University of Southampton

Faculty of Social Sciences
School of Mathematical Sciences

The Truck-Porters Routing Problem

by

Mohammed Alammar

ORCID: 0009-0003-8633-4318

Thesis for the degree of Doctor of Philosophy

July 2023

Contents

Declaration of Authorship

I, Mohammed Alammar, declare that the thesis entitled "Truck-Porters Routing Problem" and the work presented in it is my own and has been generated by me as the result of my own original research. I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission;

Signature: Date:

Acknowledgements

My deep gratitude goes first and foremost to Professor Chris Potts, Doctor Antonio Martinez-Sykora, and Doctor Stefano Coniglio all of whom expertly guided and patiently supported me to becoming the researcher I am today. This thesis would not be completed without their constant encouragement and comprehensive advices during the research. I am truly grateful to every member of staff that I met at the University of Southampton for their help and support provided for me from my first day at the university.

I wish to express my heartfelt gratitude, appreciation, and warmest affection to my beloved parents, my dear wife, my beautiful daughter, and my family for their endless support and love through the duration of my studies. My appreciation and gladness also go to the government of the Kingdom of Saudi Arabia for giving me the chance to complete a Doctor of Philosophy degree at one of the best universities in the world. I would certainly be remiss to not mention and thank the sponsorship authority, Shaqra University, for giving me this opportunity to achieve my dream.

Chapter One

Introduction

1.1 Motivation

The movement of goods from a transportation hub to the final delivery destination is known as last-mile delivery. The growth of population and so demand for goods in urban areas considerably increases causing environmental pollution and traffic congestion. Furthermore, the increase of road taxes, congestion charge, in urban areas impacting the total cost of a product. According to Van Goor (1980), transportation costs often form a considerable part of the total cost of a product and represent up to 10% of the final price of a product (Coyle et al., 1996). Last mile logistics is the least efficient stage of the supply chain that causes up to 28% of the total transportation costs (Rodrigue et al., 2016). Thus, delivering items efficiently in urban areas plays a crucial role in total costs of supply chains.

The Dutch dairy industry uses truck-trailer combinations for the distribution of final products. The products have to be delivered to customers located in varies regions including crowded cities. This implies that serving them with a truck-trailer combination may required much more time than serving them using the truck alone. Therefore, the trailer is often parked while the truck serve some customers (Gerdessen, 1996). In fact, this is an application of the well-known *truck and trailer routing problems* (TTRPs). In the simplest and most studied version of the TTRPs, the *capacitated TTRPs* (CTTRPs), a homogeneous limited fleet of capacitated trucks and trailers are available at a depot where the fright originates. A set of customers has to be served and some of the customers can only be reached by a truck without a trailer, called *truck customers*. The rest of customers can be visited either by a truck alone or by a truck pulling a trailer, called *vehicle customers*. In order to serve these two types of customers, three types of routes can be planned: *pure truck routes*, *pure vehicle routes*, and *complete vehicle routes*.

A pure truck route is a route that is carried out by a truck alone. A pure vehicle route visits vehicle customers by a complete vehicle without any sub-tour. Finally, a complete vehicle route consists of a main tour, starting and ending at the depot and travelled by a complete vehicle, and one or more sub-tours travelled by the truck alone. Each sub-tour starts and ends at a given location visited in the main tour, called *transshipment location*, where the trailer is temporarily parked and where a load transfer from a truck to its trailer can be performed. Then, the truck alone serves some truck customers and returns to the transshipment location. The trailer is attached to the truck and continue its main tour.

In the CTTRP, the transshipment locations correspond to customer sites. However, a generalised version of the CTTRP gives the option of decoupling a trailer from a truck in locations that do not necessarily correspond to a customer site. The aim of solving the TTRP is to determine the optimal set of vehicles routes such that each customer is visited by a compatible vehicle while the total cost of the system is minimised. Several heuristic algorithms to solve the CTTRP are proposed in the literature (Caramia and Guerriero, 2010, Chao, 2002, Lin et al., 2009b, Scheuerer, 2006, Villegas et al., 2011, 2013). The CTTRP with time windows, also, has received some attention (Derigs et al., 2013, Lin et al., 2011). So far, one exact method has been proposed for the generalisation of the CTTRP by Drexl et al. (2011). For a detailed review about the TTRP and its variants, we refer the reader to the survey paper of Cuda et al. (2015).

