1$ or not $s_{jrkl} = 0$, $j \in N$, $r \in R$, $k, l \in K$, $l \neq k$
t_{jk}^{arr}	Time of arrival of vehicle k at node j, $j \in N$ and $k \in K$
t_{jk}^{dep}	Time of departure of vehicle k at node j, $j \in N$ and $k \in K$
x_{ijk}	Binary variable indicating whether vehicle k uses arc (i, j), $x_{ijk} = 1$ or not $x_{ijk} = 0$, $(i, j) \in A$, $k \in K$
y_{ijkr}	Binary variable indicating whether vehicle k carries request r on arc (i, j), $y_{ijkr} = 1$ or not $y_{ijkr} = 0$, $(i, j) \in A$, $k \in K$, $r \in R$

Model PDPTOD

$$\text{Minimize} \sum_{k\in\bar{K}}\sum_{(i,j)\in A} c_{ij}x_{ijk} + \sum_{k\in\tilde{K}}\sum_{(i,j)\in A}(p^{\text{detour}}c_{ij} + p^{\text{stop}})x_{ijk} - \sum_{k\in\tilde{K}}\sum_{(o(k),j)\in A}(p^{\text{detour}}c_{o(k)d(k)} + p^{\text{stop}})x_{o(k)jk} \tag{4.1}$$

s.t.

$$\sum_{(i,j)\in A} x_{ijk} \leq 1 \qquad \forall k \in K,\ i = o(k) \tag{4.2}$$

$$\sum_{(i,j)\in A} x_{ijk} = \sum_{(j,l)\in A} x_{jlk} \qquad \forall k \in K,\ i = o(k),\ l = d(k) \tag{4.3}$$

$$\sum_{(i,j)\in A} x_{ijk} - \sum_{(j,i)\in A} x_{jik} = 0 \qquad \forall k \in K,\ \forall i \in N \setminus \{o(k), d(k)\} \tag{4.4}$$

$$t^{\text{dep}}_{ik} + \tau_{ij} - t^{\text{arr}}_{jk} \leq M(1 - x_{ijk}) \qquad \forall (i,j) \in A,\ \forall k \in K \tag{4.5}$$

$$t^{\text{arr}}_{jk} \leq t^{\text{dep}}_{jk} \qquad \forall j \in N,\ \forall k \in K \tag{4.6}$$

$$y_{ijkr} \leq x_{ijk} \qquad \forall (i,j) \in A, \forall k \in K,\ \forall r \in R \tag{4.7}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} y_{ijkr} = 1 \qquad \forall r \in R,\ i = o(r) \tag{4.8}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} y_{ijkr} = 1 \qquad \forall r \in R,\ i = d(r) \tag{4.9}$$

$$\sum_{k\in K}\sum_{(i,j)\in A} y_{ijkr} - \sum_{k\in K}\sum_{(j,i)\in A} y_{jikr} = 0 \qquad \forall r \in R,\ \forall i \in T \tag{4.10}$$

$$\sum_{(i,j)\in A} y_{ijkr} - \sum_{(j,i)\in A} y_{jikr} = 0 \qquad \forall k \in K,\ \forall r \in R,\ \forall i \in N \setminus (T \cup \{o(r), d(r)\}) \tag{4.11}$$

$$\sum_{(j,i)\in A} y_{jikr} + \sum_{(i,j)\in A} y_{ijlr} \leq s_{jrkl} + 1 \qquad \forall r \in R,\ \forall i \in T,\ \forall k \in K,\ \forall l \in K,\ k \neq l \tag{4.12}$$

$$t^{\text{arr}}_{jk} - t^{\text{dep}}_{jl} \leq M(1 - s_{jrkl}) \qquad \forall r \in R,\ \forall j \in T,\ \forall k \in K,\ \forall l \in K,\ k \neq l \tag{4.13}$$

$$\sum_{(i,j)\in A} x_{ij\tilde{k}} \leq \alpha_{\tilde{k}} \qquad \forall \tilde{k} \in \tilde{K} \tag{4.14}$$

$$\sum_{(i,j)\in A} c_{ij}x_{ij\tilde{k}} \leq \beta_{\tilde{k}}c_{o(\tilde{k})d(\tilde{k})} \qquad \forall \tilde{k} \in \tilde{K} \tag{4.15}$$

$$\sum_{(i,j)\in A} \tau_{ij}x_{ij\bar{k}} \leq \gamma_{\bar{k}} \qquad \forall \bar{k} \in \bar{K} \tag{4.16}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall k \in K \tag{4.17}$$

$$y_{ijkr} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall k \in K,\ \forall r \in R \tag{4.18}$$

$$s_{jrkl} \in \{0,1\} \qquad \forall j \in N,\ \forall r \in R,\ \forall k \in K,\ \forall l \in K \tag{4.19}$$

$$t^{\text{dep}}_{jk} \in \mathbb{R}^+_0 \qquad \forall j \in N,\ \forall k \in K \tag{4.20}$$

$$t^{\text{arr}}_{jk} \in \mathbb{R}^+_0 \qquad \forall j \in N,\ \forall k \in K \tag{4.21}$$

The objective function (4.1) quantifies the costs incurred in fulfilling all requests. The costs consist of three terms. The first term quantifies the costs incurred by RDs. The second term quantifies the detour costs and the compensation per stop for ODs. The third term reduces the costs by the length of the route of the OD originally planned and by one stop, because the ODs are only compensated according to the length of their respective detour and the number of additional stops. Therefore, the last two terms reflect the overall compensation for ODs. Note that ODs can go directly to their destination without affecting the objective, because the second and third term cancel each other out.

Constraints (4.2) - (4.6) model the vehicle flow, whereas Constraints (4.8) - (4.13) model the request flow, and Constraints (4.7) connect both flows. The following passage describes these constraints in detail. Constraints (4.2) and (4.3) ensure that each vehicle starts at most one route from its origin and ends at its destination depot, if it has left its origin depot. Constraints (4.4) conserve vehicle flows. If a vehicle k uses arc (i, j), Constraints (4.5) ensure that the arrival time at node j is greater than the departure time at node i plus the traveling time τ_{ij} from node i to j. Constraints (4.6) ensure that vehicle k does not depart at node j before it has arrived at the same node. Constraints (4.5) and (4.6) prevent subtours. Constraints (4.7) guarantee the connection of the vehicle flow and the request flow. Constraints (4.8) and (4.9) ensure that requests are picked up and delivered. Constraints (4.10) conserve request flow at TPs where requests have to be transported by any one vehicle. Constraints (4.11) conserve request flow at every other node. In contrast to TPs, the same vehicle bringing the request to that node has to leave with that exact same request. Constraints (4.12) enforce s_{jrkl} to 1 if there is a transfer between vehicle k and l. Constraints (4.13) ensure that the vehicle k transferring load to vehicle l arrives before vehicle l departs. Constraints (4.14) restrict the number of stops of ODs. Constraints (4.15) limit the length of the detour of ODs, Constraints (4.16) the duration of the RDs routes. Constraints (4.17), (4.18) and (4.19) define the binary decision variables. Constraints (4.20) and (4.21) define the domains of the continuous variables.

The model PDPTOD contains the traveling salesman problem as a special case that is known to be $\mathcal{NP}$-hard. Therefore, the PDPTOD is $\mathcal{NP}$-hard as well. The MIP can only be solved for small instances with a commercial MIP solver, as will be shown in Section 4.6. Consequently, we suggest developing heuristic solution approaches.

4.5 Solution Approach

We propose an adaptive large neighborhood search (ALNS) embedded in a simulated annealing framework for the solution of the PDPTOD. As mentioned before, the ALNS has been applied to many variants of the VRP, including the PDPTW (Ropke and Pisinger 2006) and the PDPT (Masson et al. 2013), which is closely related to our problem. Compared to the ALNS of Masson et al. (2013), we do not use operators that directly remove transshipment points (TPs) and therefore requests routed through this TP. Instead, TPs are only removed from a solution if no request is served via this TP. Our ALNS uses no additional heuristic to select a subset of TPs during insertion. The two insertion operators working with TPs may evaluate either all TPs or just one TP at random.

Furthermore, operators are adapted to capture and utilize the characteristics of occasional drivers (ODs). The proposed ALNS differs from existing approaches in three further major respects: (1) we suggest using combined operators, i.e., different operators could be used during one iteration, (2) the number of requests is sampled from a binomial and not from a uniform distribution, and (3) the initial temperature of the simulated annealing approach is obtained via a sampling procedure. In the following section, we give a high level introduction to the ALNS scheme. Next, we describe the destroy and repair operators that are specially designed for the PDPTOD in detail.

4.5.1 Adaptive large neighborhood search

Figure 4.2 depicts the general scheme of the ALNS proposed. The ALNS extends the large neighborhood search by choosing each destroy and repair operator depending on its past performance during the search. The initial solution required is obtained by randomly applying repair operators until every request is served, i.e., until the solution is complete and feasible. We use a simulated annealing acceptance criterion to diversify the search. The probability of accepting a deteriorating solution depends on the difference in costs of the candidate solution $f(Solution)$ and the cost of the best solution obtained so far $f(BestSolution)$, as well as the current temperature $Temp$. The temperature $Temp$ is used as stopping criterion (line 4 of Figure 4.2) and is determined for every instance with $Temp = -\frac{\overline{\Delta E}}{ln(\chi_0)}$ using the formula from (Johnson et al. 1989) cited in (Ben-Ameur 2004), where $\overline{\Delta E}$ is an estimation of the cost increase of strictly positive transitions and χ_0 a parameter expressing the probability of accepting a deteriorating solution. We execute n_0 iterations of the ALNS in order to generate the transitions. The temperature is reduced by $CoolRate$ after each iteration (line 14). As long as the temperature is above the minimum temperature, $TempLow$ another iteration is executed. At the end of every iteration, the weights are updated according to the performance of the operator. The score of an operator is increased by σ_1 if the operator was involved in producing a new global best solution, σ_2, if a new solution is found that has lower costs than the current solution, or σ_3, if the new solution has higher costs but is accepted. The probability of choosing an operator depends on the scores obtained during the search and is calculated in the same way as in Ropke and Pisinger (2006).

```
1  CurrentSolution ← InitialSolution
2  BestSolution ← CurrentSolution
3  Temp ← GetInitialTemp()
4  while Temp > TempLow do
5  |   Choose Repair and Destroy Operator
6  |   Solution ← Repair(Destroy(CurrentSolution))
7  |   if f(Solution) < f(BestSolution) then
8  |   |   CurrentSolution ← Solution
9  |   |   BestSolution ← CurrentSolution
10 |   else if accept(f(Solution), f(BestSolution), Temp) then
11 |   |   CurrentSolution ← Solution
12 |   Update Operator Weights
13 |   Temp ← CoolRate · Temp
```

Figure 4.2: ALNS algorithm in simulated annealing framework

In the following Subsections 4.5.2 and 4.5.3, we give a brief overview of the operators implemented. The first three destroy operators as well as the first three repair operators are adapted from Ropke and Pisinger (2006) to match our problem setting. If there are no TPs and no ODs, the problem is reduced to a classical PDP and can be solved while using only the first three destroy and repair operators. If there are additionally ODs available, but no TPs, the scenario is denoted as Pickup and Delivery Problem with Occasional Drivers (PDPOD) and can be solved without both insertion operators for TPs. Table 4.3 presents an overview of the operators used for solving the three models, i.e., PDP, PDPOD and PDPTOD.

Table 4.3: Overview of operators used

Operator	**ALNS-PDP**	**ALNS-PDPOD**	**ALNS-PDPTOD**
Random removal	✓	✓	✓
Worst removal	✓	✓	✓
Shaw removal	✓	✓	✓
Least efficient removal		✓	✓
Combined removal	✓	✓	✓
Random insertion	✓	✓	✓
Best insertion	✓	✓	✓
Regret-3 insertion	✓	✓	✓
Best-with-random TP insertion			✓
Best-with-best TP insertion			✓
Combined insertion	✓	✓	✓

4.5.2 Destroy operators

We use five destroy operators that remove q requests (pickup and delivery nodes) from the current solution. The number of requests q to be removed is determined in every iteration and sampled from a binomial distribution with sample size $|R|$ and probability p^{binom}. The idea is to introduce a bias for lower q, but still maintain the possibility of removing a high number of requests. Uniformly distributed q usually focuses on a narrow range and therefore limits the diversification.

Random removal The random removal operator is the simplest and fastest one. It randomly removes q requests from the solution. If the pickup and delivery of a request are served by two vehicles (i.e., a TP was used), it will be checked whether the TP is used by other requests on the same two routes. If the TP is not used any more, it will be removed from both routes. The same check is implemented for every other destroy operator.

Worst removal The worst removal operator iterates across all routes of ODs and regular drivers (RDs) to find the request that increases the cost the most and removes this request.

Shaw removal The shaw removal operator removes requests that are similar to each other. The similarity $Rel(r_1, r_2)$ is measured by the distance D of pickup $o(r_1)$ to $o(r_2)$ and D of delivery $d(r_1)$ to $d(r_2)$ and the number of common ODs that are able to serve both requests. The reasoning behind this is that these requests can potentially be handled by the same OD, leading to reduced detours. The lower the $Rel(r_1, r_2)$, the more related are requests r_1 and r_2.

$$Rel(r_1, r_2) = D_{o(r_1),o(r_2)} + D_{d(r_1),d(r_2)} + (1 - \frac{|\tilde{K}_{r_1} \cap \tilde{K}_{r_2}|}{\min(|\tilde{K}_{r_1}|, |\tilde{K}_{r_2}|)}) \tag{4.22}$$

The shaw removal operator chooses the first request to be removed with the random removal operator. The following requests are chosen from a sorted array of all remaining requests, whereas the array R^{Re} is sorted by increasing $Rel(r_1, r_2)$. $\text{uniform}(0,1)^{p^{\text{Shaw}}}|R^{\text{Re}}|$ determines the index of the request to be removed from the array. $p^{\text{Shaw}} \geq 1$ introduces randomness in selecting the requests.

Least efficient removal It is only worthwhile using RDs and ODs when they fulfill requests in an efficient way, i.e., without long detours. The least efficient removal operator removes a request from the route with the lowest efficiency. The cost per request, $cost_r$ is used as proxy for efficiency. It measures the relative cost by dividing the total cost, $cost_k$ of a route k by the number of requests $|R_k|$ served via this specific route. R_k defines the requests served via route k. The higher the cost per request, the lower the efficiency.

$$cost_r = \frac{cost_k}{|R_k|} \qquad \forall r \in R_k,\ k \in K \tag{4.23}$$

Combined removal The combined removal operator removes a request by randomly applying one of the aforementioned operators. The probability depends on the operator weights and is selected by a roulette wheel selection approach. The operator is chosen for every request and therefore may be different for requests in the same iteration. The combined removal operator fosters diversification.

4.5.3 Repair operators

Repair operators insert requests that have previously been removed into the solution. We use six operators to build a new solution, three of which are relatively standard and do not use TPs. The fourth and fifth operator ensure the use of TPs. In some cases these two operators are not able to insert a request while using a TP because for example no OD is able to serve the request without exceeding the maximum detour they are willing to make. If the request cannot be inserted using a TP the request is inserted via the best insertion operator.

Random insertion The random insertion operator inserts a request randomly into the current solution without using a TP. This means the pickup and delivery of one request has to be served by exactly one vehicle. The operator may use both types of drivers, RDs or ODs. The route in which the request is to be inserted is chosen randomly from the set of possible drivers. The set of possible drivers contains only RDs and ODs for which the request is in reach, i.e., pickup and delivery can be served without violating the detour constraint. This set is calculated once and reduces the runtime that would otherwise be wasted on fruitless attempts assigning requests to drivers who are for sure unable to serve these requests. This set becomes considerably smaller as β_k decreases.

Best insertion Instead of inserting the request randomly, the best insertion operator determines which route and which position increase the costs least. The operator iterates across the set of candidate RDs and ODs and chooses the driver who can accommodate the request with the least additional effort.

Regret-3 insertion The operator shows some foresight. It calculates the cost difference between the best vs. second best and second best vs. third best route in which the request can be inserted. The operator iterates across all requests to be inserted and inserts the request with the highest regret value. Before calculating the actual regret values, requests that can be served by fewer than three routes because of the detour constraint are inserted in order of ascending number of potential routes. This means requests with only one potential route are inserted first followed by requests with two potential routes, and so on.

Best-with-random TP insertion This operator enforces the use of a random TP when inserting the request. Figure 4.3 depicts the procedure. The procedure calculates the costs for all potential pickup and delivery routes when using a random TP. The pickup route is the route where the pickup will be inserted, the delivery route the route where the delivery of the same request will be inserted. The TP can either be already in the route or has to be inserted. In the second case, it is inserted at its least costly position in this route. If the insertion of the TP was successful, the algorithm proceeds and inserts the pickup before the TP. Again, if the insertion of the pickup was successful, the algorithm will succeed, otherwise it will try the next pickup route. The insertion of the delivery with transshipment is performed in a similar manner. The only difference lies in the position where the delivery has to be inserted. The delivery is inserted after the TP. For all potential (pickup route) - (TP) - (delivery route) combinations the combination with the lowest cost is determined and the request is inserted accordingly.

Best-with-best TP insertion The second-last repair operator is similar to the best-with-random TP insertion operator. They differ in the fact that the best-with-best TP insertion operator iterates across all possible TPs and uses the (pickup route) – (TP) – (delivery route) combination with the lowest cost. The benefit is the choice of a TP that fits into the routes by means of short detours. The obvious drawback is the high computation time.

Combined insertion The combined insertion operator inserts a request by randomly applying one of the aforementioned operators. The operator is selected in the same fashion as the combined removal operator. It can be worthwhile inserting just some of the requests with operators that force the use of one TP, and the rest without transshipments.

```
Require: Request r with pickup o(r) and delivery d(r)
Choose random t
for pickup route ∈ potential pickup routes do
    CostPickupRoute ← f(pickuproute)
    if t not in pickup route then
        insert t in pickup route at best position
    if t in pickup route then
        insert o(r) in pickup route at best position before t
    if o(r) in pickup route then
        for delivery route ∈ potential delivery routes do
            CostDeliveryRoute ← f(deliveryroute)
            if pickup route ≠ delivery route then
                if t not in delivery route then
                    insert t in delivery route at best position
                if t in delivery route then
                    insert d(r) in delivery route at best position after t
                    if d(r) in delivery route then
                        Cost ← f(pickup route)+f(delivery route)−CostPickupRoute−CostDeliveryRoute
Determine (pickup route) – (TP) – (delivery route) combination with lowest Cost
Insert request accordingly
```

Figure 4.3: Best-with-random TP insertion operator

4.6 Computational Experiments

In this section we introduce the instances used, briefly show how we determined the parameters, evaluate the performance of the ALNS and lastly provide some managerial insights by means of a sensitivity analysis. All experiments were conducted on an AMD Ryzen 7 2700X CPU with eight cores and 16 GB of RAM. We used Gurobi 8.1 as exact solver and LKH-3 (LKH-3 can be downloaded from `http://akira.ruc.dk/~keld/research/LKH-3/`) as a benchmark heuristic for the larger PDP instances. The ALNS was coded in C++.

4.6.1 Instance generation

The pickup and delivery problem with transshipments and occasional drivers (PDPTOD) is a new problem and therefore no benchmark instances exist. We used Solomon instances (Solomon 1987) and Li & Lim instances (Li and Lim 2003) to generate appropriate instances. All pickups originate from a central depot (VRP-like) for Solomon instances, and every request has a unique origin (PDP-like) for Li & Lim instances. We only used the coordinates for the requests from these instances, as we do not consider capacities or time windows. We randomly generated the origins and destinations of occasional drivers (ODs) in an area with a lower left corner $[0.9 \cdot \min(x_r), 0.9 \cdot \min(y_r)]$ and an upper right corner $[1.1 \cdot \max(x_r), 1.1 \cdot \max(y_r)]$. Note that x_r denotes the x coordinate of requests and y_r the y coordinate. Four transshipment points (TPs) are generated in the same manner for the case where TPs are randomly distributed in the relevant distribution area (scenario TR). Scenario T4 assumes four TPs at the locations specified in Table 4.4, which locates the TPs according to the requests. Scenario TR and T4 resemble the model PDPTOD. Scenario T0 goes without any TPs and therefore resembles the model without transshipments, i.e., pickup and delivery problem with occasional drivers (PDPOD).

Table 4.4: Regions and locations of transshipment points (TPs) in scenario T4

Region	Location
North	$[\min(x_r) + 0.5 \cdot \text{range}(x_r), \min(y_r) + 0.75 \cdot \text{range}(y_r)]$
East	$[\min(x_r) + 0.75 \cdot \text{range}(x_r), \min(y_r) + 0.5 \cdot \text{range}(y_r)]$
South	$[\min(x_r) + 0.5 \cdot \text{range}(x_r), \min(y_r) + 0.25 \cdot \text{range}(y_r)]$
West	$[\min(x_r) + 0.25 \cdot \text{range}(x_r), \min(y_r) + 0.5 \cdot \text{range}(y_r)]$

There are 12 request scenarios, each with three different TP scenarios and 30 different OD distributions. As naming convention we propose *Request Distribution-R-T-OD-n*. For example the instance *R101-R25-TR-OD25-1* assumes 25 requests distributed according to R101, randomly distributed TPs and 25 ODs with randomly distributed origin and destination. *R101-R25-T4-OD25-1* is the same instance except for the locations of TPs. This leads to 810 VRP-like instances and 270 PDP-like instances. Table 4.5 presents an overview of all instances tested.

Table 4.5: Overview of test instances

Request Distribution	#Requests	TP Scenario	#ODs
VRP-like			
R101	25	T0, TR, T4	25
	50	T0, TR, T4	50
	100	T0, TR, T4	100
RC101	25	T0, TR, T4	25
	50	T0, TR, T4	50
	100	T0, TR, T4	100
C101	25	T0, TR, T4	25
	50	T0, TR, T4	50
	100	T0, TR, T4	100
PDP-like			
lr201	51	T0, TR, T4	50
lrc201	51	T0, TR, T4	50
lc201	51	T0, TR, T4	50

4.6.2 Parameter tuning and analyses of operators

Parameters are tuned using the first instance of the TR, T4 and T0 scenario of R101, RC101, C101 with 100 requests and lr201, lrc201, lc201 with 51 requests (*R101-R100-TR-OD100-1, R101-R-T4-OD100-1,..., lc201-R51-T4-OD50-1*), resulting in 12 tuning instances for the ALNS procedure. Every instance is solved three times. We tune only the parameters that have shown the most impact on the algorithm performance found in preliminary experiments, i.e., χ_0, $TempLow$, $coolRate$, and p^{binom}. We set default values to the parameters similar to values found in literature or use values that seemed to deliver the best performance in preliminary experiments. We then fix all parameters except for one that is altered in a specific range. Finally, we decide on a set of parameters that promises a fair compromise of solution quality against runtime. Table 4.6 summarizes the parameters used by the ALNS procedure independent of the underlying problem situation, whether PDPOD or PDPTOD.

Table 4.7 shows the performance of the ALNS when removing one operator and using all the

Table 4.6: Parameters and values for ALNS

Parameter	Default value	Range tested	Chosen value
n_0	400	-	400
χ_0	0.25	[0.05, 0.10, ..., 0.90, 0.95]	0.35
$TempLow$	1	[2, 1, 0.5, 0.2, 0.1, 0.05, 0.01, 0.005]	0.5
$coolRate$	0.9996	[0.999, 0.9991, ..., 0.9999]	0.9998
p^{binom}	0.35	[0.05, 0.10, ..., 0.55, 0.60]	0.35
p^{shaw}	6	-	6
W_i	1	-	1
σ_1	30	-	30
σ_2	1	-	1
σ_3	10	-	10
$Rate$	0.1	-	0.1

remaining operators to solve PDPTOD. Column "Deviation" shows the deviation of the solution quality compared to the solutions obtained by ALNS with all operators. We used the same instances as for parameter tuning. We found that every operator contributes to the overall solution quality. Shaw removal, best insertion and best-with-best TP insertion operator are especially powerful operators that should not be left out. Even the random operators seem to increase the solution quality by introducing some diversification, which ultimately helps finding better solutions.

Table 4.7: Performance without specific operators

Without Operator	Deviation [%]
Random removal	+5.61
Worst removal	+4.46
Shaw removal	+8.15
Least efficient removal	+2.90
Combined removal	+5.35
Random insertion	+3.66
Best insertion	+14.88
Regret-3 insertion	+4.46
Best-with-random TP insertion	+3.76
Best-with-best TP insertion	+12.14
Combined insertion	+6.70

4.6.3 Performance evaluation

The performance of the ALNS is evaluated by comparing its results against the exact solution obtained by solving model PDPTOD with Gurobi. As Gurobi only solves small instances within reasonable times, we evaluate the performance of the ALNS against the heuristic solution of the LKH-3 for larger instances with up to 100 requests. Note that LKH-3 is only able to solve the classical PDP without ODs and without TPs, i.e., $|T| = 0$, which is a special case of model PDPTOD. We denote this case as a "baseline scenario." Lastly, the solutions of instances with ODs and TPs are compared against the previously solved baseline scenario, i.e., the case without ODs and without TPs.

4.6.3.1 Benchmark against gurobi

Twelve small instances with $|R| \leq 10$, $|T| = 4$, $|\tilde{K}| = 5$, $\alpha = 12$, $\beta = 2.0$, $p^{\text{stop}} = 0$, $p^{\text{detour}} = 1.0$ are solved with Gurobi and the ALNS. The time limit for Gurobi was set to 7200s. Gurobi and the ALNS are run only once. Table 4.8 presents the results. Note that the column "Gap" represents

Table 4.8: Results Gurobi vs. ALNS solution for instances with $|R| \leq 10$, $|T| = 4$, $|\tilde{K}| = 5$

Instances	**Gurobi**			**ALNS**			
	Cost	Runtime [s]	Gap [%]	Cost	Runtime [s]	Gap[%]	TP used
R101-R8-RT-OD5	130	682	0	130	<1	0	y
RC101-R8-RT-OD5	59	7200	12	59	<1	12	y
C101-R8-RT-OD5	49	7200	57	36	<1	41	y
lr201-R8-RT-OD5	115	10	0	115	<1	0	y
lrc201-R8-RT-OD5	162	7200	29	162	<1	29	n
lc201-R8-RT-OD5	229	2529	0	229	<1	0	n
R101-R10-RT-OD5	173	579	0	173	<1	0	n
RC101-R10-RT-OD5	124	7200	59	86	<1	41	y
C101-R10-RT-OD5	58	7200	52	55	<1	49	y
lr201-R10-RT-OD5	130	40	0	130	<1	0	y
lrc201-R10-RT-OD5	-	7200	-	171	<1	-	n
lc201-R10-RT-OD5	154	1113	0	155	<1	1	n

the MIP Gap, i.e., the difference vs. the current lower bound. Gurobi finds the optimal solution ($Gap = 0$) within the time limit for six instances. For five instances it achieves a feasible integer solution, it fails only for one instance. The ALNS achieves the same solutions for five instances. For three instances it finds a better solution within the given time limit of the ALNS procedure. The ALNS achieves slightly worse results only for one instance. The ALNS needs less than 1s for each instance, while Gurobi needs 4013s on average. The ALNS achieves comparable results for small instances. Unfortunately, it is not possible to compare the performance against Gurobi for larger instances because the runtime becomes prohibitive. However, we analyze the quality of the ALNS procedure by comparing the solutions achieved against the LKH-3 procedure, assuming the baseline scenario where ODs and TPs are not present, i.e., the PDP modeling approach.

4.6.3.2 Benchmark against LKH-3

This section compares the results of the ALNS approach with the LKH-3 approach, assuming scenarios without ODs and TPs. The LKH-3 and the ALNS are both run three times, the best solution found, and the average runtime is reported in Table 4.9. The difference in terms of costs is reported in column Gap [%], where $Gap = \dfrac{Cost_{\text{ALNS}} - Cost_{\text{LKH}-3}}{Cost_{\text{LKH}-3}}$.

Both heuristics achieve the same results for ten out of twelve instances. In one case the ALNS produces slightly worse results, in one case slightly better. The LKH-3 needs 10s on average, the ALNS only 4s. The ALNS produces high-quality solutions for cases with up to 100 requests, assuming no TPs and no ODs, within a short computation time. Note that the results for this baseline scenario serve as an upper bound for the following experiments. The objective value cannot get worse if TPs or ODs are available because it is always possible to only use regular drivers (RDs) without deploying ODs and without fulfilling requests via TPs.

Table 4.9: Results of LKH-3 vs. ALNS for instances with $|T| = 0$, $|\tilde{K}| = 0$

Instance	**LKH-3**		**ALNS**		
	Cost	Runtime [s]	Cost	Runtime [s]	Gap [%]
R101-R25-T0-OD0	311	<1	311	<1	0
RC101-R25-T0-OD0	226	<1	226	<1	0
C101-R25-T0-OD0	133	<1	133	<1	0
R101-R50-T0-OD0	455	<1	455	3	0
RC101-R50-T0-OD0	370	<1	370	2	0
C101-R50-T0-OD0	242	<1	242	2	0
lr201-R51-T0-OD0	709	35	723	4	2.0
lrc201-R51-T0-OD0	743	11	737	4	-0.8
lc201-R51-T0-OD0	543	<1	543	2	0
R101-R100-T0-OD0	629	17	629	10	0
RC101-R100-T0-OD0	637	29	637	10	0
C101-R100-T0-OD0	501	17	501	8	0

4.6.3.3 Comparison with baseline scenario

Table 4.10 shows the results obtained by solving instances with up to 100 requests, 100 ODs, $\alpha = 12$, $\beta = 2.0$, $p^{\text{stop}} = 0$ and $p^{\text{detour}} = 1.0$. Cost [%] is a standardized value. It is calculated by dividing the average cost, $cost^{\text{av}}$, by the best known cost of the baseline scenario, $cost^{\text{base}}$, without TPs and without ODs, as denoted in Table 4.9.

Table 4.10: Results achieved by approach PDPOD (T0) vs. approach PDPTOD (TR, T4) for instances with $|R|, |\tilde{K}| \leq 100$

		Cost [%]			**Hypotheses**[1]			**Average runtime [s]**		
		T0	TR	T4	T0=TR	T0=T4	TR=T4	T0	TR	T4
R101	R25, OD25	64.60[2]	54.38	50.75	***	***	**	1.10	3.19	3.84
	R50, OD50	55.86	51.64	46.92	***	***	***	3.63	14.13	19.14
	R100, OD100	42.22	41.32	39.01	o	**	**	11.85	115.63	161.02
RC101	R25, OD25	81.59	61.64	54.53	***	***	***	1.09	3.06	3.37
	R50, OD50	67.80	56.65	51.63	***	***	**	3.53	15.49	18.57
	R100, OD100	50.82	49.68	46.79	o	***	*	13.25	113.55	168.93
C101	R25, OD25	98.02	82.31	75.76	***	***	***	0.46	2.18	2.35
	R50, OD50	80.29	68.35	59.82	***	***	***	3.52	11.49	15.10
	R100, OD100	48.32	49.56	44.25	o	***	**	14.09	113.97	145.89

[1] Shows if the hypothesis can be rejected and the according p-value. ***: $p \leq 0.001$, **: $p \leq 0.01$, *: $p \leq 0.05$, o: $p > 0.05$

[2] Example calculation:

$$Cost\ [\%] = \frac{cost^{\text{av}}}{cost^{\text{base}}} \cdot 100 = 64.60 \quad \text{with}$$

$$cost^{\text{av}} = \frac{\sum_{i=1}^{30} f(\text{R101} - \text{R25} - \text{T0} - \text{OD25} - \text{i})}{30} = 200.91$$

$$cost^{\text{base}} = 311$$

A Wilcoxon test on the paired samples is executed to check the following three hypotheses.

- Column T0=TR: "Scenarios T0 and TR lead to the same solution quality"
- Column T0=T4: "Scenarios T0 and T4 lead to the same solution quality"
- Column TR=T4: "Scenarios TR and T4 lead to the same solution quality"

The PDPTOD modeling approach usually leads to solutions with significantly lower costs compared to the approach of neglecting TPs, i.e., the PDPOD modeling approach. We are only unable to show a significant difference in the solution quality of scenario T0 and TR for instances with 100 requests. In general, the costs achieved by approach PDPTOD (TR, T4) should not be higher than the costs achieved by approach PDPOD (T0) because TPs are only used if cost advantages can be realized. The ALNS procedure successfully achieves cost advantages in all but one case (C101-R100-TR-OD100). We can also reject the hypotheses that scenarios TR and T4 lead to the same solution quality, i.e., the evidence provided in Table 4.10 shows that a reasonable placement of TPs has the potential to decrease costs.

The ALNS procedure scales well with the instance size. Small instances with R25, OD25 are solved in few seconds. Larger instances are solved in less than 20 seconds, except for instances with R100, OD100 and TR or T4, which take considerably more time because of the increasing number of possible solutions when TPs are available.

4.6.4 Sensitivity analysis

Since crowdshipping is a relatively new delivery concept, it is important to understand how the characteristics of planning scenarios impact the economic advantages of this concept. The following investigation may therefore be of interest for companies who are experimenting with crowdshipping. We analyze the impact of the number and flexibility of ODs, the number and location of TPs and the compensation scheme for ODs on diverse dependent variables. Dependent variables are total costs, the share of ODs used and the share of requests served via TPs. We use randomly distributed and clustered VRP-like instances with $|R| = 50$ (R101-R50 and C101-R50) to illustrate these effects. Please note that because of space limitations we do not present the results for the additional instances mentioned in Table 4.5 in detail as long as these instances result in equivalent effects.

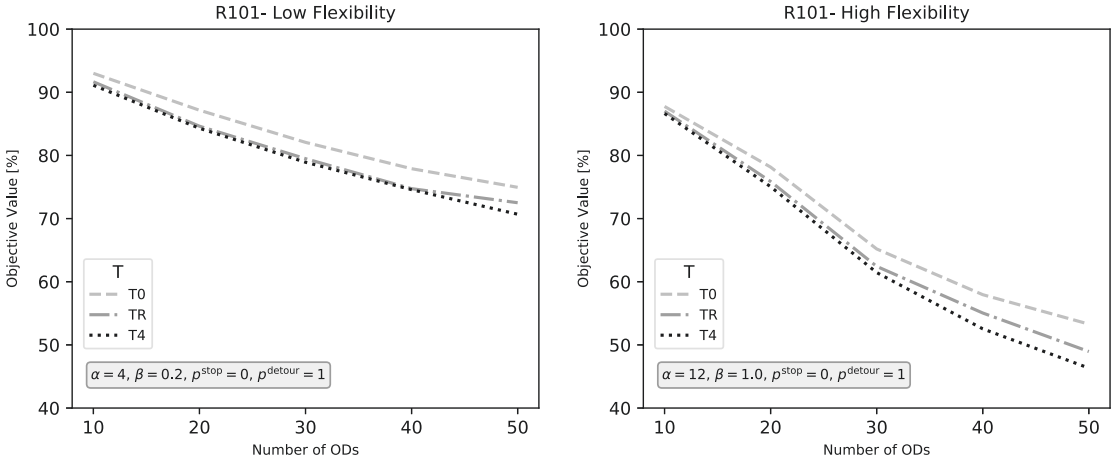
We assume three flexibility levels: low ($\alpha = 4, \beta = 0.2$), medium ($\alpha = 8, \beta = 0.6$) and high ($\alpha = 12, \beta = 1.0$). As a compensation scheme we assume $p^{\text{detour}} = 1$ and $p^{\text{stop}} = 0$, i.e., ODs receive the same compensation as RDs. In this case, costs are synonymous with the total distance traveled. The number of available ODs varies within the following range: $|\tilde{K}| = [10, 20, 30, 40, 50]$.

4.6.4.1 Analyzing total costs

Figures 4.4 and 4.5 show the impact of the number of ODs, the flexibility level of ODs and the transshipment scenario chosen on the objective value achieved. The horizontal and vertical axes denote the number of ODs and the standardized objective value [%], respectively. The objective value quantifies the percentage of the total costs achieved compared to the total costs in a scenario without any ODs. Three different lines (light-grey dashed line, grey dotdashed line, black dotted line) represent the transshipment scenario (T0, TR, T4).

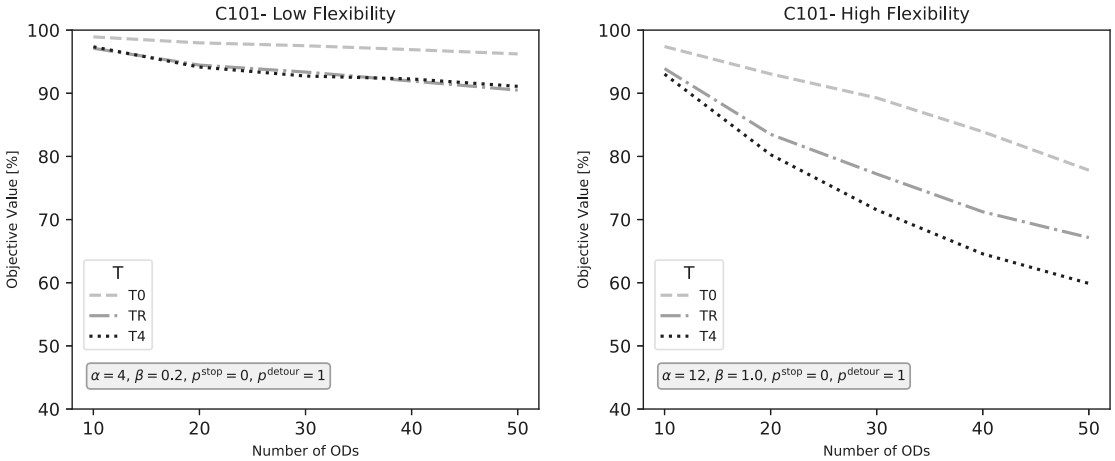
It is possible to achieve significant cost savings if more ODs are available. However, the cost savings strongly depend on the flexibility of ODs and how the requests are distributed. The cost savings per additional OD are higher if the requests are randomly distributed and the flexibility of the ODs is high (see Figure 4.4b). Minor cost effects can be observed on clustered instances with

less flexible ODs, especially if no transshipment points (T0) are available (see Figure 4.5a). In the event of clustered requests, the ODs should offer a certain degree of flexibility in order to be able to serve an entire cluster with a limited number of ODs, since an entire cluster can also be efficiently served by a single RD (see Figures 4.5a vs. 4.5b). Thus, if the flexibility of ODs is limited, it is more cost efficient to serve the entire cluster with only one RD and not with a combination of ODs and RDs. In addition, the impact of TPs on the objective value is greater for clustered instances (C101) than for random instances (R101). This becomes especially visible if the TPs are placed in a somewhat reasonable manner, e.g., scenario T4 (see black dotted line in Figure 4.5b). The average effect of TPs on the objective value is relatively small in our numerical experiments. This confirms the findings of Sampaio Oliveira et al. (2020). They found small positive cost effects if the driver shift length is long and the distance between pickup and delivery points is short. However, we observe additionally to their findings that the impact of TPs on the objective value strongly depends on the distribution of requests.



(a) Low Flexibility (b) High Flexibility

Figure 4.4: Objective value depending on the number of available ODs, R101-R50

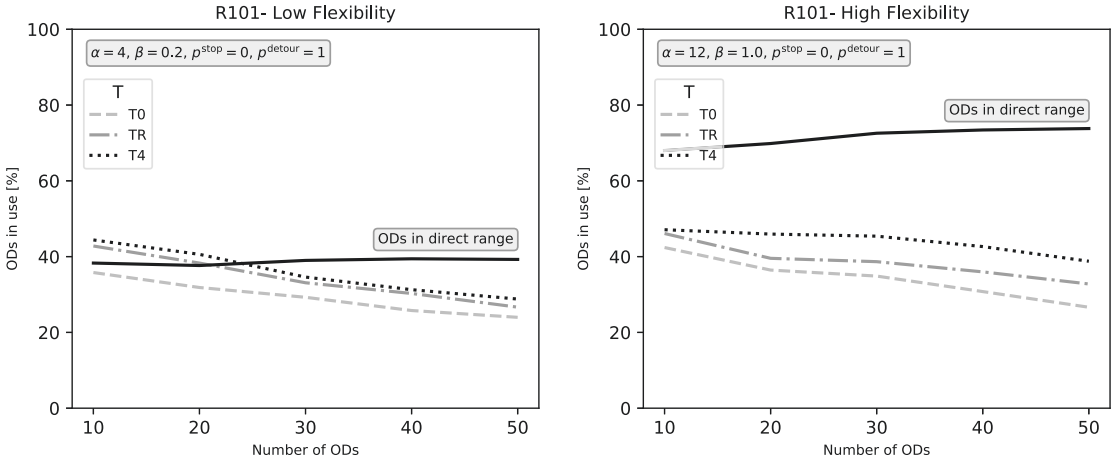


(a) Low Flexibility (b) High Flexibility

Figure 4.5: Objective value depending on the number of available ODs, C101-R50

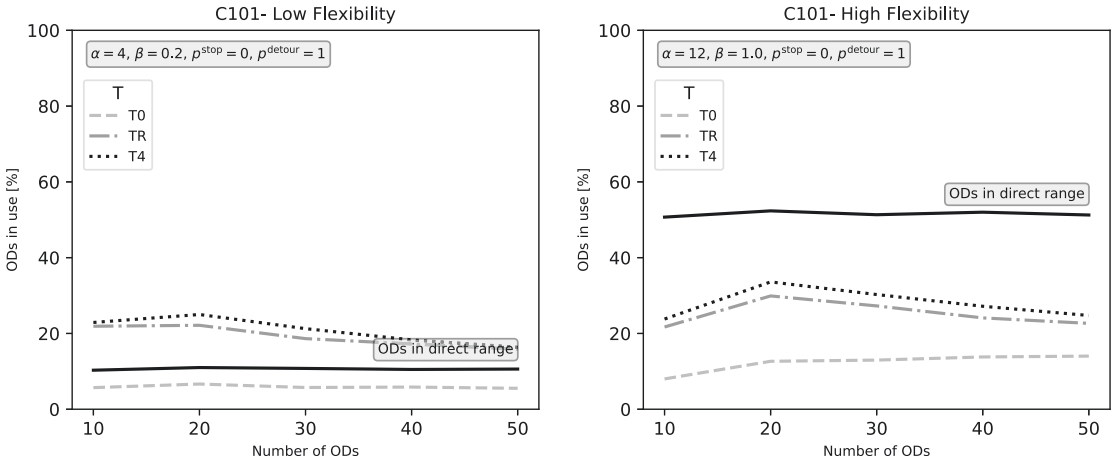
4.6.4.2 Analyzing the share of ODs used

We assume that the number of ODs used is important to achieve long-term stability. Low utilization may lead to OD dissatisfaction. If ODs do not receive tasks repeatedly, it is likely that they will no longer offer their services on the platform and henceforth fewer ODs will be available to fulfill the requests. We therefore analyze this issue in the present subsection. Figures 4.6 and 4.7 show the impact of the number of available ODs, the flexibility and the transshipment scenario on the share of the entire set of available ODs used to fulfill the requests. Note that the share of ODs used quantifies the average number of ODs deployed compared to the total number of available ODs.