Motivated by the idea of decoupling a trailer from a truck at a transshipment location in the TTRP, and for the aim of designing an efficient distribution system to deliver parcels in central cities with less number of delivery vehicles entering central cities, we introduce the *truck-porters routing problem* (TPRP). The TPRP combines driving and walking to serve customers located in urban areas for a single driver and one transshipment location. In the TPRP, a truck and a trailer are available at the depot. The truck is allowed to carry heavy and light items while the trailer is allowed to carry light items only. The trailer is then attached to the truck before the truck-driver departs from the depot towards the transshipment location, or just the depot (assuming that there is no customers served by the truck-trailer combination), where the trailer is temporarily parked and a pre-determined number of porters are waiting. At this stage, transferring heavy items from the truck to the trailer is not permitted. However, light items can be transferred from the truck to the trailer and vice versa.

In the TPRP, a single truck and a limited number of identical porters are available at the depot to undertake deliveries in which some customers must be visited by the truck, called *truck customers*, some must be served by a porter, called *porter customers*, and the remainder can be visited either by the truck or by a porter, which we refer to as *unconstrained customers*. Porters are limited by the total weight of items that they can carry and by a total working time constraint. However, a porter can revisit the depot to collect further items for delivery. The TPRP problem consists of designing a set of minimum-cost routes, where each route starts and ends at the depot and satisfies capacity and travel time constraints. At any feasible solution of the TPRP, there are two types of routes: a truck's route where customers are visited by the truck, and a porter's route where customers are visited by a porter. To illustrate that, there are four possible routes:

1. a route covered by the truck visiting truck customers and/or unconstrained customers;

2. a route covered by the truck visiting truck customers only;

3. a route covered by a porter visiting porter customers only;

4. a route covered by a porter visiting porter customers and/or unconstrained customers.

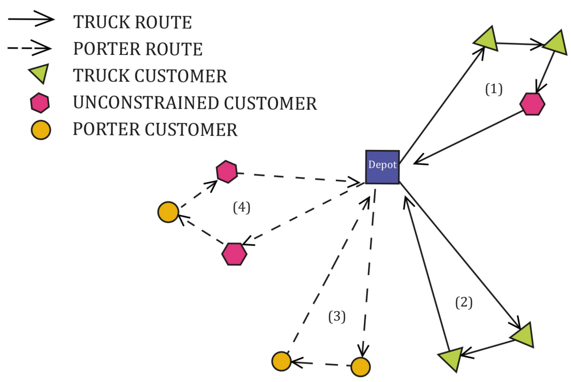

Figure 1.1: Illustration of the four possible routes in any TPRP feasible solution where the square refers to the parking location (the depot), triangles correspond to truck customers, circles referred to porter customers and octagons are unconstrained customers.

Combining walking and driving for last-mile parcel deliveries has recently been studied in the literature (Allen et al., 2020, Martinez-Sykora et al., 2020, McLeod et al., 2020, Nguyen et al., 2019). However, we believe that this is the first work that considers three different types of customers in which some of them must be served by porters (walking), some of them must be served by the truck (driving), and the remaining customers can be visited either by the truck or by a porter.

1.2 Objectives

The objectives of this thesis are as follows:

- Introduce the truck-porters routing problem (TPRP);
- Review some vehicle routing problem (VRP) variants and present the main formulations and solution techniques for VRPs;
- Propose two mathematical formulations and several families of valid inequalities for the TPRP;
- Create a set of instances for the TPRP sampled from a VRP instance;
- Design and implement a branch-and-cut algorithm for the TPRP;
- Design and implement a variable neighborhood search (VNS) algorithm for the TPRP;
- Introduce the truck-porters routing problem with satellites (TPRPS);
- Design and implement a VNS algorithm for the TPRPS.

1.3 Thesis outline

The remaining chapters of this thesis are structured as follows.