(a) Low Flexibility (b) High Flexibility

Figure 4.6: Share of ODs used depending on the number of available ODs, R101-R50



(a) Low Flexibility (b) High Flexibility

Figure 4.7: Share of ODs used depending on the number of available ODs, C101-R50

Obviously, the deployment of an OD greatly depends on her/his willingness to undertake detours, i.e., her/his flexibility to make extra trips. Thus, an OD would never be deployed for a trip that is located outside her/his direct range. In the instances analyzed, only a limited share of ODs could directly serve a request. We visualize this share of ODs by adding a solid black line into the charts

of Figures 4.6 and 4.7. If a request is located outside the direct range of an OD, the OD can only be deployed during a fulfillment process if the request is served via a TP. This means that in cases without TPs (T0, light-grey dashed line), the share of ODs deployed cannot be higher than the average share of ODs who are in the direct range of a request (solid black line). Figures 4.6 and 4.7 also show that the share of ODs used greatly depends on the request distribution, but only slightly depends on the number of available ODs. This means that the absolute number of ODs used increases with the number of available ODs. Therefore it is not against the ODs' interests to have more ODs to compete with, in some cases it is even beneficial for the ODs.

Higher flexibility increases utilization. The platform provider should therefore ensure that the ODs are flexible, i.e., willing to deviate from the route originally planned, and to accept additional stops. The share of ODs used is higher for instances where requests are randomly distributed (see Figure 4.6). In these cases, many ODs get to serve only a small number of requests per OD. In contrast, for clustered instances the utilization of ODs is relatively low (see Figure 4.7) because clusters are better served with just a few ODs. Figure 4.8 shows an illustrative example where the upper left cluster – far away from the origin of requests – is served by one OD only, who – of course – offers high flexibility.

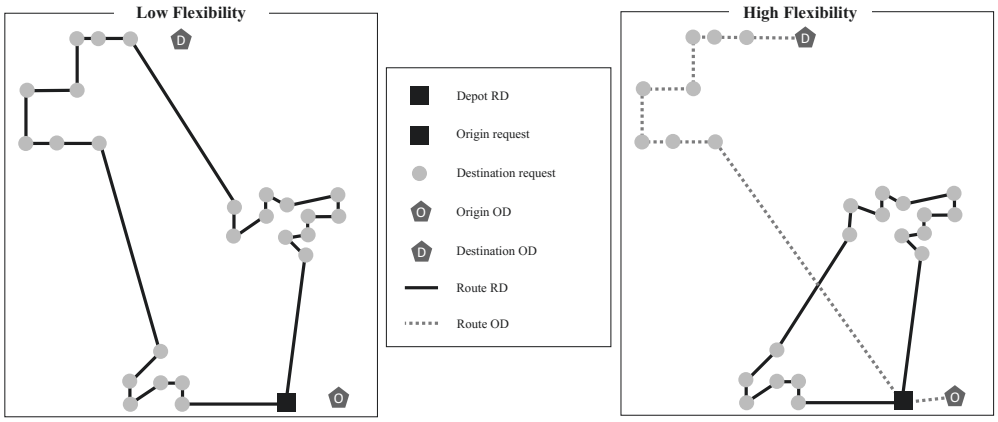


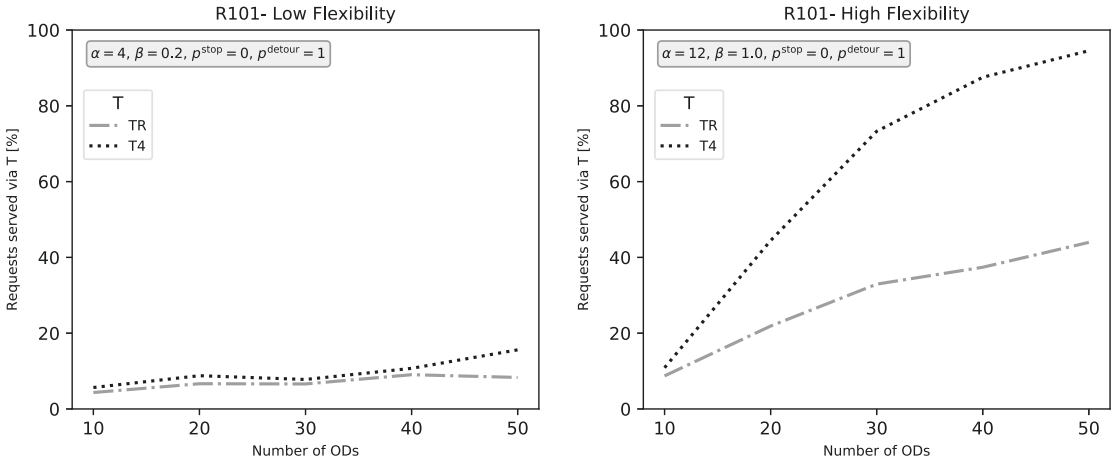
Figure 4.8: Illustrative example of a cluster served by only one OD

TPs improve utilization. ODs who were not able to serve any request without TPs because of the detour constraint now find themselves in the range of some requests delivered via TPs and can also be deployed. This effect becomes more pronounced as flexibility decreases. In some cases (e.g., Figure 4.7a), the number of ODs used is even higher than the average share of ODs located in the direct range of at least one request (solid black line vs. TR and T4 scenarios).

The results shown focus on VRP-like instances. The share of ODs deployed, however, is much higher for PDP-like instances, which are not presented here in detail. In those cases, the deployment of ODs only slightly depends on the flexibility offered by ODs, the availability of TPs and the total number of available ODs. A platform operating in PDP-like circumstances can therefore guarantee higher and more stable utilization of ODs than a platform operating in VRP-like circumstances.

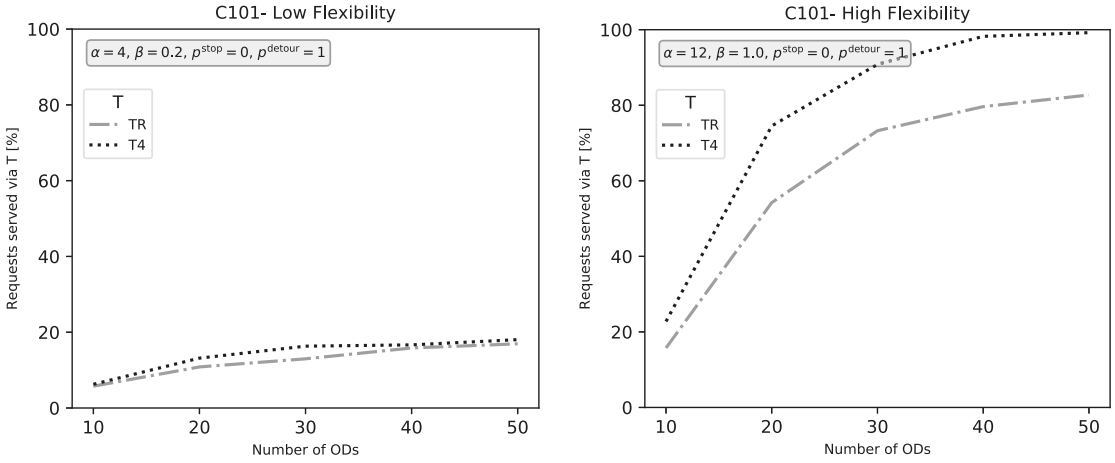
4.6.4.3 Analyzing the share of requests served via TPs

Figures 4.9 and 4.10 show the impact of the number of ODs, the flexibility level and the transshipment scenario on the share of requests served via TPs.



(a) Low Flexibility (b) High Flexibility

Figure 4.9: Share of requests served via TPs depending on the number of available ODs, R101-R50



(a) Low Flexibility (b) High Flexibility

Figure 4.10: Share of requests served via TPs depending on the number of available ODs, C101-R50

Note that the charts do not show results for the instances that exclude TPs, i.e., scenario T0. Obviously, in those cases no request can be fulfilled via TPs.

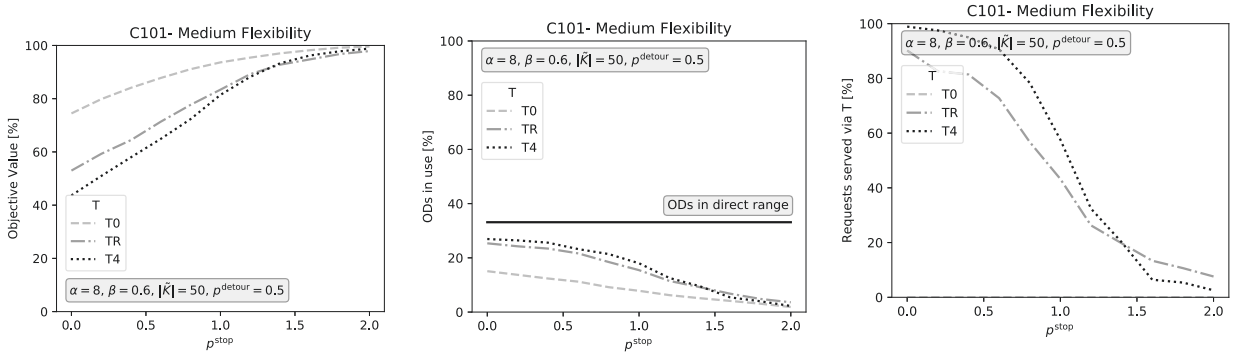
The share of requests served via transshipment points considerably depends on the flexibility level offered by the ODs, and greatly depends on where the TPs are located in the distribution area. In cases where ODs only offer low flexibility, TPs are rarely used (see Figures 4.9a and 4.10a). Nevertheless, the availability of TPs increases the deployment of ODs, as demonstrated in the previous subsection. However, in cases where ODs offer high flexibility, the utilization of TPs substantially increases with the increasing number of available ODs (see Figures 4.9b and 4.10b). The T4 scenario (black dotted line), in addition, reveals the relevance of placing TPs appropriately

in the distribution area. In this case the TPs are located in accordance with where requests occur. Thus, almost every request is served via a TP. The solution for clustered instances results in a two-stage distribution structure for 64% of the requests, on average, where the TPs (first stage) are served via RDs and via ODs in the second stage. ODs additionally take over the first stage of the transport for randomly distributed instances. TPs are at most used for 48% of requests in the case of PDP-like instances (randomly generated instances, scenario T4 with 50 OD), and ODs usually serve the first and second stage. A two-stage distribution structure is therefore not obvious for PDP-like instances. Please note that a distribution structure larger than two stages cannot occur since the ALNS procedure developed only allows using a maximum of one TP when fulfilling a delivery request.

Summarizing the observations above, it is highly important to locate the TPs in accordance with the demand arising, and not just randomly. In addition, a platform provider should preferably set up a two-stage structure if ODs offer high flexibility and clustered requests originate mostly from the depot.

4.6.4.4 Analyzing the compensation per additional stop

Figure 4.11 shows the impact of the compensation paid to ODs per additional stop, p^{stop}, on the objective value (4.11a), on the share of ODs used (4.11b) and on the share of requests served via TPs (4.11c). The C101 instance type with medium flexibility serves here to illustrate the effect, as it is very pronounced in this case. Nevertheless, the effects can also be observed in other instances, albeit in a less pronounced form. As expected, cost savings decrease as the compensation per stop



(a) Objective Value

(b) Proportion of ODs used

(c) Proportion of requests served via TPs

Figure 4.11: Impact of compensation rate p^{stop} on the results, C101-R50, medium flexibility

increases. The share of ODs used also decreases gradually with increasing p^{stop}. For both Figures 4.11a and 4.11b, it can be observed that there seems to be a critical compensation per stop, when the usage of TPs is no longer beneficial. This critical compensation can be determined from Figure 4.11c. The share of requests served via TPs for the T4 scenario drops from a stable level of around 90% if $p^{\text{stop}} < 0.7$ to under 20% if $p^{\text{stop}} > 1.4$.

4.6.4.5 Analyzing the influence of temporal flexibility

The model formulation principally assumes sufficiently high temporal flexibility of ODs such that delivery schedules that result from the associated task assignments to ODs can always be realized. However, drivers generally have limited temporal flexibility, e.g., commuters should arrive at their working place at a specific time and cannot leave their working place before the end of work. They cannot generally organize their private schedule in order to fulfill all tasks that might be realizable related to their spatial degree of freedom. The duration of an OD's route is not explicitly limited within our model formulation. The model formulation only limits the number of stops and the length of the detour for each individual OD's offer. This however implicitly limits the duration of the OD's routes generated. Nevertheless, the duration of a route can become unacceptably long if an OD has to wait at a TP for a parcel that is not yet available at their desired departure time.

Feasible OD schedules can only be assured for all instances if the modeling approach contains concrete time windows for all trips and the associated detours offered by an OD. This would however substantially increase model complexity and would also possibly cause some waiting times for ODs who may have to wait at TPs, as traveling times may be stochastic in reality. Nevertheless, we cope with this issue by letting ODs indicate whether their trip is planned before midday or after. ODs only deliver parcels to TPs but do not gather parcels there if they offer their services before midday. In contrast, ODs gather parcels from TPs but do not deliver to TPs if they offer their services after midday. The direct delivery of parcels is not affected by these assumptions. The commitments of the ODs guarantees that all parcels that are not directly delivered are available at TPs before or at midday at the latest. Table 4.11 shows the results obtained by solving instances, assuming half of the ODs are available before midday and the other half after midday.

Table 4.11: Results achieved by approach PDPOD (T0) vs. approach PDPTOD (TR, T4) for instances with (un)limited temporal flexibility

Instances		**Costs** [%]				
		T0	**TR**		**T4**	
			Limited	Unlimited	Limited	Unlimited
R101	R50, OD50	55.86	52.12	51.64	52.12	46.92
RC101	R50, OD50	67.80	66.63	56.65	63.99	51.63
C101	R50, OD50	80.29	72.78	68.35	70.32	59.82

The results are generated in the same manner as the results displayed in Table 4.10. In that case, the potential TPs' cost savings are obviously lower than when relaxing ODs' potential temporal constraints. Nevertheless, the savings are still significant depending on the type of instance. Instances *C101*, for example, allows for about 10% additional cost savings when TPs are available, i.e., 80.29% for scenario T0 and 70.32% for scenario T4 with limited temporal flexibility. The gap between the limited and unlimited TR scenarios is more pronounced for clustered instances (C, RC) because TPs are more frequently used compared to random scenarios (R), see Figures 4.9b and 4.10b.

4.6.4.6 Managerial insights

This section summarizes the findings of the sensitivity analysis and gives some advice on how to include crowdshipping in the last-mile delivery process. The use of ODs has the potential for cost savings independent of the request distribution (R, RC, C) or fulfillment structure (VRP-like vs. PDP-like), but is dependent on the flexibility offered by ODs. The platform provider should therefore create incentives for ODs that guarantee at least a moderate level of OD flexibility.

The benefits from TPs depend on the request distribution. If requests are evenly distributed in the area, savings will be high and the utilization of ODs will be acceptable, even without TPs. However, for clustered or semi-clustered (C, RC) request distributions, TPs reduce costs and increase the utilization of ODs, which would otherwise be rather low. The location of TPs is of considerable importance and should be carefully determined according to the distribution of requests. As soon as a critical mass of ODs is available, the platform provider should consider establishing a two-stage distribution system where RDs serve only TPs and ODs take over the last-mile delivery from TPs to the destination of requests. The fact that TPs potentially increase the number of ODs deployed is an especially valid reason for introducing TPs.

Customer requests fulfilled late or even unfulfilled lead to dissatisfied customers, which in the end also reduces the number of requests. Future initiatives to set up crowdshipping platforms should therefore carefully address the questions of how to attract a critical mass of ODs and how to guarantee a high service level in all circumstances. The analyses presented here reveal some valuable insights related to these questions. Above all, a business model that deploys a sufficient number of RDs alongside ODs increases the level of service, i.e., fulfills a high share of requests, thus maintaining high customer satisfaction, which is important to keep customers using the platform. In addition to this general delivery framework, offering fair compensation for ODs, as assumed in our analyses, is obviously a relevant factor attracting an adequate number of ODs. However, fair payment on a one-off basis is insufficient to keep ODs on the platform in the long term if ODs are only rarely deployed. TPs increase the deployment of ODs in all settings analyzed, but this in turn depends on the compensation per stop for ODs. As a result, the compensation per additional stop, p^{stop}, has to be carefully quantified in order to benefit from operating TPs. On the one hand, almost all delivery requests are assigned by the platform provider to RDs – for economic reasons – if the cost rate is set too high. In that case, TPs remain unused and ODs are also not deployed although they would be highly motivated to offer their services. On the other hand, ODs will most likely not participate if the cost rate is set too low. In that case, a compensation solely for the detour would generally be insufficient to motivate potential ODs.

Summarizing these thoughts, setting up appropriate TPs increases the utilization of ODs. The employment of RDs ensures a high customer satisfaction, so customers keep on using the platform. A crowdshipping business model operating TPs and employing RDs, defines suitable circumstances for analyzing and setting up an appropriate compensation scheme for ODs. Then, in a later phase of business activity, i.e., as soon as a critical mass of requests and ODs is available, the platform provider may further develop their business concept. For example, they may only use ODs without RDs, or apply the aforementioned two-stage distribution approach.

4.7 Conclusions and Future Areas of Research

Conclusions This article introduces a variant of the static pickup and delivery problem with transshipments and occasional drivers, termed PDPTOD. In this setting, occasional and regular drivers (ODs and RDs) may pickup parcels at the origin, deliver these directly, or transfer them to other drivers at dedicated transshipment points (TPs). The platform provider can choose between RDs and rather constrained ODs who are willing to deviate from the route they had originally planned and take on the fulfillment of one or more requests. We develop a novel MIP model (PDPTOD) that we use to solve small instances. In order to solve larger instances with up to 100 requests, we develop a heuristic solution approach based on an adaptive large neighborhood search (ALNS) for the optimization problem (PDPTOD) formulated. The ALNS procedure produces equivalent results for small instances, compared to the exact solution obtained by Gurobi, and for larger instances compared to results obtained by LKH heuristics. The ALNS scales well with the problem size. The numerical experiments based on known instances from literature extended by ODs and TPs show that the usage of both significantly reduces total costs. The cost savings potential highly depends on the assumed flexibility of ODs. The cost savings are only moderate if ODs are unwilling to make extra trips. In contrast, substantial cost savings are achievable if ODs are willing to make additional trips. In addition, setting up TPs reduces total costs as well. This occurs in both cases, i.e., where requests are distributed randomly in the delivery area, and where requests predominantly emerge in specific regions of the delivery area. The availability of TPs will certainly increase the utilization of ODs and will lead to a more equal distribution of workload between RDs and ODs. This is a very relevant aspect when setting up a crowdshipping platform. ODs are more likely to offer their services on a platform if they are consistently assigned requests to serve. To get the most out of setting up TPs, it is important to locate them in accordance with the demand arising, and not just randomly. The scenario where we positioned TPs in accordance with demand achieves superior results in almost all cases compared to the scenario where TPs are randomly distributed in the delivery area.

Future Areas of Research The suggested modeling and solution approach includes specific – possibly restricted – assumptions that offer avenues for future research.

(a) ODs are only restricted by spatial limitations; strict temporal restrictions are not included in our setting. Time windows for ODs are a realistic assumption and should be included in further research. The effect of the length of time windows on the critical mass can be examined, for example.

(b) The present research neglects delivery time windows, too. Approaches that assume customer-preferred delivery time windows or time windows with higher customer attendance rates will certainly increase the probability of successful deliveries.

(c) The timeline of the modeling approach assumes that all data are known when the planning takes place. Customer requests and offers from ODs may however arrive on a dynamic and/or

stochastic basis. Specialized modeling and solution approaches are required for those planning situations.

(d) The locations of the TPs in the scenario analyzed are determined via a simple heuristic. It can be assumed that a more elaborate model and solution procedure for planning the location of TPs will impact the results, and will also increase the use of ODs.

(e) A two-stage approach where RDs only serve the first stage and ODs the second stage, i.e., the very last mile to customers, could be a viable option for established platforms with many registered ODs, and therefore offers an additional avenue for further research.

(f) The compensation for ODs has to be determined carefully because it is probably the most important reason for ODs to participate in the long run. Studies dealing with fair compensation for ODs and determining appropriate types of compensation schemes would be of great value to further encourage crowdsourced logistics in the future.

(g) ODs are not obliged to accept the delivery of requests. Solutions may therefore not be robust if many ODs decline requests, or do not fulfill requests even if they have pledged to do so. Empirical research is needed to quantify the risk of unfulfilled requests and identify measures to motivate ODs to accept requests.

Bibliography

Agatz, N., Campbell, A., Fleischmann, M., Savelsbergh, M., 2011. Time slot management in attended home delivery. Transportation Science 45 (3), 435–449.

Amarouche, Y., Guibadj, R. N., Moukrim, A., 2018. A neighborhood search and set cover hybrid heuristic for the two-echelon vehicle routing problem. In: Borndörfer, R., Storandt, S. (Eds.), 18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018). Vol. 65 of OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 11:1–11:15.
URL `http://drops.dagstuhl.de/opus/volltexte/2018/9716`

Applegate, D., Bixby, R., Chvátal, V., 2007. The Traveling Salesman Problem. Princeton University Press.
URL `https://www.ebook.de/de/product/6393107/david_l_applegate_robert_e_bixby_vasek_chvatal_the_traveling_salesman_problem.html`

Applegate, D., Bixby, R., Chvátal, V., Cook, W., 2006. Concorde TSP solver.
URL `http://www.math.uwaterloo.ca/tsp/concorde/`

Archetti, C., Savelsbergh, M., Speranza, M. G., 2016. The vehicle routing problem with occasional drivers. European Journal of Operational Research 254 (2), 472–480.

Arnold, F., Sörensen, K., 2019. Knowledge-guided local search for the vehicle routing problem. Computers & Operations Research 105, 32–46.

Arnold, F., Sörensen, K., 2021. A progressive filtering heuristic for the location-routing problem and variants. Computers & Operations Research 129, 105166.

Arslan, A. M., Agatz, N., Kroon, L., Zuidwijk, R., 2018. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. Transportation Science 53 (1), 222–235.

Barreto, S., Ferreira, C., Paixão, J., Santos, B. S., 2007. Using clustering analysis in a capacitated location-routing problem. European Journal of Operational Research 179 (3), 968–977.

Belhaiza, S., Hansen, P., Laporte, G., 2014. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers & Operations Research 52, 269–281.

Belhaiza, S., M'Hallah, R., Ben Brahim, G., 2017. A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 1319–1326.

Belhaiza, S., M'Hallah, R., Brahim, G. B., Laporte, G., 2019. Three multi-start data-driven evolutionary heuristics for the vehicle routing problem with multiple time windows. Journal of Heuristics 25 (3), 485–515.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., Bengio, S., 2017. Neural combinatorial optimization with reinforcement learning.

Ben-Ameur, W., 2004. Computing the initial temperature of simulated annealing. Computational Optimization and Applications 29 (3), 369–385.

Breunig, U., Schmid, V., Hartl, R. F., Vidal, T., 2016. A large neighbourhood based heuristic for two-echelon routing problems. Computers & Operations Research 76, 208–225.

Buldeo Rai, H., Verlinde, S., Merckx, J., Macharis, C., 2017. Crowd logistics: an opportunity for more sustainable urban freight transport? European Transport Research Review 9 (3), 39.

Cardenas, I., Beckers, J., Vanelslander, T., Verhetsel, A., Dewulf, W., 2016. Spatial characteristics of failed and successful e-commerce deliveries in Belgian cities. In: Information Systems, Logistic and Supply Chain Conf.

Chao, I.-M., Golden, B. L., Wasil, E., 1993. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. American Journal of Mathematical and Management Sciences 13 (3-4), 371–406.

Chen, W., Mes, M., Schutten, M., 2017. Multi-hop driver-parcel matching problem with time windows. Flexible Services and Manufacturing Journal 30 (3), 517–553.

Contardo, C., Cordeau, J.-F., Gendron, B., 2013. A GRASP + ILP-based metaheuristic for the capacitated location-routing problem. Journal of Heuristics 20 (1), 1–38.

Contardo, C., Cordeau, J.-F., Gendron, B., 2014. An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. INFORMS Journal on Computing 26 (1), 88–102.

Cook, W., 2012. In pursuit of the traveling salesman : mathematics at the limits of computation. Princeton University Press, Princeton.

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

Cordeau, J.-F., Maischberger, M., 2012. A parallel iterated tabu search heuristic for vehicle routing problems. Computers & Operations Research 39 (9), 2033–2050.

Cortés, C. E., Matamala, M., Contardo, C., 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. European Journal of Operational Research 200 (3), 711–724.

Crainic, T. G., Mancini, S., Perboli, G., Tadei, R., 2011. Multi-start heuristics for the two-echelon vehicle routing problem. In: Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg, pp. 179–190.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., Song, L., 2018. Learning combinatorial optimization algorithms over graphs.

Dayarian, I., Savelsbergh, M., 2020. Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. Production and Operations Management 29 (9), 2153–2174.

de Jong, C., Kant, G., van Vlient, A., 1996. On finding minimal route duration in the vehicle routing problem with multiple time windows. Manuscript, Department of Computer Science, Utrecht University, Holland.

Devari, A., Nikolaev, A. G., He, Q., 2017. Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers. Transportation Research Part E: Logistics and Transportation Review 105, 105–122.

Doerrzapf, L., Berger, M., Breitfuss, G., Remele, E., 2016. Crowd Delivery als neues Lieferkonzept zur Stärkung des lokalen Marktplatzes. REAL CORP 2016 Proceedings, 197–206.

Drexl, M., 2012. Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. Transportation Science 46 (3), 297–316.

Drexl, M., 2013. Applications of the vehicle routing problem with trailers and transshipments. European Journal of Operational Research 227 (2), 275 – 283.

Drexl, M., Schneider, M., 2015. A survey of variants and extensions of the location-routing problem. European Journal of Operational Research 241 (2), 283–308.

Duhamel, C., Lacomme, P., Prins, C., Prodhon, C., 2010. A GRASP×ELS approach for the capacitated location-routing problem. Computers & Operations Research 37 (11), 1912–1923.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., Bresson, X., 2020. Benchmarking graph neural networks.

Edwards, J., McKinnon, A., Cherrett, T., McLeod, F., Song, L., 2009. The impact of failed home deliveries on carbon emissions: Are collection/delivery points environmentally-friendly alternatives. In: 14th Annual Logistics Research Network Conference. Citeseer, p. M117.

Einav, L., Farronato, C., Levin, J., 2016. Peer to peer markets. Annual Review of Economics 8 (1), 615–635.

Enthoven, D. L., Jargalsaikhan, B., Roodbergen, K. J., uit het Broek, M. A., Schrotenboer, A. H., 2020. The two-echelon vehicle routing problem with covering options: City logistics with cargo bikes and parcel lockers. Computers & Operations Research 118, 104919.

Escobar, J. W., Linfati, R., Baldoquin, M. G., Toth, P., 2014a. A granular variable tabu neighborhood search for the capacitated location-routing problem. Transportation Research Part B: Methodological 67, 344–356.

Escobar, J. W., Linfati, R., Toth, P., 2013. A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. Computers & Operations Research 40 (1), 70–79.

Escobar, J. W., Linfati, R., Toth, P., Baldoquin, M. G., 2014b. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. Journal of Heuristics 20 (5), 483–509.

Estellés-Arolas, E., de Guevara, F. G.-L., 2012. Towards an integrated crowdsourcing definition. Journal of Information Science 38 (2), 189–200.

Favaretto, D., Moretti, E., Pellegrini, P., 2007. Ant colony system for a VRP with multiple time windows and multiple visits. Journal of Interdisciplinary Mathematics 10 (2), 263–284.

Figliozzi, M. A., 2010. An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. Transportation Research Part C: Emerging Technologies 18 (5), 668–679.

Florio, A. M., Feillet, D., Hartl, R. F., 2018. The delivery problem: Optimizing hit rates in e-commerce deliveries. Transportation Research Part B: Methodological 117, 455–472.

Fu, Z., Eglese, R., Li, L. Y. O., 2008. A unified tabu search algorithm for vehicle routing problems with soft time windows. Journal of the Operational Research Society 59 (5), 663–673.

Gershuny, J., Sullivan, O., 2017. United Kingdom Time Use Survey, 2014-2015.

Hashimoto, H., Ibaraki, T., Imahori, S., Yagiura, M., 2006. The vehicle routing problem with flexible time windows and traveling times. Discrete Applied Mathematics 154 (16), 2271–2290.

Hübner, A., Kuhn, H., Wollenburg, J., 2016. Last mile fulfilment and distribution in omni-channel grocery retailing. International Journal of Retail & Distribution Management 44 (3), 228–247.

Helsgaun, K., 2000. An effective implementation of the lin–kernighan traveling salesman heuristic. European Journal of Operational Research 126 (1), 106 – 130.

Helsgaun, K., 2017. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems: Technical report. Roskilde Universitet.

Hemmelmayr, V. C., Cordeau, J.-F., Crainic, T. G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research 39 (12), 3215–3228.

Hermes, 2020. Allgemeine Geschäftsbedingungen für den Versand von Hermes Päckchen. Hermes Germany GmbH.
URL `https://www.myhermes.de/content/pdf/agb_verpackungsrichtlinien.pdf`

Hillier, F., Lieberman, G., 1995. Introduction to operations research. McGraw-Hill, New York.

Hoogeboom, M., Dullaert, W., Lai, D., Vigo, D., 2020. Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. Transportation Science 54 (2), 400–416.

Hübner, A., Holzapfel, A., Kuhn, H., Obermair, E., 2019. Distribution in Omni-channel grocery retailing: An analysis of concepts realized. Series in Supply Chain Management. Springer, pp. 283–309.

Hutter, F., Xu, L., Hoos, H. H., Leyton-Brown, K., 2014. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence 206, 79–111.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., Yagiura, M., 2005. Effective local search algorithms for routing and scheduling problems with general time-window constraints. Transportation Science 39 (2), 206–232.

Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., Yagiura, M., 2008. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. Discrete Applied Mathematics 156 (11), 2050–2069.

Jie, W., Yang, J., Zhang, M., Huang, Y., 2019. The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. European Journal of Operational Research 272 (3), 879–904.

Joerss, M., Schroeder, J., Neuhaus, F., Klink, C., Mann, F., 2016. The future of last mile. Tech. rep., McKinsey & Company: Travel, Transport and Logistics.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., Schevon, C., 1989. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Operations Research 37 (6), 865–892.

Joshi, C. K., Cappart, Q., Rousseau, L.-M., Laurent, T., Bresson, X., 2020. Learning TSP requires rethinking generalization.

Kafle, N., Zou, B., Lin, J., 2017. Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. Transportation Research Part B: Methodological 99, 62–82.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., Talbi, E.-G., 2021. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. European Journal of Operational Research.

Kool, W., van Hoof, H., Welling, M., 2019. Attention, learn to solve routing problems!

Koskosidis, Y. A., Powell, W. B., Solomon, M. M., 1992. An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints. Transportation Science 26 (2), 69–85.

Kritzinger, S., Tricoire, F., Doerner, K. F., Hartl, R. F., Stützle, T., 2017. A unified framework for routing problems with a fixed fleet size. International Journal of Metaheuristics 6 (3), 160–209.

Larsen, R., Pacino, D., 2019. Fast delta evaluation for the vehicle routing problem with multiple time windows.
URL `https://arxiv.org/abs/1905.04114`

Le, T. V., Stathopoulos, A., Woensel, T. V., Ukkusuri, S. V., 2019. Supply, demand, operations, and management of crowd-shipping services: A review and empirical evidence. Transportation Research Part C: Emerging Technologies 103, 83–103.

Li, H., Lim, A., 2003. A metaheuristic for the pickup and delivery problem with time windows. International Journal on Artificial Intelligence Tools 12 (02), 173–186.

Lopes, R. B., Ferreira, C., Santos, B. S., 2016. A simple and effective evolutionary algorithm for the capacitated location–routing problem. Computers & Operations Research 70, 155–162.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58.

Luo, J., Chen, M.-R., 2014. Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the MDVRP and the MDVRPTW. Computers & Industrial Engineering 72, 84–97.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laganà, D., 2017. The vehicle routing problem with occasional drivers and time windows. In: Sforza, A., Sterle, C. (Eds.), Optimization and Decision Science: Methodologies and Applications. Springer International Publishing, Cham, pp. 577–587.

Marques, G., Sadykov, R., Deschamps, J.-C., Dupas, R., 2020. An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. Computers & Operations Research 114, 104833.

Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science 47 (3), 344–355.

McKinnon, A., 2016. Crowdshipping: a communal approach to reducing urban traffic levels? Workingpaper, Kühne Logistics University.

Mühlbauer, F., Fontaine, P., 2021. A parallelised large neighbourhood search heuristic for the asymmetric two-echelon vehicle routing problem with swap containers for cargo-bicycles. European Journal of Operational Research 289 (2), 742 – 757.

Montoya-Torres, J. R., Franco, J. L., Isaza, S. N., Jiménez, H. F., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering 79, 115–129.

Mourad, A., Puchinger, J., Chu, C., 2019. A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological 123, 323–346.

Mouthuy, S., Massen, F., Deville, Y., Van Hentenryck, P., 2015. A multistage very large-scale neighborhood search for the vehicle routing problem with soft time windows. Transportation Science 49 (2), 223–238.

Nagata, Y., 1997. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In: Proc. of 7th Int. Conf. on Genetic Algorithms, 1997.

Nagata, Y., Bräysy, O., Dullaert, W., 2010. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research 37 (4), 724–737.

Okholm, H. B., Thelle, M. H., Möller, A., Basalisco, B., Rolmer, S., 2013. E-commerce and delivery: A study of the state of play of EU parcel markets with particular emphasis on e-commerce. European Commission.

Özarık, S. S., Veelenturf, L. P., Van Woensel, T., Laporte, G., 2021. Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. Transportation Research Part E: Logistics and Transportation Review 148, 102263.

Pan, S., Giannikas, V., Han, Y., Grover-Silva, E., Qiao, B., 2017. Using customer-related data to enhance e-grocery home delivery. Industrial Management & Data Systems 117 (9), 1917–1933.

Perboli, G., Tadei, R., Vigo, D., 2011. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. Transportation Science 45 (3), 364–380.

Pirkwieser, S., Raidl, G. R., 2010. Variable neighborhood search coupled with ILP-based very large neighborhood searches for the (periodic) location-routing problem. In: Hybrid Metaheuristics. Springer Berlin Heidelberg, pp. 174–189.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & Operations Research 34 (8), 2403–2435.

Pitney Bowes, 2021. Pitney Bowes shipping index 2021.
URL `https://www.pitneybowes.com/content/dam/pitneybowes/us/en/shipping-index/parcel_shipping_index_ebook_final.pdf`

Praet, S., Martens, D., 2019. Efficient parcel delivery by predicting customers' locations. Decision Sciences 51 (5), 1202–1231.

Prins, C., Prodhon, C., Calvo, R. W., 2004. Nouveaux algorithmes pour le problème de localisation et routage sous contraintes de capacité. MOSIM'04 2, 1115–1122.

Prins, C., Prodhon, C., Calvo, R. W., 2006a. A memetic algorithm with population management (MA|PM) for the capacitated location-routing problem. In: Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg, pp. 183–194.

Prins, C., Prodhon, C., Calvo, R. W., 2006b. Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. 4OR 4 (3), 221–238.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., Calvo, R. W., 2007. Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. Transportation Science 41 (4), 470–483.

Prodhon, C., Prins, C., 2008. A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. In: Hybrid Metaheuristics. Springer Berlin Heidelberg, pp. 43–57.

Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. European Journal of Operational Research 238 (1), 1–17.

Punakivi, M., Saranen, J., 2001. Identifying the success factors in e-grocery home delivery. International Journal of Retail & Distribution Management 29 (4), 156–163.

Punakivi, M., Yrjölä, H., Holmström, J., 2001. Solving the last mile issue: Reception box or delivery box? International Journal of Physical Distribution & Logistics Management 31 (6), 427–439.

Qi, W., Li, L., Liu, S., Shen, Z.-J. M., 2018. Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact. Manufacturing & Service Operations Management 20 (4), 737–751.

Rais, A., Alvelos, F., Carvalho, M., 2014. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. European Journal of Operational Research 235 (3), 530–539.

Raviv, T., Tenzer, E. Z., 2018. Crowd-shipping of small parcels in a physical internet. Workingpaper, Tel Aviv University.

Renaud, J., Laporte, G., Boctor, F. F., 1996. A tabu search heuristic for the multi-depot vehicle routing problem. Computers & Operations Research 23 (3), 229–235.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40 (4), 455–472.

Rouges, J.-F., Montreuil, B., 2014. Crowdsourcing delivery: New hyperconnected business models to reinvent delivery. In: 1st Physical Internet Conference IPIC 2014. Québec, Canada.

Sadati, M. E. H., Çatay, B., Aksen, D., 2021. An efficient variable neighborhood search with tabu shaking for a class of multi-depot vehicle routing problems. Computers & Operations Research 133, 105269.

Sadykov, R., Uchoa, E., Pessoa, A., 2021. A bucket graph-based labeling algorithm with application to vehicle routing. Transportation Science 55 (1), 4–28.

Sampaio Oliveira, A., Savelsbergh, M., Veelenturf, L., van Woensel, T., 2019. Crowd-based city logistics. In: Sustainable Transportation and Smart Logistics. Elsevier, Amsterdam, pp. 381–400.

Sampaio Oliveira, A., Savelsbergh, M., Veelenturf, L. P., Van Woensel, T., 2020. Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. Networks 76 (2), 232–255.

Schaap, H., Schiffer, M., Schneider, M., Walther, G., 2019. A large neighborhood search for the vehicle routing problem withmultiple time windows. Working Paper.

Schneider, M., Drexl, M., 2017. A survey of the standard location-routing problem. Annals of Operations Research 259 (1-2), 389–414.

Schneider, M., Löffler, M., 2019. Large composite neighborhoods for the capacitated location-routing problem. Transportation Science 53 (1), 301–318.

Schwerdfeger, S., Boysen, N., 2020. Optimizing the changing locations of mobile parcel lockers in last-mile distribution. European Journal of Operational Research 285 (3), 1077–1094.

Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N., Woensel, T. V., 2022. Two-echelon vehicle routing problems: A literature review. European Journal of Operational Research.

Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35 (2), 254–265.

Solomon, M. M., Desrosiers, J., 1988. Survey paper—time window constrained routing and scheduling problems. Transportation Science 22 (1), 1–13.

Soto Setzke, D., Pfluegler, C., Schreieck, M., Froehlich, S., Wiesche, M., Krcmar, H., 2017. Matching drivers and transportation requests in crowdsourced delivery systems. In: Twenty-third Americas Conference on Information Systems. Boston.

Subramanian, A., Uchoa, E., Ochi, L. S., 2013. A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research 40 (10), 2519–2531.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 31 (2), 170–186.

Taillard, É. D., Helsgaun, K., 2019. POPMUSIC for the travelling salesman problem. European Journal of Operational Research 272 (2), 420–429.

Ting, C.-J., Chen, C.-H., 2013. A multiple ant colony optimization algorithm for the capacitated location routing problem. International Journal of Production Economics 141 (1), 34–44.

Tuzun, D., Burke, L. I., 1999. A two-phase tabu search approach to the location routing problem. European Journal of Operational Research 116 (1), 87–99.

van Duin, J. H. R., de Goffau, W., Wiegmans, B., Tavasszy, L. A., Saes, M., 2016. Improving home delivery efficiency by using principles of address intelligence for B2C deliveries. Transportation Research Procedia 12, 14–25.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60 (3), 611–624.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2013a. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European Journal of Operational Research 231 (1), 1–21.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2013b. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research 40 (1), 475–489.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2014a. Implicit depot assignments and rotations in vehicle routing heuristics. European Journal of Operational Research 237 (1), 15–28.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2014b. A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research 234 (3), 658–673.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2015. Timing problems and algorithms: Time decisions for sequences of activities. Networks 65 (2), 102–128.

Voigt, S., Frank, M., Fontaine, P., Kuhn, H., 2021. The vehicle routing problem with availability profiles. Working Paper.

Voigt, S., Kuhn, H., 2022. Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. Networks 79 (3), 403–426.

von Abrams, K., 2021. Global ecommerce forecast 2021.

Wang, K., Shao, Y., Zhou, W., 2017. Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. Transportation Research Part D: Transport and Environment 57, 262–276.

Wang, Y., Zhang, D., Liu, Q., Shen, F., Lee, L. H., 2016. Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions. Transportation Research Part E: Logistics and Transportation Review 93, 279–293.

Wong, R. T., 2008. Vehicle routing for small package delivery and pickup services. In: Operations Research/Computer Science Interfaces. Springer US, pp. 475–485.

Yu, B., Yang, Z.-Z., Xie, J.-X., 2011. A parallel improved ant colony optimization for multi-depot vehicle routing problem. Journal of the Operational Research Society 62 (1), 183–188.

Yu, V. F., Jodiawan, P., Hou, M.-L., Gunawan, A., 2021. Design of a two-echelon freight distribution system in last-mile logistics considering covering locations and occasional drivers. Transportation Research Part E: Logistics and Transportation Review 154, 102461.

Yu, V. F., Lin, S.-W., Lee, W., Ting, C.-J., 2010. A simulated annealing heuristic for the capacitated location routing problem. Computers & Industrial Engineering 58 (2), 288–299.