- Chapter 2 gives an overview of some VRP variants and extensions. The focus is on variants that some resemblance to the TPRP. The chapter also includes formulations and solution techniques that are commonly used to tackle VRPs.
- Chapter 3 presents a branch-and-cut algorithm for the TPRP. Specifically, it presents:
    1. two mathematical formulations for the TPRP;
    2. several families of valid inequalities that can be included in any formulation in order to strengthen its linear relaxation;
    3. a tabu search algorithm designed and used to solve the separation problem of the capacity constraints and valid inequality families;
    4. an explanation of the method used to create a set of small-size instances; and

5. computational results to assess the performance of the branch-and-cut algorithm.

- Chapter 4 contains a VNS algorithm for the TPRP. It presents:
  1. the designed VNS algorithm for the TPRP;
  2. an explanation of the method used to create the set of large-size instances; and
  3. computational results of this chapter.
- Chapter 5 contains a VNS algorithm for the TPRPS. In this chapter:
  1. the TPRPS is introduced;
  2. the designed VNS algorithm for the TPRPS is presented;
  3. a real-world instance is considered and solved; and
  4. the importance of the TPRPS is demonstrated.
- Chapter 6 concludes the work done in this thesis.

Chapter Two

An overview of the vehicle routing problems

2.1 Introduction

Dantzig and Ramser (1959) introduced the vehicle routing problem (VRP) under the name of "The truck dispatching problem" with the aim of determining an optimal routing plan for a fleet of homogeneous trucks to deliver gasoline to gas stations. Five years later, Clarke and Wright (1964) proposed an effective greedy heuristic for the approximation of the VRP called the savings algorithm. Subsequently, an enormous number of papers have been published in international operational research and transportation journals, presenting mathematical models and proposing exact and (meta)heuristic algorithms for the optimal and approximate solution of different VRP variants. However, most of the well-studied VRP variants are different from the one introduced by Dantzig and Ramser (1959) and Clarke and Wright (1964) as they considered real-world features such as time-dependent travel times (reflecting traffic congestion), time windows for pickup and delivery, and input information that changes dynamically over time. Such features add substantial complexity to the VRP which is considered as an $\mathcal{NP}$-hard problem of combinatorial optimisation (Lenstra and Rinnooy Kan, 1981).

The VRP was born more than sixty years ago and, thus, in order to give a brief review of the problem, we subdivided this chapter into four sections. Section 2.2 presents important VRP variants that have been summarised and fruitfully studied in the literature including VRP with limited capacity, time windows, backhauls, heterogeneous fleet, multiple depot, split deliveries, and stochastic demands. Variants are ordered on the basis of their influence in the VRP literature. In order to assess their influence, we use the state-of-the-art taxonomic review introduced by Braekers et al. (2016) since it is the most recent. Other variants with environmental considerations, which have recently received a great attention in the literature of the VRP because of their positive environmental and ecological impacts, are also presented in this chapter including the pollution routing problem and the VRP in reverse logistics. The electric, multi-trip, and two-echelon VRP are also reviewed. Because the problem has a long research history, an extensive survey would be inappropriate. Thus, Section 2.2 briefly introduces the definition, application, classification, and related noteworthy articles for each variant. Section 2.3 describes the three main formulations to model VRPs, namely the vehicle flow formulations, the commodity flow formulations, and the set partitioning formulations. Section 2.4 reviews exact techniques for the VRPs including the branch-and-bound, branch-and-cut, and branch-and-price algorithms. Section 2.5 reviews some popular heuristic and metaheuristic algorithms for the VRP and its variants.