Appendix A

Appendix for The Vehicle Routing Problem with Availability Profiles

A.1 Parameters

Table A.1: Parameters for HALNS

Parameter	**Meaning**	**Chosen Value**
n_p	Size of the initial population	48
$gen^{\max}$	Number of generations	30
gen^{stop}	Number of generations without improvement	10
it^{limit}	Number of iterations without improvement (one ALNS run)	30,000 (initial population) 7,500 (following generations)
β	Cool rate in SA	0.9999
χ_0	Acceptance probability in SA	0.25
n_0	Number of iterations for determining the initial SA temperature	400
ω	Weight for penalties drawn for every ALNS run	$unif(0.01, 1.00) \cdot C^{\text{VRP}}$
p^{binom}	Probability for binomial distribution drawn for every ALNS run	$unif(0.12, 0.24)$
σ_1	Score for operator - new best solution	35
σ_2	Score for operator - new best current solution	2
σ_3	Score for operator - worse solution, but accepted via SA	1

A.2 Results for VRPTW, VRPMTW and VRPSTW Benchmark Instances

Tables A.2-A.5 present the detailed results for VRPTW, VRPMTW and VRPSTW benchmark instances. The first column shows the *Instance* name, the second column the previous *BKS* and the following columns show the average and best results achieved for each approach. The last rows present cumulated and averaged measures: Σ sums up the objective values over all instances, *Avg gap* shows the average gap to the previous BKS, *# BKS* counts the number of (current) BKS achieved with one approach, *Avg T* shows the average runtime, *CPU* the processor used and its *Passmark* score. **Bold** entries state the BKS, entries with an asterisk (*) are new BKS found by the HALNS. Further explanations are given below.

Additionally, Figures A.1 (VRPTW) and A.2 (VRPMTW) plot the average and best cumulated objective values against respective standardized runtimes, $\bar{T}_{\text{scaled}}$ *[s]*, as if run on the AMD Ryzen 9 3900X using the passmark single thread ratings. The graphs of the HALNS are generated by varying the stopping condition, $gen^{\text{stop}} = [1, \ldots, 10]$.

A.2.1 Results for Solomon and Desrosiers (1988) VRPTW Instances

Figure A.1 and Table A.2 show the detailed results for the VRPTW benchmark instances.

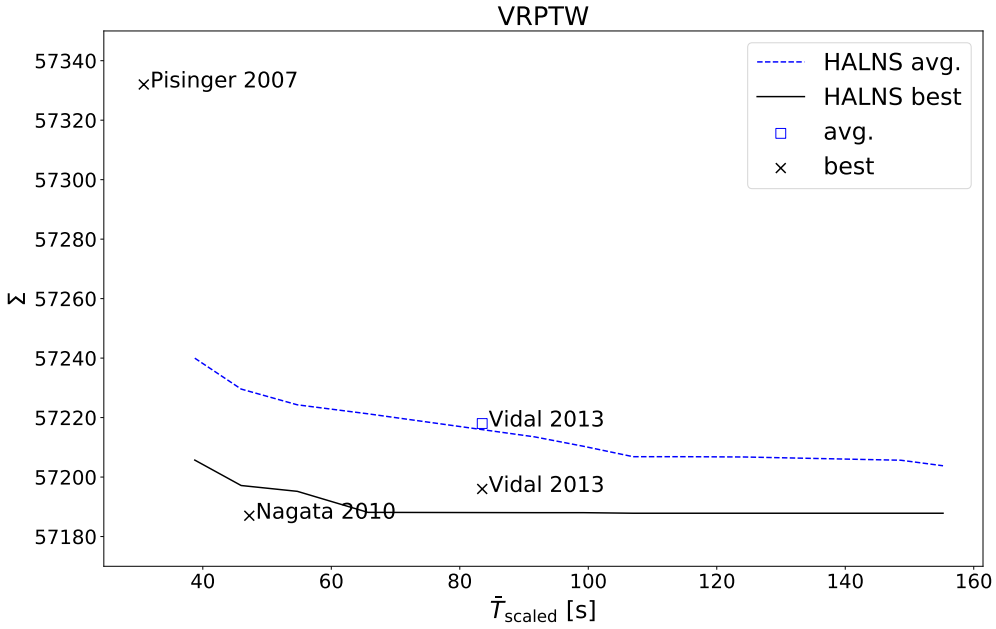


Figure A.1: VRPTW Benchmark

Table A.2: Detailed results for VRPTW instances

Instance	BKS	HALNS	
		Best 10	Avg 10
C101.100.10	828.94	**828.94**	**828.94**
C102.100.10	828.94	**828.94**	**828.94**
C103.100.10	828.06	828.07	828.07
C104.100.10	824.78	**824.78**	**824.78**
C105.100.10	828.94	**828.94**	**828.94**
C106.100.10	828.94	**828.94**	**828.94**
C107.100.10	828.94	**828.94**	**828.94**
C108.100.10	828.94	**828.94**	**828.94**
C109.100.10	828.94	**828.94**	**828.94**
C201.100.3	591.56	**591.56**	**591.56**
C202.100.3	591.56	**591.56**	**591.56**
C203.100.3	591.17	**591.17**	**591.17**
C204.100.3	590.6	**590.6**	**590.6**
C205.100.3	588.88	**588.88**	**588.88**
C206.100.3	588.49	**588.49**	**588.49**
C207.100.3	588.29	**588.29**	**588.29**
C208.100.3	588.32	**588.32**	**588.32**
R101.100.19	1650.8	**1650.8**	**1650.8**
R102.100.17	1486.12	**1486.12**	**1486.12**
R103.100.13	1292.68	**1292.68**	**1292.68**
R104.100.9	1007.31	**1007.31**	**1007.31**
R105.100.14	1377.11	**1377.11**	**1377.11**
R106.100.12	1252.03	**1252.03**	**1252.03**
R107.100.10	1104.66	**1104.66**	1104.74
R108.100.9	960.88	**960.88**	961.44
R109.100.11	1194.73	**1194.73**	1195.81
R110.100.10	1118.84	**1118.84**	1118.9
R111.100.10	1096.72	1096.73	1096.73
R112.100.9	982.14	982.25	987.51
R201.100.4	1252.37	**1252.37**	**1252.37**
R202.100.3	1191.7	**1191.7**	**1191.7**
R203.100.3	939.5	**939.5**	939.82
R204.100.2	825.52	**825.52**	826.17
R205.100.3	994.43	**994.43**	**994.43**
R206.100.3	906.14	**906.14**	**906.14**
R207.100.2	890.61	**890.61**	**890.61**
R208.100.2	726.82	**726.82**	**726.82**
R209.100.3	909.16	**909.16**	**909.16**
R210.100.3	939.37	**939.37**	941.35
R211.100.2	885.71	**885.71**	890.9
RC101.100.14	1696.95	**1696.95**	**1696.95**
RC102.100.12	1554.75	**1554.75**	**1554.75**
RC103.100.11	1261.67	**1261.67**	**1261.67**
RC104.100.10	1135.48	**1135.48**	**1135.48**
RC105.100.13	1629.44	**1629.44**	**1629.44**
RC106.100.11	1424.73	**1424.73**	**1424.73**
RC107.100.11	1230.48	**1230.48**	1230.57
RC108.100.10	1139.82	**1139.82**	**1139.82**
RC201.100.4	1406.94	**1406.94**	**1406.94**
RC202.100.3	1365.65	**1365.65**	1366.19
RC203.100.3	1049.62	1050.45	1050.6
RC204.100.3	798.46	**798.46**	**798.46**
RC205.100.4	1297.65	**1297.65**	**1297.65**
RC206.100.3	1146.32	**1146.32**	**1146.32**
RC207.100.3	1061.14	**1061.14**	**1061.14**
RC208.100.3	828.14	**828.14**	**828.14**
Σ	57187	57188	57204
Avg gap	0.00%	0.00%	0.03%
# BKS	56	52	
	Avg T	155s	
	CPU	Ryzen 9 3900X	
	Passmark	2731	

A.2.2 Results for Belhaiza et al. (2014) VRPMTW Instances

Figure A.2 and Table A.3 show the detailed results for the VRPMTW benchmark instances. Columns Belhaiza et al. (2014), Larsen and Pacino (2019), Schaap et al. (2019) and HALNS report the best and average results from 10 runs, while columns Belhaiza et al. (2017) and Hoogeboom et al. (2020) only show results from a single run. Columns K indicate the number of vehicles used in the corresponding best solution found.

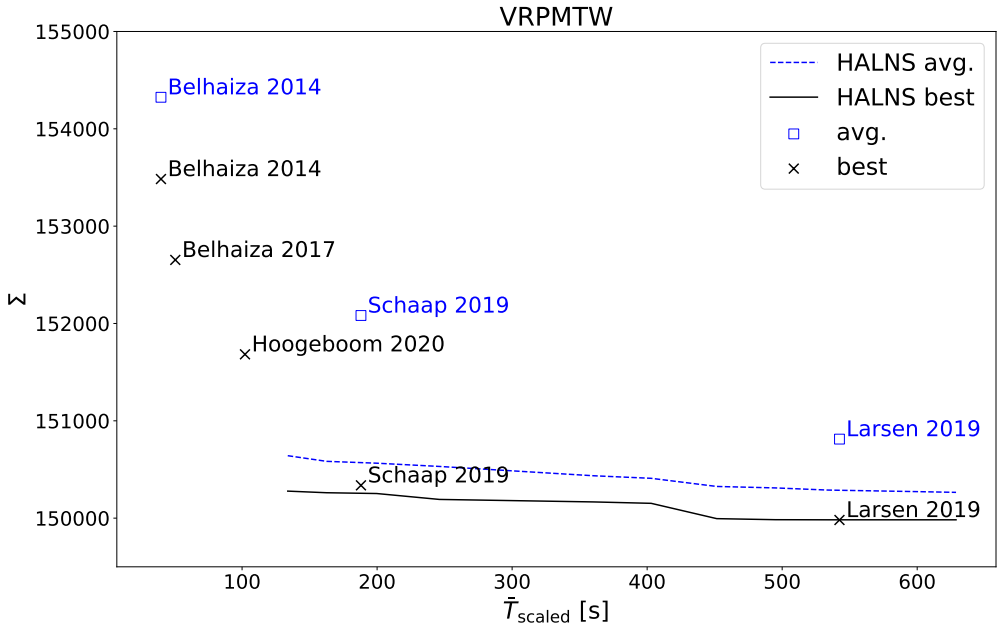


Figure A.2: VRPMTW Benchmark

Table A.3: Detailed Results for Belhaiza et al. (2014) VRPMTW Instances

Instance	Belhaiza 14			Belhaiza 17		Larsen 19			Schaap 19			Hoogeboom 20		HALNS		
	K	Best	Avg	K	Best/Avg	K	Best	Avg	K	Best	Avg	K	Best/Avg	K	Best	Avg
rm101	10	2977.2	3005	10	2970.2	10	**2968.8**	2969	10	2977.2	2991.9	10	2974.2	10	2971.3	2975.6
rm102	9	2759.4	2759.4	9	2725.3	9	**2705.9**	2716.7	9	2730	2897.9	9	2726.7	9	2706.7	2713.2
rm103	9	2692.5	2710.5	9	2681.5	9	**2680.4**	2681.6	9	2693.4	2702.8	9	2699	9	2680.5	2681.7
rm104	9	2696.6	2719.2	9	2691.8	9	**2690.7**	2691.5	9	2697.6	2701.4	9	**2690.7**	9	2691.4	2692.6
rm105	9	2688.8	2711	9	2687.2	9	**2683.7**	**2683.7**	9	2689.7	2699.9	9	2691.5	9	**2683.7**	2685.5
rm106	9	**2691.9**	**2691.9**	9	2700.9	9	2700.9	2703.9	9	2707.1	2717.5	9	2700.9	9	2700.9	2700.9
rm107	9	2690.8	2714.7	9	**2685.1**	9	**2685.1**	**2685.1**	9	2686	2700.4	9	2695.9	9	**2685.1**	**2685.1**
rm108	9	2729.1	2729.1	9	**2716**	9	**2716**	2717.4	9	2727.6	2774.8	9	2725.5	9	**2716**	**2716**
rm201	3	3711.4	3720.5	3	3686.4	2	2745.6	2756	2	2753.3	2780.6	2	2739.9	2	**2738.4***	**2738.4***
rm202	2	2698.1	2717.3	2	2681	2	2681.3	2685	2	2691.3	2703.6	2	**2679.9**	2	2681.3	2681.3
rm203	2	2686.1	2702	2	**2673.6**	2	2679	2679.7	2	2684.4	2693.6	2	2675.9	2	2677.9	2678.4
rm204	2	2680.5	2691.2	2	**2664.9**	2	2672.8	2673.1	2	2678.4	2683.4	2	2671.7	2	2672.1	2674
rm205	2	2671	2688.2	2	**2651.3**	2	2671	2671.1	2	2671	2678.8	2	2665.2	2	2671	2671.2
rm206	2	2686.3	2704.9	2	2672.8	2	2679	2679	2	2688.3	2697.2	2	**2665.6**	2	2679	2679
rm207	2	2678.2	2696.2	2	**2657.3**	2	2674.7	2678.2	2	2682.4	2693.8	2	2674.7	2	2674.4	2674.4
rm208	2	2673.9	2690.7	2	2663.6	2	2673.9	2676.2	2	2678.6	2686	2	**2662.2**	2	2673.9	2674.3
cm101	10	3089.2	3102.4	10	3101.2	10	**3073**	3126.8	10	3126.7	3172.6	10	3180.4	10	3144.6	3178.4
cm102	12	3426.9	3426.9	11	3339.7	12	3392.5	3463	12	3467.8	3481.7	11	3433.6	11	**3331.1***	3436.8
cm103	12	3532.7	3572.7	12	3520.5	11	**3434.6**	3477.1	11	3436.8	3471.9	11	3477.3	11	3435	3473.7
cm104	14	4051.3	4058	14	4048	13	**3859.6**	3907.1	13	3870.8	3893.5	13	3862.6	13	3870.4	3885.5
cm105	11	3060.6	3077.3	10	**3010.2**	10	3037.9	3051.7	10	3018.8	3061.7	10	3055.6	10	3044.1	3060.7
cm106	10	2992.4	3020.2	10	2982.2	10	**2980.2**	2983.3	10	2982.9	2999.8	10	2982.2	10	2980.3	2980.6
cm107	11	3256.5	3292.3	11	3256.5	10	**3077.9**	**3077.9**	10	**3077.9**	3080.9	10	**3077.9**	10	**3077.9**	**3077.9**
cm108	10	2968.7	2973.1	10	2967.3	10	2965.4	2966.5	10	2965.6	2968.6	10	2966.7	10	**2965.1***	**2965.1***
cm201	5	4436.6	4452.5	5	4390.3	5	**4361.9**	4374.4	5	4392.2	4419.6	5	4392.6	5	4377.7	4388
cm202	6	4998.8	5024.9	6	4990.8	6	**4982.4**	4987.4	6	4996.5	5005.6	6	4988.8	6	**4982.4**	4987.9
cm203	5	4445.8	4484.6	5	4446.7	5	**4440.3**	4454.4	5	4445.5	4463.8	5	4461.6	5	4440.4	4451.4
cm204	5	4335.2	4372.4	5	4332.2	5	**4317.5**	4317.6	5	4321.5	4335.7	5	4318.3	5	**4317.5**	4319.9
cm205	4	3863.5	3883.2	4	3819.1	4	3816.8	3833.9	4	3818.7	3838.4	4	3840.8	4	**3816***	3820.7
cm206	4	3722	3743.2	4	3709.4	4	**3699**	3712	4	**3699**	3713.1	4	3718.9	4	**3699**	**3699**
cm207	4	3968.4	3977.8	4	3933.1	4	**3917.4**	3934.4	4	3921.9	3941.7	4	3945.7	4	**3917.4**	3919
cm208	4	3771.1	3793.2	4	3730.5	4	3714.6	3719.5	4	3716.1	3738.3	4	3755.8	4	**3713.9***	3717.2
rcm101	10	**3062**	3086.6	10	3063.5	10	**3062**	3062.2	10	3067.7	3070.8	10	3063.5	10	3062.4	3063.3
rcm102	10	3132.2	3142.3	10	3127.7	10	**3110.8**	3113.8	10	**3110.8**	3157.2	10	3133.7	10	**3110.8**	3111.1
rcm103	10	3152.9	3163.8	10	3131.8	10	**3121.3**	3121.9	10	3124.9	3138.9	10	3152.1	10	**3121.3**	3121.8
rcm104	10	3119.6	3134.6	10	3111.8	10	**3111**	3111.9	10	3111.2	3124.1	10	3129.1	10	**3111**	**3111**
rcm105	10	3187.9	3210.7	10	3185.9	10	3165.3	3169.5	10	3172.8	3203.8	10	3187.8	10	**3164.6***	3166.2
rcm106	10	3218.9	3218.9	10	3193.4	10	**3158.2**	3164.6	10	3173.1	3191.4	10	3195.1	10	**3158.2**	3162.3
rcm107	11	3488.9	3514	11	**3487.7**	11	**3487.7**	3490.2	11	**3487.7**	3497.3	11	3488.9	11	**3487.7**	**3487.7**
rcm108	11	3592.7	3592.7	11	3532.6	11	3531.1	3531.1	11	3532.5	3544.7	11	3530.6	11	**3529.5***	3530.8
rcm201	2	2804	2827.8	2	2778.7	2	2715.4	2748.1	2	2701.7	2736.4	2	2706.8	2	**2695.9***	**2695.9***
rcm202	2	2836.9	2846.8	2	2815.9	2	2716.8	2733.8	2	2722.3	2749.1	2	2767.4	2	**2716.3***	2719.1
rcm203	2	2721.9	2725.4	2	2722	2	**2704.8**	2714.2	2	2707.3	2736.2	2	2825.5	2	**2704.8**	2704.9
rcm204	2	2726.5	2743.1	2	2698.4	2	2692.6	2699.3	2	2696.5	2703.9	2	2719.1	2	**2690.5***	**2690.5***
rcm205	2	2754.5	2775.7	2	2754.5	2	2718.8	2725.1	2	2721.2	2733.3	2	2724.2	2	**2711.6***	**2712.3**
rcm206	2	2812.7	2830.6	2	2769.6	2	**2721.9**	2743.1	2	2725.9	2764.1	2	2781.3	2	**2721.9**	2722.9
rcm207	3	3749.8	3786.8	3	3749.8	2	2861	3216.9	2	2863.4	3585.4	3	3744.9	2	**2857.6***	**2857.6***

Continued on next page

Table A.3 – *Continued from previous page*

rcm208	2	2791.4	2817.2	2	2742.7	2	**2722.7**	2732.5	2	**2722.7**	2755.1	2	2735.1	2	**2722.7**	2724.2
Σ	316	153484	154324	314	152653	310	149981	150812	310	150337	152082	310	151683	309	149983	150265
Avg gap		2.60%	3.16%		2.04%		0.13%	0.70%		0.37%	1.60%		1.33%		0.13%	0.31%
# BKS		2			8		27			5			5		28 (12 new)	
Avg T		64s			81s		600s			185s			113s		629s	
CPU		i5 3.3 GHz			i5 3.3 GHz		i7-4790K			i7 3.7GHz			i7 4 GHz		Ryzen 9 3900X	
Passm.		1704			1704		2469			2776			2469		2731	

A.2.3 Results for VRPSTW Instances

We compare our results for the VRPSTW against the approaches of Vidal et al. (2014b), Mouthuy et al. (2015) and Kritzinger et al. (2017). Tables A.4 and A.5 present the detailed results for types 1 and 2 with $\alpha = 1$ as described above.

Table A.4: Detailed results for VRPSTW instances type 1 (only lateness), $\alpha = 1$

Instance	BKS	Kritzinger		Vidal		HALNS	
		Best 10	Avg 10	Best 10	Avg 10	Best 10	Avg 10
C101.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C102.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C103.100.10	828.06	828.07	828.07	**828.06**	**828.06**	828.07	828.07
C104.100.10	824.78	**824.78**	**824.78**	**824.78**	**824.78**	**824.78**	**824.78**
C105.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C106.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C107.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C108.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C109.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**	**828.94**
C201.100.3	591.56	**591.56**	**591.56**	**591.56**	**591.56**	**591.56**	**591.56**
C202.100.3	591.56	**591.56**	**591.56**	**591.56**	**591.56**	**591.56**	**591.56**
C203.100.3	591.17	**591.17**	**591.17**	**591.17**	**591.17**	**591.17**	**591.17**
C204.100.3	590.6	**590.6**	**590.6**	**590.6**	**590.6**	**590.6**	**590.6**
C205.100.3	588.88	**588.88**	**588.88**	**588.88**	**588.88**	**588.88**	**588.88**
C206.100.3	588.49	**588.49**	**588.49**	**588.49**	**588.49**	**588.49**	**588.49**
C207.100.3	588.29	**588.29**	**588.29**	**588.29**	**588.29**	**588.29**	**588.29**
C208.100.3	588.32	**588.32**	**588.32**	**588.32**	**588.32**	**588.32**	**588.32**
R101.100.19	1562.58	**1562.58**	1562.98	**1562.58**	1562.89	**1562.58**	**1562.58**
R102.100.17	1379.11	**1379.11**	1379.62	**1379.11**	1379.21	**1379.11**	**1379.11**
R103.100.13	1159.28	1159.54	1160.64	**1159.28**	1159.51	**1159.28**	1159.41
R104.100.9	999.77	1003.73	1009.02	**999.77**	**999.77**	**999.77**	1000.04
R105.100.14	1347.75	**1347.75**	1348.89	**1347.75**	**1347.75**	**1347.75**	**1347.75**
R106.100.12	1236.58	**1236.58**	1237.29	**1236.58**	**1236.58**	**1236.58**	1236.6
R107.100.10	1083.62	1084.96	1089.84	**1083.62**	**1083.62**	**1083.62**	1083.66
R108.100.9	946.6	949.94	951.24	**946.6**	947.04	**946.6**	947.22
R109.100.11	1173.21	**1173.21**	1176.4	**1173.21**	**1173.21**	**1173.21**	**1173.21**
R110.100.10	1106.66	**1106.66**	1114.66	1107.26	1111.57	1109.58	1110.22
R111.100.10	1074.84	1080.25	1086.36	**1074.84**	1076.41	**1074.84**	1077.03
R112.100.9	971.31	972.11	981.82	**971.31**	975.78	**971.31**	971.94
R201.100.4	1237.11	**1237.11**	1237.17	**1237.11**	**1237.11**	**1237.11**	**1237.11**
R202.100.3	1165.32	**1165.32**	1169.23	**1165.32**	**1165.32**	**1165.32**	**1165.32**
R203.100.3	933.52	937.35	942.96	**933.52**	934.01	934.1	934.1
R204.100.2	824.02	832.38	840.79	**824.02**	824.73	**824.02**	825.4
R205.100.3	994.43	**994.43**	1006.79	**994.43**	**994.43**	**994.43**	**994.43**
R206.100.3	906.14	912.81	920.13	**906.14**	**906.14**	**906.14**	**906.14**
R207.100.2	887.28	908.7	1044.87	**887.28**	888.44	**887.28**	**887.28**
R208.100.2	726.82	728.92	735.26	**726.82**	727.08	**726.82**	**726.82**
R209.100.3	909.16	909.3	917.21	**909.16**	**909.16**	**909.16**	909.31
R210.100.3	938.34	948.8	958.58	**938.34**	941.95	**938.34**	**938.34**
R211.100.2	885.71	901.18	923.85	**885.71**	892.5	890.79	890.79
RC101.100.14	1590.22	**1590.22**	1591.59	**1590.22**	**1590.22**	**1590.22**	1591.23
RC102.100.12	1428.21	**1428.21**	1429.9	**1428.21**	**1428.21**	**1428.21**	**1428.21**
RC103.100.11	1239.54	**1239.54**	1242.33	**1239.54**	1239.73	**1239.54**	1240.91
RC104.100.10	1126.31	**1126.31**	1128.74	**1126.31**	**1126.31**	**1126.31**	**1126.31**
RC105.100.13	1450.84	**1450.84**	1451.38	**1450.84**	**1450.84**	**1450.84**	**1450.84**
RC106.100.11	1349.3	**1349.3**	1350.17	**1349.3**	1349.72	1350.57	1353.63
RC107.100.11	1208.81	**1208.81**	1208.96	**1208.81**	1208.98	**1208.81**	**1208.81**
RC108.100.10	1118	**1118**	1119.61	**1118**	1118.31	1119.59	1119.59
RC201.100.4	1380.33	**1380.33**	1380.47	**1380.33**	**1380.33**	**1380.33**	**1380.33**
RC202.100.3	1317.28	**1317.28**	1322.17	**1317.28**	**1317.28**	**1317.28**	1317.37
RC203.100.3	1040.77	1046.05	1057.1	**1040.77**	1045	**1040.77**	**1040.77**
RC204.100.3	797.04	797.41	809.09	**797.04**	**797.04**	**797.04**	**797.04**
RC205.100.4	1297.65	1299.61	1305.97	**1297.65**	1298	**1297.65**	1299.08
RC206.100.3	1135.26	**1135.26**	1135.9	**1135.26**	**1135.26**	**1135.26**	**1135.26**
RC207.100.3	1056.88	1061.14	1073.58	**1056.88**	1058.16	**1056.88**	**1056.88**
RC208.100.3	827.67	829	834.82	**827.67**	827.9	**827.67**	**827.67**
Σ	55987.56	56084.33	56411.68	55988.16	56019.79	55999.01	56012.04
Avg gap	0.00%	0.19%	0.81%	0.00%	0.06%	0.02%	0.04%
# BKS	56/56	36		55		50	
	Avg T	600s		349s		156s	
	CPU	Xeon E7-8837		Opteron 2.2G		Ryzen 9 3900X	
	Passmark	1124		445		2731	

Table A.5: Detailed results for VRPSTW instances type 2 (earliness and lateness), $\alpha = 1$

Instance	BKS	Vidal		HALNS	
		Best 10	Avg 10	Best 10	Avg 10
C101.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C102.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C103.100.10	828.06	**828.06**	**828.06**	828.07	828.07
C104.100.10	824.78	**824.78**	**824.78**	**824.78**	**824.78**
C105.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C106.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C107.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C108.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C109.100.10	828.94	**828.94**	**828.94**	**828.94**	**828.94**
C201.100.3	591.56	**591.56**	**591.56**	**591.56**	**591.56**
C202.100.3	591.56	**591.56**	**591.56**	**591.56**	**591.56**
C203.100.3	591.17	**591.17**	**591.17**	**591.17**	**591.17**
C204.100.3	590.6	**590.6**	**590.6**	**590.6**	**590.6**
C205.100.3	588.88	**588.88**	**588.88**	**588.88**	**588.88**
C206.100.3	588.49	**588.49**	**588.49**	**588.49**	**588.49**
C207.100.3	588.29	**588.29**	**588.29**	**588.29**	**588.29**
C208.100.3	588.32	**588.32**	**588.32**	**588.32**	**588.32**
R101.100.19	1546.91	**1546.91**	**1546.91**	1557.82	1561.92
R102.100.17	1377.38	**1377.38**	**1377.38**	1378.44	1379.8
R103.100.13	1158.31	**1158.31**	1158.83	1158.48	1159.71
R104.100.9	1000.33	**1000.33**	1004.57	1002.68	1003.7
R105.100.14	1342.57	**1342.57**	**1342.57**	1347.61	1347.72
R106.100.12	1223.09	**1223.09**	**1223.09**	1226.13	1226.8
R107.100.10	1079.12	**1079.12**	1080.9	1081.51	1083.76
R108.100.9	945.64	945.64	948.23	**944.23***	947.33
R109.100.11	1164.68	**1164.68**	**1164.68**	1167.88	1173.72
R110.100.10	1104.59	**1104.59**	1108.3	1107.34	1111.01
R111.100.10	1065.76	**1065.76**	**1065.76**	1066.47	1070.24
R112.100.9	969.91	**969.91**	991.5	969.98	970.25
R201.100.4	1235.14	**1235.14**	**1235.14**	1237.11	1237.11
R202.100.3	1159.76	**1159.76**	**1159.76**	1165.32	1165.32
R203.100.3	934.1	934.1	937.04	**934.09***	**934.09***
R204.100.2	820.9	**820.9**	837.21	821.54	821.54
R205.100.3	994.43	**994.43**	996.24	**994.43**	**994.43**
R206.100.3	906.54	906.54	910.99	**906.14***	**906.14***
R207.100.2	906.81	906.81	937.79	**887.28***	891.01
R208.100.2	730.52	730.52	735.31	**726.82***	727.18
R209.100.3	909.16	**909.16**	911.61	**909.16**	**909.16**
R210.100.3	938.77	938.77	948.91	**938.34***	**938.34***
R211.100.2	912.39	912.39	921.81	**885.71***	890.26
RC101.100.14	1584.2	**1584.2**	**1584.2**	1588.58	1591.72
RC102.100.12	1409.36	**1409.36**	**1409.36**	1416.67	1417.69
RC103.100.11	1231.67	**1231.67**	**1231.67**	1239.54	1241.41
RC104.100.10	1121.84	**1121.84**	1123.25	1126.31	1127.07
RC105.100.13	1433.37	**1433.37**	**1433.37**	1436.99	1437.04
RC106.100.11	1334.89	**1334.89**	**1334.89**	1338.42	1338.49
RC107.100.11	1203.06	**1203.06**	**1203.06**	1207.04	1208.39
RC108.100.10	1115.44	**1115.44**	**1115.44**	1119.59	1119.84
RC201.100.4	1380.33	**1380.33**	**1380.33**	**1380.33**	**1380.33**
RC202.100.3	1312.05	**1312.05**	**1312.05**	**1312.05**	1312.06
RC203.100.3	1044.74	1044.74	1047.43	**1040.65***	**1040.65***
RC204.100.3	796.68	**796.68**	796.91	797.04	797.05
RC205.100.4	1297.86	**1297.86**	1300.98	1297.97	1298.78
RC206.100.3	1135.26	**1135.26**	1135.44	**1135.26**	**1135.26**
RC207.100.3	1056.88	**1056.88**	1061.92	**1056.88**	**1056.88**
RC208.100.3	827.67	**827.67**	832.3	**827.67**	**827.67**
Σ	55886.4	55886.4	56021.42	55909.8	55955.17
Avg gap	0.00%	0.00%	0.26%	0.00%	0.07%
# BKS	48/56	48		31 (8 new)	
	Avg T	1797s		159s	
	CPU	Opteron 2.2G		Ryzen 9 3900X	
	Passmark	445		2731	

A.2.4 Results for VRPAP Instances

Table A.6 compares the HALNS algorithm with the lower and upper bound described in Section 1.4. Column $C^{\text{trans}}_{\text{VRP}}$ shows the lowest possible transportation costs, achieved by solving a VRP, Column $C^{\text{failed}}_{\text{VRPTW}}$ the lowest possible failed-delivery costs, and Column $C^{\text{lower}}_{\text{VRPAP}}$ the lower bound, derived by the summation of both terms before. Column *Best 10* and *Avg 10* show the best and average result of ten runs. Column $C^{\text{upper}}_{\text{VRPAP}}$ signifies the upper bound. Columns Δ^{lb} and Δ^{ub} represent the percentage gaps to the lower and upper bound, respectively. Column *Avg T [s]* shows the average runtime of ten runs in seconds.

Table A.6: Results for VRPAP instances

Instance	$C^{\text{trans}}_{\text{VRP}}$	$C^{\text{failed}}_{\text{VRPTW}}$	$C^{\text{lower}}_{\text{VRPAP}}$	Best 10	Avg 10	$C^{\text{upper}}_{\text{VRPAP}}$	Δ^{lb}	Δ^{ub}	Avg T [s]
R101									
A	865.95	86.6	952.55	1213.82	1217	1241.08	27%	-2%	811
V	865.95	86.6	952.55	1279.87	1286.43	1355.9	34%	-6%	779
W	865.95	173.19	1039.14	1253.02	1260.39	1326.98	21%	-6%	1020
M	865.95	86.6	952.55	1191.85	1198.11	1269.92	25%	-6%	819
AV	865.95	86.6	952.55	1216.4	1223.22	1303.43	28%	-7%	1044
WM	865.95	129.89	995.84	1163.34	1168.37	1304.21	17%	-11%	1001
AVWM	865.95	108.24	974.19	1180.49	1188.29	1298.67	21%	-9%	1068
R201									
A	651.3	65.13	716.43	915.79	920.04	936.57	28%	-2%	1074
V	651.3	65.13	716.43	919.24	920.19	1012.77	28%	-9%	1047
W	651.3	130.26	781.56	907.79	910.85	987.63	16%	-8%	1043
M	651.3	65.13	716.43	919.76	921.02	962.04	28%	-4%	979
AV	651.3	65.13	716.43	887.67	888.56	967.83	24%	-8%	968
WM	651.3	97.7	749	882.62	882.91	976.62	18%	-10%	969
AVWM	651.3	81.41	732.71	888.17	889.87	968.94	21%	-8%	1023
C101									
A	824.78	82.48	907.26	1141.24	1141.41	1187.68	26%	-4%	699
V	824.78	82.48	907.26	1167.89	1169.7	1287.65	29%	-9%	681
W	824.78	164.96	989.74	1149.36	1152.89	1216.96	16%	-6%	929
M	824.78	82.48	907.26	1125.82	1125.82	1255.81	24%	-10%	769
AV	824.78	82.48	907.26	1106.09	1107.16	1225.29	22%	-10%	958
WM	824.78	123.72	948.5	1054.43	1057.28	1257.05	11%	-16%	1009
AVWM	824.78	103.1	927.88	1062.39	1063.63	1239.23	14%	-14%	871
C201									
A	584.28	58.43	642.71	857.86	867.97	862.51	33%	-1%	1129
V	584.28	58.43	642.71	860.06	861.97	889.98	34%	-3%	764
W	584.28	116.86	701.14	856.55	858.52	885.89	22%	-3%	814
M	584.28	58.43	642.71	857.72	857.91	866.95	33%	-1%	838
AV	584.28	58.43	642.71	864.48	870.26	885.89	35%	-2%	860
WM	584.28	87.64	671.92	832.31	835.94	881.27	24%	-6%	789
AVWM	584.28	73.04	657.32	851.95	855.96	892.6	30%	-5%	996
RC101									
A	995.59	99.56	1095.15	1361.97	1363.45	1396.22	24%	-2%	904
V	995.59	99.56	1095.15	1519.48	1521.66	1588.96	39%	-4%	1148
W	995.59	199.12	1194.71	1438.19	1444.22	1521.06	20%	-5%	990
M	995.59	99.56	1095.15	1383.98	1391.03	1468.79	26%	-6%	1134
AV	995.59	99.56	1095.15	1393.96	1397.17	1475.27	27%	-6%	1273
WM	995.59	149.34	1144.93	1332.51	1333.9	1442.61	16%	-8%	952
AVWM	995.59	124.45	1120.04	1393.37	1399.14	1482.93	24%	-6%	766
RC201									
A	655.35	65.54	720.89	933.33	938.44	944.36	29%	-1%	1113
V	655.35	65.54	720.89	947.69	951.66	1022.61	31%	-7%	1031
W	655.35	131.07	786.42	937.76	942.06	1006.55	19%	-7%	1043
M	655.35	65.54	720.89	931.21	932.37	959.17	29%	-3%	805
AV	655.35	65.54	720.89	917.57	919.01	949.6	27%	-3%	787
WM	655.35	98.3	753.65	907.02	908.96	979.94	20%	-7%	964
AVWM	655.35	81.92	737.27	918.96	919.17	960.09	25%	-4%	931
Avg	762.88	95.36	858.23	1069.64	1072.95	1141.56	25%	-6%	943

A.3 Results for Components

Figure A.3 plots the average objective values of ten HALNS runs as a function of runtime when deactivating one component at a time (or activating local improvement, respectively) as described in Section 1.6.2.3. As before, the graphs are generated by varying the stopping condition, $gen^{\text{stop}} = [1, ..., 10]$.

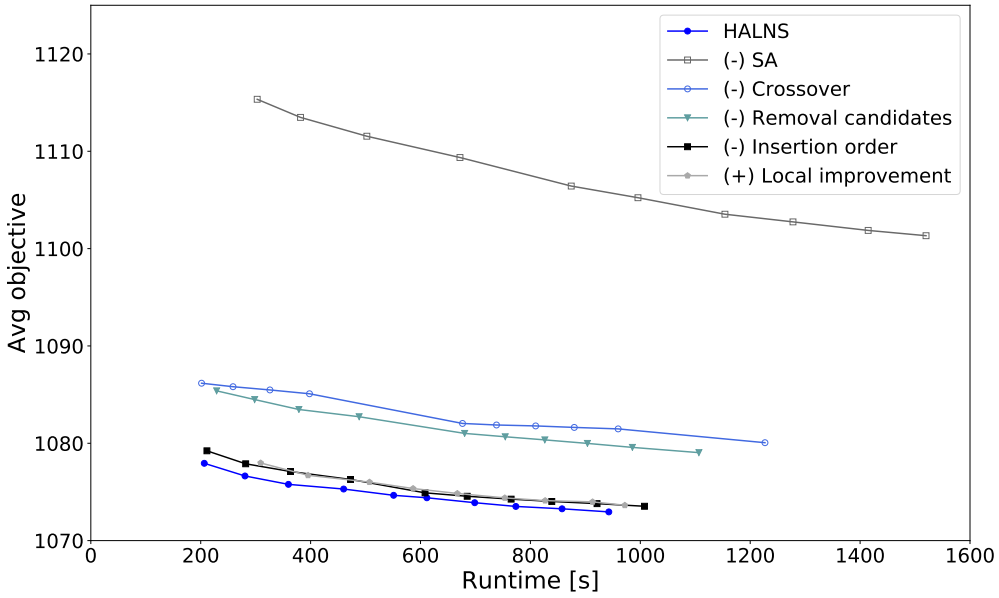


Figure A.3: HALNS performance with and without specific components for VRPAP instances

A.4 UKTUS Availability Profiles

Figure A.4 shows the average APs depending on age and employment status. The x axis shows the time from 8-18 o'clock, while the y axis represents the average availability in percent. It can be observed that the average availability for unemployed people is generally higher compared to employed people and that average availability increases with age in most cases. Furthermore the profiles up to age 60 represent a V-profile with highest availability in the morning and evening hours, whereas elderly people seem to return home during midday and leave again in the afternoon, resulting in a W-profile.

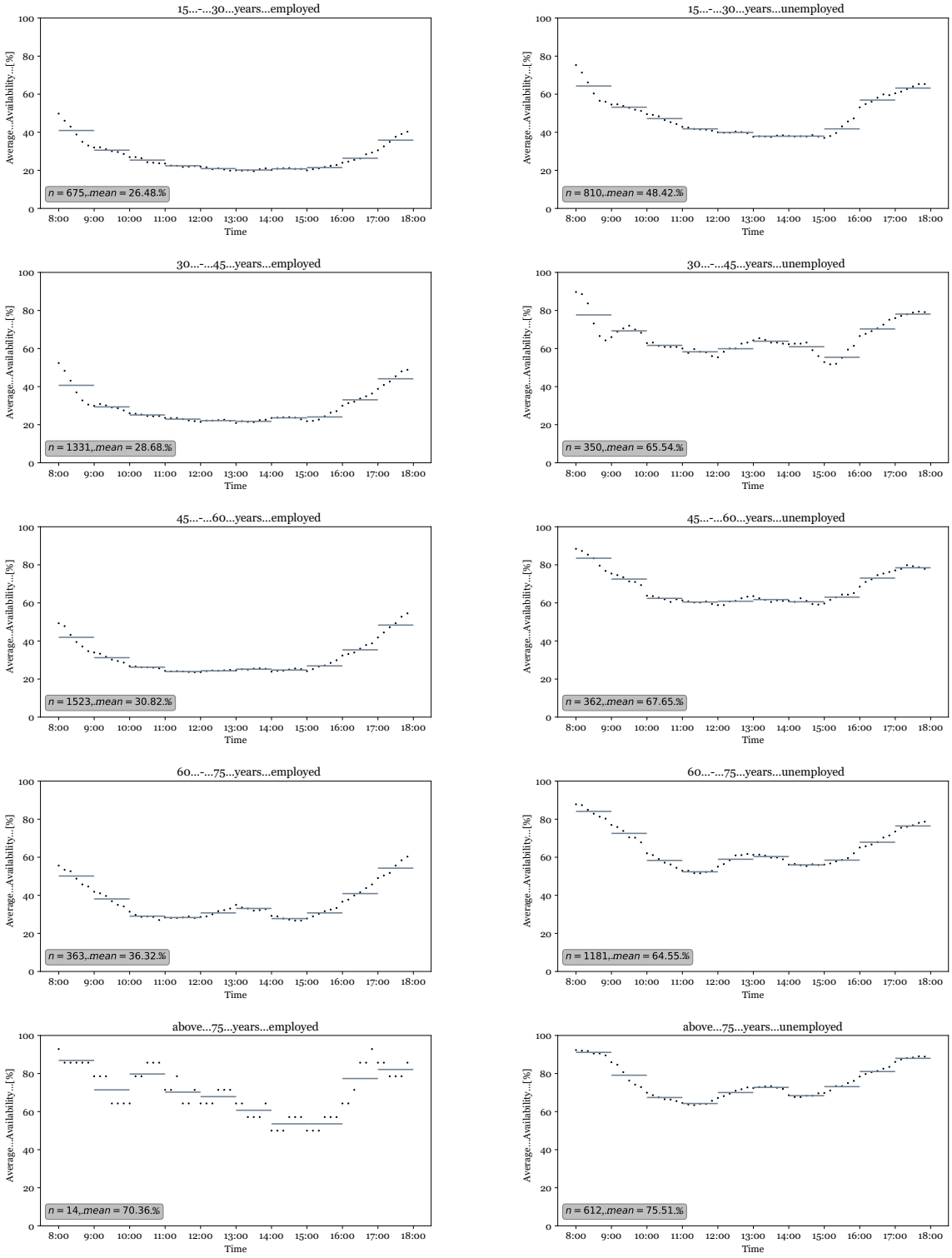


Figure A.4: UKTUS Availability Profiles depending on Age and Work Status

Appendix B

Appendix for Hybrid Adaptive Large Neighborhood Search for Vehicle Routing Problems with Depot Location Decisions

B.1 Parameters

Table B.1: Parameters for HALNS

Parameter	**Meaning**	**Chosen Value**
n^{P}	Size of the initial population	12
$gen^{\max}$	Maximum number of generations	10 in Section 2.5.2 100 in Section 2.5.3 50 in Section 2.5.4
it^{stop}	Number of ALNS iterations without improvement	10,000
α	Cool rate in SA	0.9997
χ_0	Acceptance probability in SA	0.10
n_0	Number of iterations for determining the initial SA temperature	400
ω	Weight for penalties drawn at the beginning of every ALNS run	$unif(0.0001, 0.004) \cdot f(\hat{s})$
p^{binom}	Probability for binomial distribution drawn at the beginning of every ALNS run	$unif(0.10, 0.35)$
σ_1	Score for new best solution	35
σ_2	Score for new best current solution	2
σ_3	Score for worse solution, but accepted via SA	1
γ	Reaction factor for adapting scores for operators and p^{binom}	0.3

B.2 Scaling of Runtimes

Table B.2 tests whether runtimes can reasonably be standardized by using the passmark single thread rating. The table shows the runtimes of the HALNS ($n^{\mathrm{P}} = 12,\ gen^{\max} = 50$) on both CPUs (AMD Ryzen 3900X and AMD Ryzen 2700X) for all three problem classes. The last column shows the actual runtime reduction [%]. The passmark single thread rating seems to be a reasonably good approximation for standardizing runtimes, as the runtime of running all instances is reduced to 91.4% when running on the 3900X vs. 89.3% when estimated via passmark rating.