2.2 Important VRP variants

2.2.1 Capacitated VRP

The VRP is one of the most widely studied topics in the field of operational research and its often defined with capacity and/or route length restrictions. When capacity constraints are present, the problem known as the capacitated vehicle routing problem (CVRP) which is the most studied version of the VRP. The study by Braekers et al. (2016) finds 277 articles in the VRP literature published between 2009 and 2015, with more than 90% of these articles considering the CVRP. The CVRP consists of establishing routes with minimum cost, defined as the sum of the costs of the arcs belonging to the circuits, for identical limited-capacity vehicles such that each vehicle starts and ends its route at the depot, each customer is visited exactly once by a vehicle and the sum of all demands on any route does not exceed the vehicle capacity. Although the CVRP is a generalisation of the well-known traveling salesman problem (TSP), the basic version of the former, the CVRP, appears to be much more difficult to solve involving the same number of cities (Laporte et al., 1986). Several integer programming formulations such as two-index and three-index vehicle-flow formulations, single-commodity, two-commodity and multi-commodity flow formulations, and set partitioning formulations have been proposed to solve the CVRP (Laporte, 1992, 2009, Letchford and Salazar-González, 2006, 2015, Semet et al., 2014). Among these various approaches, the most successful exact algorithms for the CVRP are based on set partitioning formulations augmented with different families of cutting planes (Baldacci et al., 2008b, Fukasawa et al., 2006, Pecin et al., 2017, Poggi and Uchoa, 2014). However, large-size instances usually cannot be solved to optimality using exact methods due to the high computational complexity (Baldacci et al., 2010). Thus, heuristic and metaheuristic approaches have been developed to tackle this problem and find good, but not necessarily guaranteed optimal, solutions within reasonable amount of computing times. Furthermore, the majority of studies about the CVRP in the literature focus on heuristic, metaheuristic or hybrid methods including evolutionary algorithms (Baker and Ayechew, 2003, Berger and Barkaoui, 2003, Nagata and Bräysy, 2009, Prins, 2004), ant colony optimisation (Reimann et al., 2004, Yu et al., 2009), simulated annealing (Lin et al., 2009a, Osman, 1993), tabu search (Cordeau et al., 2001, Gendreau et al., 1994, Taillard, 1993, Toth and Vigo, 2003), path-relinking (Ho and Gendreau, 2006), adaptive memory procedures (Rochat and Taillard, 1995, Tarantilis, 2005, Tarantilis and Kiranoudis, 2002), large neighborhood search (Ergun et al., 2006, Pisinger and Ropke, 2007), variable neighborhood search (Chen et al., 2010, Kytöjoki et al., 2007), deterministic annealing (Golden et al., 1998, Li et al., 2005), honey-bees mating optimisation (Marinakis et al., 2010), hybrid Clarke and Wright's savings heuristic (Juan et al., 2010), and artificial bee colony (Brajevic, 2011, Simsir and Ekmekci, 2019, Szeto et al., 2011).

2.2.2 VRP with time windows

The vehicle routing problem with time windows (VRPTW) is an extension of the CVRP where the service at any customer starts within a given time interval, so-called a time window. During service, the vehicle must remain at the customer location. There are two types of time windows extensively studied in the literature, soft time windows and hard time windows. The soft time windows present a trade-off between not violating a time window or incurring a penalty cost (e.g., dial-a-ride problems). By contrast, the hard time windows, which has been used more widely, must be satisfied. In the latter case, if a vehicle arrives too early at a customer, it must wait until the customer is ready to begin service, and the vehicle cannot arrive late. Important applications for hard time windows including bank and postal deliveries, industrial refuse collection, school bus services, security patrol service, and urban newspaper distribution. It seems that hard time-window constraints naturally model many real-world situations, thus explaining their wide usage. No heuristic approaches for the VRP with time windows or due dates appeared until Russell (1977) proposes an effective heuristic for the M-tour traveling salesman problem in which m salesman are used to visit customers. Before that, the VRPTW only appears in case studies (Knight and Hofer, 1968, Madsen, 1976, Pullen and Webb, 1967). More recently, the literature on the VRPTW has grown substantially to become the second most studied variant of the VRP with 37.92% of the total published articles between 2009 and 2015 (Braekers et al., 2016). The same study indicates that 30.58%, 5.81% and 1.53% of these articles considered hard time windows, a mix of hard and soft time windows and soft time windows, respectively. An overview of research on the VRPTW prior to 2014 is provided by Desaulniers et al. (2014).