Table B.2: Analysis of Runtime Scaling Factor

Problem Class	**Runtime [s]**		**Runtime [%]**
	AMD Ryzen 3900X	AMD Ryzen 2700X	
2E-VRP	214	226	94.7
LRP	245	290	84.5
MDVRP	510	556	91.7
All	**252**	**276**	**91.4**

Table B.3 lists the used CPUs with their respective passmark single thread rating (`https://www.cpubenchmark.net/`). Mühlbauer and Fontaine (2021) use 4 threads; we therefore multiply the runtimes by four (1). Not all authors clearly state the CPU used. In these cases we made an assumption based on the information given (2). Vidal et al. (2012) use an AMD Opteron 250 with 2.4 GHz, but report their runtimes as if run on an Intel Pentium 4 with 3 GHz (3). No passmark rating could be found for a small number of CPUs used (4).

Table B.3: CPUs used

Author	CPU	Clock Speed	Rating	Notes
Voigt et al.	AMD Ryzen 9 3900X	3.8 GHz	2731	
Voigt et al.	AMD Ryzen 7 2700X	3.7 GHz	2439	
Wang et al. (2017)	Intel Xeon E5-2670	2.6 GHz	1460	
Amarouche et al. (2018)	Intel Xeon E5-2670v2	2.5 GHz	1592	
Mühlbauer and Fontaine (2021)	Intel i5-6200	2.4 GHz	1600	1
Breunig et al. (2016)	Intel Xeon E5-2670v2	2.5 GHz	1592	
Hemmelmayr et al. (2012)	AMD Opteron 275	2.2 GHz	445	
Enthoven et al. (2020)	Intel Xeon E5-2680v3	2.5 GHz	1774	
Yu et al. (2021)	Intel i7-8700	3.2 GHz	2666	
Schneider and Löffler (2019)	Intel Xeon E5-2670	2.6 GHz	1460	
Arnold and Sörensen (2021)	AMD Ryzen 3 1300X	3.5 GHz	2084	
Contardo et al. (2013)	Intel Xeon E5462	3.0 GHz	1215	
Hemmelmayr et al. (2012)	AMD Opteron 275	2.2 GHz	445	
Lopes et al. (2016)	Intel i7-4790	3.6 GHz	2226	
Escobar et al. (2014a)	Intel Core2 Duo T6400	2.0 GHz	799	2
Ting and Chen (2013)	AMD Athlon XP 2500+	1.83 GHz	353	
Escobar et al. (2013)	Intel Core2 Duo T6400	2.0 GHz	799	2
Yu et al. (2010)	Intel Core2 Quad Q8400	2.6 GHz	1152	2
Duhamel et al. (2010)	Intel Core2 Quad Q9550	2.83 GHz	1228	2
Prins et al. (2007)	Intel Pentium 4	2.4 GHz	360	
Prins et al. (2006a)	Intel Pentium 4	2.4 GHz	360	
Prins et al. (2006b)	Intel Pentium 4	2.4 GHz	360	
Vidal et al. (2012)	Intel Pentium 4	3.0 GHz	561	3
Arnold and Sörensen (2019)	AMD Ryzen 3 1300X	3.5 GHz	2084	
Luo and Chen (2014)	Intel Pentium 4	2.8 GHz	506	2
Subramanian et al. (2013)	Intel Core i7	2.93 GHz	1403	
Pisinger and Ropke (2007)	Intel Pentium 4	3.0 GHz	561	
Sadati et al. (2021)	Intel Core i7-8700	3.2 GHz	2666	
Cordeau and Maischberger (2012)	Intel Xeon X7350	2.93 GHz	1181	
Escobar et al. (2014b)	Intel Core2 Duo T6400	2.0 GHz	799	2
Renaud et al. (1996)	Sun SparcStation 10	NA	100	4
Cordeau et al. (1997)	Sun SparcStation 10	NA	100	4
Chao et al. (1993)	Sun 4/370	NA	100	4

B.3 Detailed Results on Benchmark Instances

Tables B.4 - B.6 present the detailed results for 2E-VRP, LRP and MDVRP benchmark instances. The first column shows the *Instance* name, the second column the (previous) *BKS*, column *Best 5/10* the best result obtained after five or ten runs of the HALNS, and *Avg 5/10* the average results across five or ten runs. *Runtime [s]* shows the average runtime in seconds of the HALNS run on the AMD Ryzen 9 3900X. The second-last row presents the *Avg gap* to the previous BKS and *Avg T* the runtime averaged across all instances of the problem class. The last row *# BKS* counts the number of (current) BKS achieved. **Bold** entries state the BKS, entries additionally marked with an asterisk (*) are improving solutions against the previous BKS. Tables B.7 - B.9 detail the solutions of newly found BKS. The new BKS for LRP instance *P122212* was found during parameter tuning. Please note that customer indices start at $n_s + 1$, e.g., instance *P122212* has ten depots, so the first customer is denoted by $n_s + 1 = 11$.

Table B.4: Detailed results on 2E-VRP instances

Instance	BKS	HALNS				
		Best 5	Avg 5	Best Gap [%]	Avg Gap [%]	Runtime [s]
Set2a_E-n22-k4-s6-17	417.07	**417.07**	**417.07**	0	0	25.96
Set2a_E-n22-k4-s8-14	384.96	**384.96**	**384.96**	0	0	27.96
Set2a_E-n22-k4-s9-19	470.6	**470.6**	**470.6**	0	0	28.03
Set2a_E-n22-k4-s10-14	371.5	**371.5**	**371.5**	0	0	26.59
Set2a_E-n22-k4-s11-12	427.22	**427.22**	**427.22**	0	0	28.52
Set2a_E-n22-k4-s12-16	392.78	**392.78**	**392.78**	0	0	28.41
Set2a_E-n33-k4-s1-9	730.16	**730.16**	**730.16**	0	0	37.84
Set2a_E-n33-k4-s2-13	714.63	**714.63**	**714.63**	0	0	40.82
Set2a_E-n33-k4-s3-17	707.48	**707.48**	**707.48**	0	0	37.06
Set2a_E-n33-k4-s4-5	778.74	**778.74**	**778.74**	0	0	39.66
Set2a_E-n33-k4-s7-25	756.85	**756.85**	**756.85**	0	0	36.99
Set2a_E-n33-k4-s14-22	779.05	**779.05**	**779.05**	0	0	37.08
Set2b_E-n51-k5-s2-17	597.49	**597.49**	**597.49**	0	0	62.86
Set2b_E-n51-k5-s4-46	530.76	**530.76**	**530.76**	0	0	59.79
Set2b_E-n51-k5-s6-12	554.81	**554.81**	**554.81**	0	0	65.6
Set2b_E-n51-k5-s11-19	581.64	**581.64**	**581.64**	0	0	61.65
Set2b_E-n51-k5-s27-47	538.22	**538.22**	**538.22**	0	0	56.73
Set2b_E-n51-k5-s32-37	552.28	**552.28**	**552.28**	0	0	58.8
Set2b_E-n51-k5-s2-4-17-46	530.76	**530.76**	**530.76**	0	0	77.42
Set2b_E-n51-k5-s6-12-32-37	531.92	**531.92**	**531.92**	0	0	105.35
Set2b_E-n51-k5-s11-19-27-47	527.63	**527.63**	**527.63**	0	0	85.39
Set2c_E-n51-k5-s2-17	601.39	**601.39**	**601.39**	0	0	58.94
Set2c_E-n51-k5-s4-46	702.33	**702.33**	**702.33**	0	0	62.27
Set2c_E-n51-k5-s6-12	567.42	**567.42**	**567.42**	0	0	61.74
Set2c_E-n51-k5-s11-19	617.42	**617.42**	**617.42**	0	0	64.43
Set2c_E-n51-k5-s27-47	530.76	**530.76**	**530.76**	0	0	58.71
Set2c_E-n51-k5-s32-37	752.59	**752.6**	**752.6**	0	0	68.19
Set2c_E-n51-k5-s2-4-17-46	601.39	**601.39**	**601.39**	0	0	72.71
Set2c_E-n51-k5-s6-12	567.42	**567.42**	**567.42**	0	0	61.74
Set2c_E-n51-k5-s11-19-27-47	530.76	**530.76**	**530.76**	0	0	70.44
Set3_E-n22-k4-s13-14	526.15	**526.15**	**526.15**	0	0	28.38
Set3_E-n22-k4-s13-16	521.09	**521.09**	**521.09**	0	0	28.96
Set3_E-n22-k4-s13-17	496.38	**496.38**	**496.38**	0	0	26.58
Set3_E-n22-k4-s14-19	498.8	**498.8**	**498.8**	0	0	32.09
Set3_E-n22-k4-s17-19	512.8	**512.8**	**512.8**	0	0	30.93
Set3_E-n22-k4-s19-21	520.42	**520.42**	**520.42**	0	0	29.58
Set3_E-n33-k4-s16-22	672.17	**672.17**	**672.17**	0	0	40.96
Set3_E-n33-k4-s16-24	666.02	**666.02**	**666.02**	0	0	39.23
Set3_E-n33-k4-s19-26	680.36	**680.37**	**680.37**	0	0	37.8
Set3_E-n33-k4-s22-26	680.36	**680.37**	**680.37**	0	0	37.94
Set3_E-n33-k4-s24-28	670.43	**670.43**	**670.43**	0	0	38.84
Set3_E-n33-k4-s25-28	650.58	**650.58**	**650.58**	0	0	38.74
Set3_E-n51-k5-s12-18	690.59	**690.59**	**690.59**	0	0	70.25
Set3_E-n51-k5-s12-41	683.05	**683.05**	683.84	0	0.12	72.84
Set3_E-n51-k5-s12-43	710.41	**710.41**	**710.41**	0	0	61.4
Set3_E-n51-k5-s39-41	728.54	**728.54**	**728.54**	0	0	68.18
Set3_E-n51-k5-s40-41	723.75	**723.75**	724.28	0	0.07	98.94
Set3_E-n51-k5-s40-43	752.15	**752.15**	**752.15**	0	0	73.59
Set3_E-n51-k5-s13-19	560.73	**560.73**	**560.73**	0	0	62.19
Set3_E-n51-k5-s13-42	564.45	**564.45**	**564.45**	0	0	61.64
Set3_E-n51-k5-s13-44	564.45	**564.45**	**564.45**	0	0	61.49

Continued on next page

Table B.4 – *Continued from previous page*

Set3_E-n51-k5-s40-42	746.31	**746.31**	**746.31**	0	0	68.12
Set3_E-n51-k5-s41-42	771.56	**771.56**	**771.56**	0	0	67.28
Set3_E-n51-k5-s41-44	802.91	**802.91**	**802.91**	0	0	75.83
Set4a_Instance50-1	1569.42	**1569.42**	**1569.42**	0	0	59.6
Set4a_Instance50-2	**1438.33**	**1438.32**	1438.32	0	0	63.36
Set4a_Instance50-3	1570.43	**1570.43**	**1570.43**	0	0	58.52
Set4a_Instance50-4	1424.04	**1424.04**	**1424.04**	0	0	57.82
Set4a_Instance50-5	2193.52	**2193.52**	**2193.52**	0	0	69.28
Set4a_Instance50-6	1279.87	**1279.89**	**1279.89**	0	0	56.05
Set4a_Instance50-7	1458.63	**1458.6**	**1458.6**	0	0	54.47
Set4a_Instance50-8	1363.74	**1363.76**	**1363.76**	0	0	58.74
Set4a_Instance50-9	1450.27	**1450.25**	**1450.25**	0	0	55.81
Set4a_Instance50-10	1407.65	**1407.65**	**1407.65**	0	0	56.94
Set4a_Instance50-11	2047.46	**2047.43**	**2047.43**	0	0	68.68
Set4a_Instance50-12	1209.42	**1209.46**	**1209.46**	0	0	61.3
Set4a_Instance50-13	1481.83	**1481.8**	**1481.8**	0	0	66.36
Set4a_Instance50-14	1393.61	**1393.64**	**1393.64**	0	0	63.87
Set4a_Instance50-15	1489.94	**1489.92**	**1489.92**	0	0	62.26
Set4a_Instance50-16	1389.17	**1389.2**	**1389.2**	0	0	58.22
Set4a_Instance50-17	2088.49	**2088.48**	**2088.48**	0	0	101.54
Set4a_Instance50-18	1227.61	**1227.68**	**1227.68**	0.01	0.01	58.3
Set4a_Instance50-19	1564.66	**1564.66**	**1564.66**	0	0	94.04
Set4a_Instance50-20	1272.97	**1272.98**	**1272.98**	0	0	68.38
Set4a_Instance50-21	1577.82	**1577.82**	**1577.82**	0	0	113.89
Set4a_Instance50-22	1281.83	**1281.83**	**1281.83**	0	0	73.3
Set4a_Instance50-23	1807.35	**1807.35**	**1807.35**	0	0	146.68
Set4a_Instance50-24	1282.68	**1282.69**	**1282.69**	0	0	71.11
Set4a_Instance50-25	1522.42	**1522.4**	**1522.4**	0	0	108.17
Set4a_Instance50-26	1167.46	**1167.47**	**1167.47**	0	0	64.15
Set4a_Instance50-27	1481.57	**1481.56**	1481.77	0	0.01	125.24
Set4a_Instance50-28	1210.44	**1210.46**	**1210.46**	0	0	71.09
Set4a_Instance50-29	1722.04	**1722**	**1722**	0	0	150.95
Set4a_Instance50-30	1211.59	**1211.63**	**1211.63**	0	0	62.02
Set4a_Instance50-31	1490.33	**1490.32**	**1490.32**	0	0	99.38
Set4a_Instance50-32	1199	**1199.05**	**1199.05**	0	0	72.38
Set4a_Instance50-33	1508.3	**1508.32**	1508.86	0	0.04	123.17
Set4a_Instance50-34	1233.92	**1233.96**	**1233.96**	0	0	66.05
Set4a_Instance50-35	1718.41	**1718.42**	**1718.42**	0	0	134.33
Set4a_Instance50-36	1228.89	**1228.95**	**1228.95**	0	0	65.75
Set4a_Instance50-37	1528.73	**1528.73**	**1528.73**	0	0	214.77
Set4a_Instance50-38	1169.2	**1169.2**	**1169.2**	0	0	152.93
Set4a_Instance50-39	1520.92	**1520.92**	**1520.92**	0	0	190.49
Set4a_Instance50-40	1199.42	**1199.42**	**1199.42**	0	0	140.67
Set4a_Instance50-41	1667.96	**1667.96**	**1667.96**	0	0	259.01
Set4a_Instance50-42	1194.54	**1194.54**	**1194.54**	0	0	136.1
Set4a_Instance50-43	1439.67	**1439.67**	1440.59	0	0.06	209.83
Set4a_Instance50-44	1045.13	**1045.14**	**1045.14**	0	0	128.74
Set4a_Instance50-45	1450.95	**1450.95**	**1450.95**	0	0	249.41
Set4a_Instance50-46	1088.77	**1088.79**	**1088.79**	0	0	143.61
Set4a_Instance50-47	1587.29	**1587.29**	**1587.29**	0	0	212.73
Set4a_Instance50-48	1082.2	**1082.21**	**1082.21**	0	0	113.16
Set4a_Instance50-49	1434.88	**1434.88**	**1434.88**	0	0	183.78
Set4a_Instance50-50	1083.16	**1083.16**	**1083.16**	0	0	144.94
Set4a_Instance50-51	1398.05	**1398.03**	**1398.03**	0	0	175.27
Set4a_Instance50-52	1125.69	**1125.69**	**1125.69**	0	0	95.33

Continued on next page

Table B.4 – *Continued from previous page*

Set4a_Instance50-53	1567.77	**1567.79**	1568.52	0	0.05	189.32
Set4a_Instance50-54	1127.61	**1127.66**	1132.03	0	0.39	142.55
Set4b_Instance50-1	1569.42	**1569.42**	**1569.42**	0	0	65.73
Set4b_Instance50-2	1438.33	**1438.32**	**1438.32**	0	0	74.04
Set4b_Instance50-3	1570.34	**1570.43**	**1570.43**	0.01	0.01	66.89
Set4b_Instance50-4	1424.04	**1424.04**	**1424.04**	0	0	64.9
Set4b_Instance50-5	2193.52	**2193.52**	**2193.52**	0	0	70.7
Set4b_Instance50-6	1279.87	**1279.89**	**1279.89**	0	0	63.25
Set4b_Instance50-7	1408.57	**1408.58**	**1408.58**	0	0	66.29
Set4b_Instance50-8	1360.32	**1360.32**	**1360.32**	0	0	65.19
Set4b_Instance50-9	1403.53	**1403.53**	**1403.53**	0	0	60.95
Set4b_Instance50-10	1360.56	**1360.54**	**1360.54**	0	0	61.48
Set4b_Instance50-11	2047.46	**2047.43**	**2047.43**	0	0	88.55
Set4b_Instance50-12	1209.42	**1209.46**	**1209.46**	0	0	64.11
Set4b_Instance50-13	1450.93	**1450.94**	**1450.94**	0	0	60.66
Set4b_Instance50-14	1393.61	**1393.64**	**1393.64**	0	0	64.43
Set4b_Instance50-15	1466.83	**1466.84**	**1466.84**	0	0	62.12
Set4b_Instance50-16	1387.83	**1387.85**	**1387.85**	0	0	67.61
Set4b_Instance50-17	2088.49	**2088.48**	**2088.48**	0	0	74.91
Set4b_Instance50-18	1227.61	**1227.68**	**1227.68**	0.01	0.01	62.27
Set4b_Instance50-19	1546.28	**1546.28**	1546.91	0	0.04	96.27
Set4b_Instance50-20	1272.97	**1272.98**	**1272.98**	0	0	69.84
Set4b_Instance50-21	1577.82	**1577.82**	**1577.82**	0	0	97.49
Set4b_Instance50-22	1281.83	**1281.83**	**1281.83**	0	0	74.92
Set4b_Instance50-23	1652.98	**1652.98**	**1652.98**	0	0	78.49
Set4b_Instance50-24	1282.68	**1282.69**	**1282.69**	0	0	71.72
Set4b_Instance50-25	1408.57	**1408.58**	**1408.58**	0	0	70.13
Set4b_Instance50-26	1167.46	**1167.47**	**1167.47**	0	0	66.26
Set4b_Instance50-27	1444.5	**1444.49**	**1444.49**	0	0	105.95
Set4b_Instance50-28	1210.44	**1210.46**	**1210.46**	0	0	71.68
Set4b_Instance50-29	1552.66	**1552.66**	**1552.66**	0	0	96.53
Set4b_Instance50-30	1211.49	**1211.63**	**1211.63**	0.01	0.01	66.07
Set4b_Instance50-31	1440.86	**1440.85**	1441.01	0	0.01	99.36
Set4b_Instance50-32	1199	**1199.05**	**1199.05**	0	0	71.74
Set4b_Instance50-33	1478.86	**1478.87**	**1478.87**	0	0	85.78
Set4b_Instance50-34	1233.92	**1233.96**	**1233.96**	0	0	65.83
Set4b_Instance50-35	1570.72	**1570.73**	**1570.73**	0	0	76.54
Set4b_Instance50-36	1228.89	**1228.95**	**1228.95**	0	0	65.35
Set4b_Instance50-37	1528.73	**1528.73**	**1528.73**	0	0	162.83
Set4b_Instance50-38	1163.07	**1163.07**	**1163.07**	0	0	112.17
Set4b_Instance50-39	1520.92	**1520.92**	**1520.92**	0	0	179.79
Set4b_Instance50-40	1163.04	**1163.04**	1164.76	0	0.15	172.62
Set4b_Instance50-41	1652.98	**1652.98**	**1652.98**	0	0	127.34
Set4b_Instance50-42	1190.17	**1190.17**	**1190.17**	0	0	105.48
Set4b_Instance50-43	1406.11	**1406.1**	**1406.1**	0	0	168.45
Set4b_Instance50-44	1035.03	**1035.05**	**1035.05**	0	0	103.48
Set4b_Instance50-45	1401.87	**1401.87**	**1401.87**	0	0	148.67
Set4b_Instance50-46	1058.11	**1058.1**	**1058.1**	0	0	144.73
Set4b_Instance50-47	1552.66	**1552.66**	**1552.66**	0	0	132.45
Set4b_Instance50-48	1074.5	**1074.51**	**1074.51**	0	0	91.28
Set4b_Instance50-49	1434.88	**1434.88**	**1434.88**	0	0	148.69
Set4b_Instance50-50	1065.25	**1065.3**	**1065.3**	0	0	87.9
Set4b_Instance50-51	1387.51	**1387.51**	**1387.51**	0	0	147.36
Set4b_Instance50-52	1103.42	**1103.47**	1104.17	0	0.07	121.2
Set4b_Instance50-53	1545.73	**1545.76**	**1545.76**	0	0	97.61