2.2.3 VRP with backhauls

The vehicle routing problem with backhauls (VRPB), also known as the linehaul-backhaul problem, is an extension of the CVRP in which the customers are partitioned into two subsets. The first subset contains the linehaul customers requiring deliveries, who are also known as delivery customers. The second subset consists of the backhaul customers requiring pick-ups, who are also known as pickup customers. In the VRPB, the quantities to be delivered and picked up are fixed and known in advance, and each vehicle does deliveries as well as pick-ups in one route starting and ending at the depot. The aim is to design a set of routes where all deliveries for each route are completed before any pickups are made and the vehicle capacity cannot be exceeded by either the linehaul customers or the backhaul customers associated to the route. Taking an advantage of a vehicle going back to the depot with an empty capacity by visiting some pickups customers before arriving to the depot has contributed in reducing distribution costs to the industry. For example, transportation costs and total distance travelled are decreased significantly by employing milk run logistics which is a concept derived from the VRPB (Brar and Saini, 2011).

The VRPB was first introduced by Deif and Bodin (1984). The authors developed a heuristic algorithm based on an extension of the Clarke and Wright savings algorithm (Clarke and Wright, 1964). Subsequent developments include heuristics (Goetschalckx and Jacobs-Blecha, 1989, Toth and Vigo, 1999), metaheuristics (Brandão, 2016, Osman and Wassan, 2002, Zachariadis and Kiranoudis, 2012), and exact algorithms (Mingozzi et al., 1999, Toth and Vigo, 1997). The most recent exact algorithm is proposed by Queiroga et al. (2020). They develop two branch-cut-and-price algorithms that are capable of solving to optimality instances with up to 200 customers.

The VRPB arises in various applications such as in the grocery industry where groceries are delivered to grocery stores from a distribution centre and groceries are picked up at production sites and brought to the distribution centre. Another important application is the handling of returnable bottles, where full bottles are delivered to customers and empty bottles are picked up to be returned to factories for recycling. There are many extensions of the VRPB such as the mixed VRPB, the multiple depot mixed VRPB, the VRPB with time windows, the mixed VRPB with time windows, and the VRP with simultaneous deliveries and pickups. The study of Ropke and Pisinger (2006) reviews the standard VRPB and other mentioned variants and proposes a unified adaptive large neighborhood search heuristic, which is the first unified heuristic capable of solving a large class of VRPBs. A recent comprehensive and up-to-date review of the existing literature on the VRPB and its variants by Koç and Laporte (2018) includes models, exact and heuristic algorithms, industrial applications and case studies.

2.2.4 Heterogeneous VRP

A fleet of vehicles that is characterised by different carrying capacities, speeds, costs or carbon-emission amounts is called a fleet of heterogeneous vehicles. The heterogeneous VRP (HVRP), also called mixed fleet VRP, is a variant of the VRP and it has been known since the early VRP literature. The HVRPs have received much attention due to most real-life distribution problems having customers who are served by a heterogeneous fleet of vehicles (Hoff et al., 2010). Indeed, operating a homogeneous fleet in industry is very rare as the fleet is often acquired over a long period of time and, thus, vehicles in most cases have different characteristics due to the development of technology and changing market requirements.

There are two major classes of the HVRPs. The first one is the fleet size and mix vehicle routing problem (FSM) that is introduced by Golden et al. (1984), which operates with an unlimited fleet, while the second class is the heterogeneous fixed fleet vehicle routing problem (HF) that is introduced by Taillard (1999), which assumes a predetermined fleet. Unified exact algorithms to solve both the FSM and the HF are proposed in the literature (Baldacci et al., 2010, Baldacci and Mingozzi, 2009, Choi and Tcha, 2007). Heuristic methods for the FSM (Brandão, 2009, Liu et al., 2009), and for the HF (Brandão, 2011, Euchi and Chabchoub, 2010) have been developed in the HVRP literature. A servey of HVRPs by Baldacci et al. (2008a) covers the main result dating back to 2007. An updated brief review on HVRPs covering publications in the period 2008 to 2014 is provided by Irnich et al. (2014). More recently, Koç et al. (2016) present a classification and an up-to-date review of the existing literature on HVRPs.