Continued on next page

Table B.4 – *Continued from previous page*

Set4b_Instance50-54	1113.62	**1113.66**	**1113.66**	0	0	90.87
Set5_100-5-1	1564.46	1571.43	1572.77	0.45	0.53	809.82
Set5_100-5-1b	1099.35	1109.27	1109.27	0.9	0.9	377.91
Set5_100-5-2	1016.32	**1016.32**	1017.99	0	0.16	491.56
Set5_100-5-2b	782.25	783.39	783.39	0.15	0.15	297.13
Set5_100-5-3	1045.29	**1045.29**	1046	0	0.07	589.98
Set5_100-5-3b	828.54	**828.54**	**828.54**	0	0	279.19
Set5_100-10-1	1124.93	**1124.93**	**1124.93**	0	0	2919.96
Set5_100-10-1b	911.8	913.59	913.59	0.2	0.2	1258.56
Set5_100-10-2	985.4	996.3	996.74	1.11	1.15	1195.51
Set5_100-10-2b	766.28	768.13	768.13	0.24	0.24	750.73
Set5_100-10-3	1042.63	**1042.63**	**1042.63**	0	0	1675.63
Set5_100-10-3b	848.16	**848.16**	849.56	0	0.17	1601.16
Set5_200-10-1	1537.52	1544.1	1545.98	0.43	0.55	8035.98
Set5_200-10-1b	1173.07	1175.27	1178.51	0.19	0.46	3751.62
Set5_200-10-2	1352.87	1353.21	1353.74	0.03	0.06	2947.85
Set5_200-10-2b	985.99	987.44	988.29	0.15	0.23	1635.62
Set5_200-10-3	1777.49	1782.81	1784.61	0.3	0.4	3905.13
Set5_200-10-3b	1192.35	1197.97	1198.67	0.47	0.53	2283.3
Set6_A-n51-4	652	**652**	**652**	0	0	111.32
Set6_A-n51-5	663.41	**663.41**	**663.41**	0	0	136.18
Set6_A-n51-6	662.51	**662.51**	**662.51**	0	0	227.35
Set6_B-n51-4	563.98	**563.98**	**563.98**	0	0	92.84
Set6_B-n51-5	549.23	**549.23**	**549.23**	0	0	113.16
Set6_B-n51-6	556.32	**556.32**	**556.32**	0	0	172.88
Set6_C-n51-4	689.18	**689.18**	**689.18**	0	0	109.08
Set6_C-n51-5	723.12	**723.12**	**723.12**	0	0	108.27
Set6_C-n51-6	697	**697**	**697**	0	0	156.1
Set6_A-n76-4	985.95	**985.95**	986.01	0	0.01	235.59
Set6_A-n76-5	979.15	**979.15**	**979.15**	0	0	472.36
Set6_A-n76-6	970.2	**970.2**	**970.2**	0	0	612.56
Set6_B-n76-4	792.73	**792.73**	**792.73**	0	0	236.2
Set6_B-n76-5	783.93	**783.93**	784.09	0	0.02	351.29
Set6_B-n76-6	774.17	**774.17**	775.5	0	0.17	459.42
Set6_C-n76-4	1054.89	**1054.89**	**1054.89**	0	0	254.82
Set6_C-n76-5	1115.32	**1115.32**	**1115.32**	0	0	377.73
Set6_C-n76-6	1060.52	**1060.52**	1062.85	0	0.22	745.98
Set6_A-n101-4	1194.17	**1194.17**	1194.38	0	0.02	326.11
Set6_A-n101-5	1211.38	1214.41	1214.41	0.25	0.25	699.03
Set6_A-n101-6	1155.89	1155.94	1156.56	0	0.06	959.49
Set6_B-n101-4	939.21	939.83	939.83	0.07	0.07	319.94
Set6_B-n101-5	967.82	969.07	969.07	0.13	0.13	763.59
Set6_B-n101-6	960.29	**960.29**	960.81	0	0.05	688.98
Set6_C-n101-4	1292.04	**1292.04**	1297.26	0	0.4	509.41
Set6_C-n101-5	1304.86	1305.82	1305.82	0.07	0.07	552.44
Set6_C-n101-6	1284.48	**1284.48**	1290.35	0	0.46	796.4
Avg. gap / avg. time			0.00%	0.025%	0.043%	285.8s
# BKS	207/207			190/207		

Table B.5: Detailed results on LRP instances

Instance	BKS	HALNS				
		Best 10	Avg 10	Best Gap [%]	Avg Gap [%]	Runtime [s]
20-5-1a	54793	**54793**	**54793**	0	0	25.61
20-5-1b	39104	**39104**	**39104**	0	0	21.3
20-5-2a	48908	**48908**	**48908**	0	0	23.1
20-5-2b	37542	**37542**	**37542**	0	0	22.61
50-5-1	90111	**90111**	**90111**	0	0	61.47
50-5-1b	63242	**63242**	**63242**	0	0	54.47
50-5-2	88298	**88298**	**88298**	0	0	59.08
50-5-2b	67308	**67308**	**67308**	0	0	58.93
50-5-2bis	84055	**84055**	**84055**	0	0	61.26
50-5-2bbis	51822	**51822**	**51822**	0	0	52.94
50-5-3	86203	**86203**	**86203**	0	0	66.52
50-5-3b	61830	**61830**	**61830**	0	0	54.99
100-5-1	274814	275079	275120.6	0.1	0.11	385.09
100-5-1b	213568	**213568**	213588.6	0	0.01	279.77
100-5-2	193671	**193671**	**193671**	0	0	210.24
100-5-2b	157095	**157095**	**157095**	0	0	186.1
100-5-3	200079	**200079**	**200079**	0	0	247.28
100-5-3b	152441	**152441**	**152441**	0	0	138.93
100-10-1	287661	287692	287870	0.01	0.07	394.79
100-10-1b	230989	**230989**	**230989**	0	0	222.9
100-10-2	243590	**243590**	243611	0	0.01	254.71
100-10-2b	203988	**203988**	**203988**	0	0	146.75
100-10-3	250882	**250882**	250945.6	0	0.03	332.39
100-10-3b	203114	**203114**	203404.6	0	0.14	207.38
200-10-1	474850	476472	477174.2	0.34	0.49	1478.59
200-10-1b	375177	375346	375512.6	0.05	0.09	1031.6
200-10-2	448077	448721	449138.6	0.14	0.24	1283.05
200-10-2b	373696	**373696**	373706.2	0	0	951.11
200-10-3	469433	470422	471149	0.21	0.37	1197.06
200-10-3b	362320	362630	362744.4	0.09	0.12	1091.56
P111112	1467.68	**1467.68**	**1467.68**	0	0	163.22
P111122	1448.37	**1448.37**	1448.54	0	0.01	239.21
P111212	1394.8	**1394.8**	**1394.8**	0	0	162.94
P111222	1432.29	**1432.29**	1432.6	0	0.02	234.86
P112112	1167.16	**1167.16**	**1167.16**	0	0	184.35
P112122	1102.24	**1102.24**	**1102.24**	0	0	151.51
P112212	791.66	**791.66**	**791.66**	0	0	168.88
P112222	728.3	**728.3**	**728.3**	0	0	137.87
P113112	1238.24	1238.49	1238.49	0.02	0.02	197.3
P113122	1245.3	1245.31	1245.46	0	0.01	150.63
P113212	902.26	**902.26**	**902.26**	0	0	141.88
P113222	1018.29	**1018.29**	1020.09	0	0.18	154.53
P131112	1892.17	**1892.17**	1893.86	0	0.09	481.69
P131122	1819.68	**1819.68**	1822.7	0	0.17	473.25
P131212	1960.02	1964.34	1965.4	0.22	0.27	406.93
P131222	1792.77	**1792.77**	1795.23	0	0.14	377.68
P132112	1443.32	**1443.32**	**1443.32**	0	0	368.65
P132122	1429.3	**1429.3**	1439	0	0.68	441.85
P132212	1204.42	**1204.42**	**1204.42**	0	0	427.53
P132222	924.68	**924.68**	**924.68**	0	0	484.31
P133112	1694.18	1694.68	1698.6	0.03	0.26	530.37
P133122	1392.01	**1392.01**	1393.74	0	0.12	473.67

Continued on next page

Table B.5 – *Continued from previous page*

P133212	1197.95	**1197.95**	**1197.95**	0	0	478.82
P133222	1151.37	1151.69	1151.78	0.03	0.04	385.31
P121112	2237.73	2238.59	2245.18	0.04	0.33	856.79
P121122	2137.45	**2137.45**	2140.31	0	0.13	1011.3
P121212	2195.17	**2195.17**	2199.6	0	0.2	858.64
P121222	2214.86	**2214.86**	2216.2	0	0.06	1014.78
P122112	2070.43	**2070.43**	2072.01	0	0.08	861.55
P122122	1685.52	**1685.52**	1685.69	0	0.01	1029.48
P122212	1449.93	1449.62	1449.77	-0.02	-0.01	1067.59
P122222	1082.46	1082.59	1082.59	0.01	0.01	842.09
P123112	1942.23	1949.95	1952.82	0.4	0.55	1047.41
P123122	1910.08	**1910.08**	1913.17	0	0.16	1005.32
P123212	1761.11	1760.2	1761.14	-0.05	0	1118.7
P123222	1390.86	1390.74	1390.79	-0.01	-0.01	663.32
Christ50	565.6	**565.6**	**565.6**	0	0	54.96
Christ75	848.85	**848.85**	**848.85**	0	0	120.92
Christ100	833.4	833.43	833.95	0	0.07	196.76
Das88	355.78	**355.78**	**355.78**	0	0	91.74
Das150	43919.9	**43919.9**	43920.6	0	0	410.93
Gaspelle	424.9	**424.9**	**424.9**	0	0	24.9
Gaspelle2	585.11	**585.11**	**585.11**	0	0	23.67
Gaspelle3	512.1	**512.1**	**512.1**	0	0	29.05
Gaspelle4	562.22	**562.22**	**562.22**	0	0	34.49
Gaspelle5	504.33	**504.33**	**504.33**	0	0	32.52
Gaspelle6	460.37	**460.37**	**460.37**	0	0	35.88
Min27	3062	3062.02	3062.02	0	0	28.55
Min134	5709	**5709**	**5709**	0	0	240.03
Avg. gap / avg. time			0.00%	0.020%	0.067%	388.9s
# BKS	76/79			62 (3 new)/79		

Table B.6: Detailed results on MDVRP instances

Instance	BKS	HALNS				
		Best 10	Avg 10	Best Gap [%]	Avg Gap [%]	Runtime [s]
p01	576.87	**576.87**	**576.87**	0	0	60.02
p02	473.53	**473.53**	**473.53**	0	0	54.47
p03	641.19	**641.19**	**641.19**	0	0	103
p04	1001.04	**1001.04**	**1001.04**	0	0	198.66
p05	750.03	**750.03**	**750.03**	0	0	145.98
p06	876.5	**876.5**	876.62	0	0.01	206.59
p07	881.97	**881.97**	**881.97**	0	0	203.58
p08	4369.95	4375.49	4379.97	0.13	0.23	1599.68
p09	3858.66	3862.16	3865.39	0.09	0.17	1401.68
p10	3629.6	3631.37	3632.57	0.05	0.08	1653.99
p11	3545.18	3546.06	3546.06	0.02	0.02	1032.74
p12	1318.95	**1318.95**	**1318.95**	0	0	100.22
p13	1318.95	**1318.95**	**1318.95**	0	0	85.05
p14	1360.12	**1360.12**	**1360.12**	0	0	84.2
p15	2505.42	**2505.42**	**2505.42**	0	0	334.53
p16	2572.23	**2572.23**	**2572.23**	0	0	252.51
p17	2709.09	**2709.09**	**2709.09**	0	0	239.79
p18	3702.85	**3702.85**	**3702.85**	0	0	1243.84
p19	3827.06	**3827.06**	**3827.06**	0	0	538.79
p20	4058.07	**4058.07**	**4058.07**	0	0	477.34
p21	5474.84	**5474.84**	5480.95	0	0.11	3310.95
p22	5702.16	**5702.16**	**5702.16**	0	0	1309.81
p23	6078.75	**6078.75**	**6078.75**	0	0	1072.26
pr01	861.32	**861.32**	**861.32**	0	0	44.35
pr02	1307.34	**1307.34**	**1307.34**	0	0	124.84
pr03	1803.8	**1803.8**	**1803.8**	0	0	246.15
pr04	2058.31	**2058.31**	2058.82	0	0.02	578.52
pr05	2331.2	**2331.2**	2336.91	0	0.24	1174.63
pr06	2676.3	2677.8	2683.72	0.06	0.28	1709.75
pr07	1089.56	**1089.56**	**1089.56**	0	0	69.25
pr08	1664.85	**1664.85**	**1664.85**	0	0	338.88
pr09	2133.2	**2133.2**	2136.49	0	0.15	726.5
pr10	2867.26	2870.63	2879.33	0.12	0.42	2207.43
Avg. gap / avg. time	0.00%			0.014%	0.053%	694.8s
# BKS	33/33			27/33		

Table B.7: Solution LRP instance *P122212*

LRP - P122212	**BKS**	**1449.03**		
Depot 8	Demand 1928			
Route	Demand	Distance	Duration	Sequence
Route 1	148	63.3482	63.3482	8-101-97-93-153-98-111-123-146-113-128-8
Route 2	148	37.6835	37.6835	8-105-94-119-152-100-129-140-137-136-8
Route 3	149	78.4976	78.4976	8-186-170-207-180-167-187-182-203-209-175-188-8
Route 4	148	105.137	105.137	8-99-104-106-11-174-194-190-178-208-164-8
Route 5	149	78.9653	78.9653	8-184-158-179-199-172-196-161-173-210-189-191-8
Route 6	147	55.6065	55.6065	8-92-121-122-147-88-133-86-116-91-115-8
Route 7	150	45.8852	45.8852	8-84-109-87-103-142-120-143-131-124-102-8
Route 8	150	103.39	103.39	8-165-155-162-198-202-176-157-156-16-201-8
Route 9	150	49.8417	49.8417	8-83-117-114-125-110-145-150-138-96-144-8
Route 10	149	34.1428	34.1428	8-95-118-130-148-107-89-149-132-90-139-8
Route 11	150	65.1197	65.1197	8-112-15-13-108-85-126-151-141-135-134-127-8
Route 12	145	59.4967	59.4967	8-168-206-197-183-171-195-166-192-193-205-8
Route 13	145	63.3845	63.3845	8-163-200-154-177-185-159-204-160-181-169-8
Depot 9	Demand 1009			
Route	Demand	Distance	Duration	Sequence
Route 1	141	22.6563	22.6563	9-26-22-50-56-21-51-72-58-44-9
Route 2	150	69.7896	69.7896	9-36-12-78-14-53-81-68-24-45-77-40-9
Route 3	138	25.9581	25.9581	9-41-17-61-76-71-59-33-64-67-9
Route 4	149	27.1209	27.1209	9-57-35-20-74-75-47-28-34-60-79-9
Route 5	137	19.9485	19.9485	9-18-55-80-42-19-38-25-43-49-73-9
Route 6	148	18.6211	18.6211	9-30-31-82-23-54-52-48-37-66-9
Route 7	146	24.4349	24.4349	9-65-27-69-39-62-29-32-46-63-70-9

Table B.8: Solution LRP instance *P123212*

LRP - P123212	**BKS**	**1760.20**		
Depot 3	Demand 1827			
Route	Demand	Distance	Duration	Sequence
Route 1	142	82.4028	82.4028	3-90-60-77-65-84-78-81-88-95-3
Route 2	132	67.7368	67.7368	3-16-17-24-50-28-23-27-44-3
Route 3	150	85.9019	85.9019	3-20-40-36-42-14-48-46-29-30-52-3
Route 4	150	72.8714	72.8714	3-11-43-18-19-37-13-15-45-49-26-3
Route 5	147	77.7983	77.7983	3-63-66-69-89-79-93-72-74-57-76-3
Route 6	149	68.96	68.96	3-71-82-70-68-80-75-59-92-61-94-3
Route 7	145	69.8052	69.8052	3-56-55-96-73-87-58-86-54-91-3
Route 8	142	56.1821	56.1821	3-186-198-183-196-188-177-199-190-182-197-3
Route 9	148	81.946	81.946	3-33-25-32-21-12-51-62-64-67-85-83-3
Route 10	137	45.8671	45.8671	3-178-184-185-205-193-207-180-191-209-3
Route 11	149	81.2453	81.2453	3-35-22-34-41-31-53-38-39-47-3
Route 12	147	63.2245	63.2245	3-175-201-202-194-187-206-200-204-189-192-203-3
Route 13	89	43.0705	43.0705	3-176-210-179-195-181-208-3
Depot 8	Demand 1191			
Route	Demand	Distance	Duration	Sequence
Route 1	150	47.1888	47.1888	8-107-138-132-126-100-133-121-119-128-8
Route 2	147	74.5361	74.5361	8-166-147-159-154-173-152-153-142-171-174-8
Route 3	148	65.6967	65.6967	8-151-146-169-162-157-148-163-149-150-155-8
Route 4	150	49.3432	49.3432	8-106-97-124-139-135-136-134-103-114-8
Route 5	149	32.2056	32.2056	8-125-122-108-102-130-98-116-112-137-8
Route 6	149	35.6204	35.6204	8-127-110-113-141-140-118-99-123-120-131-8
Route 7	150	68.8255	68.8255	8-144-143-172-156-168-145-164-167-160-158-165-8
Route 8	148	79.7698	79.7698	8-129-104-111-115-117-105-101-109-170-161-8

Table B.9: Solution LRP instance *P123222*

LRP - P123222	**BKS**	**1390.74**		
Depot 3	Demand			
	594			
Route	Demand	Distance	Duration	Sequence
Route 1	149	50.2523	50.2523	3-105-110-140-119-136-129-137-139-133-128-3
Route 2	149	44.5297	44.5297	3-104-122-135-115-113-111-121-142-114-123-3
Route 3	148	29.2728	29.2728	3-109-118-107-124-130-131-117-108-116-126-3
Route 4	148	34.4071	34.4071	3-106-132-141-125-127-138-112-134-120-3
Depot 4	Demand			
	500			
Route	Demand	Distance	Duration	Sequence
Route 1	78	10.1649	10.1649	4-204-203-208-202-206-4
Route 2	139	29.1737	29.1737	4-193-196-195-209-218-214-188-194-213-4
Route 3	139	28.4829	28.4829	4-191-220-207-189-211-210-212-200-197-4
Route 4	144	24.9624	24.9624	4-215-219-198-216-201-205-190-192-199-217-4
Depot 13	Demand			
	591			
Route	Demand	Distance	Duration	Sequence
Route 1	146	19.0509	19.0509	13-77-81-100-89-80-68-98-85-86-13
Route 2	150	46.464	46.464	13-88-102-69-97-87-72-91-103-90-94-96-13
Route 3	150	24.6196	24.6196	13-70-67-82-71-73-76-78-84-93-13
Route 4	145	40.7973	40.7973	13-65-79-99-74-75-101-83-66-92-95-13
Depot 19	Demand			
	690			
Route	Demand	Distance	Duration	Sequence
Route 1	147	44.2391	44.2391	19-152-186-153-146-180-160-148-167-184-183-19
Route 2	137	27.384	27.384	19-157-176-156-155-168-158-159-165-19
Route 3	140	33.505	33.505	19-162-182-164-178-154-161-171-173-185-19
Route 4	148	46.706	46.706	19-144-175-187-163-145-170-172-147-174-151-19
Route 5	118	30.0726	30.0726	19-166-179-149-150-143-169-177-181-19
Depot 20	Demand			
	683			
Route	Demand	Distance	Duration	Sequence
Route 1	150	27.9864	27.9864	20-61-24-34-40-46-25-52-43-63-62-64-20
Route 2	143	26.132	26.132	20-22-47-36-26-54-35-32-53-33-48-20
Route 3	137	18.2386	18.2386	20-39-57-50-59-21-27-49-41-20
Route 4	138	21.8631	21.8631	20-51-30-31-29-28-44-42-55-20
Route 5	115	12.4397	12.4397	20-23-56-37-45-58-38-60-20