2.2.5 Multiple depot VRP

In many real-life applications, goods must be delivered from more than one depot under some restrictions such as capacity or time window constraints. The multiple-depot VRP (MDVRP) consists of designing a set of routes with minimum cost to serve each customer by a vehicle that is assigned to one of these depots such that each vehicle departs from, and later returns to the same depot. Documented examples of the MDVRP include the delivery of meals, chemical products, soft drinks, machines, industrial gasses, and packaged food.

The MDVRP was first studied by Tillman (1969) and since then, various extensions have been discussed in the literature, including the MDVRP with time windows (Dondo and Cerdá, 2007, Giosa et al., 2002, Polacek et al., 2004), with backhauls (Min et al., 1992, Salhi and Nagy, 1999), with pickup and delivery (Nagy and Salhi, 2005), with mixed fleet (Salhi et al., 2014, Salhi and Sari, 1997), and multi-depot location routing problem (Wasner and Zäpfel, 2004, Wu et al., 2002).

Solution methods proposed in the literature for the MDVRP include exact methods (Benavent and Martínez, 2013, Braekers et al., 2014), heuristics (Gulczynski et al., 2011), and metaheuristics (Dondo and Cerdá, 2009). The only survey on the MDVRP is that of Montoya-Torres et al. (2015) who consider papers published between 1988 and 2014. They find that the number of publications on the MDVRP has increased significantly over the years. They observe that exact algorithms (branch and bound, mathematical programming) are employed in 25% of the reviewed papers, while the remaining 75% focus on heuristics or metaheuristics.

2.2.6 Split deliveries VRP

In the split deliveries vehicle routing problem (SDVRP), a customer's demand can be split among several vehicles. In other words, visiting any customer several times is possible and the demand of any customer can be greater than the vehicle capacity. The SDVRP is a relaxed version of the CVRP and it was first introduced by Dror and Trudeau (1989) who show that there can be savings generated by allowing split deliveries. These savings can reach up to 50% as shown by Archetti et al. (2006a). An empirical study by the same authors show how the savings depend on the characteristics of the instance (Archetti et al., 2008). They demonstrate how the value of customer demands with respect to the vehicle capacity has the largest influence on the saving made, especially when the average customer demand is slightly over half the vehicle's capacity and the variance of customer demand is relatively small.

Several variants of the SDVRP are introduced in the literature such as SDVRP with time windows, which has received the greatest attention (Desaulniers, 2010, Ho and Haugland, 2004), heterogeneous SDVRP (Tavakkoli-Moghaddam et al., 2007) and SDVRP with stochastic demands (Bouzaiene-Ayari et al., 1993). Other variants are reviewed in detail in a survey paper by Archetti and Speranza (2012). The two-stage local search algorithm developed by Dror and Trudeau (1989) is considered as the first heuristic method for the SDVRP. Subsequent studies propose exact approaches (Dror et al., 1994, Jin et al., 2007, Lee et al., 2006, Sierksma and Tijssen, 1998), hybrid methods and metaheuristics (Aleman and Hill, 2010, Archetti et al., 2006b, Berbotto et al., 2014, Chen et al., 2007, Jin et al., 2008), and heuristics (Chen et al., 2014, Gulczynski et al., 2010, Wang et al., 2014). A comprehensive discussion on heuristic solution approaches for the SDVRP is presented by Archetti and Speranza (2012). Real-life applications of split deliveries can be found in newspaper logistics (Song et al., 2002), food distribution (Ambrosino and Sciomachen, 2007), feed distribution in a large livestock ranch (Mullaseril et al., 1997), and waste collection (Archetti and Speranza, 2004).

2.2.7 Stochastic VRP

Stochastic vehicle routing problem (SVRP) arises whenever some elements of the problem, such as customer demands or travel times are random (Gendreau et al., 1996). In most real-world applications, uncertainty is an inherent characteristic of the problem and the probability theory is the main tool to represent the uncertainty in mathematical models in this context. Gendreau et al. (1996) propose a classification according to the stochastic parameters, and Sahinidis (2004) summarise various optimisation problems with uncertainty. In most stochastic problems studied, there is only a single vehicle, which is probably due to the complexity of these problems. There are many variants in the literature of SVRP, however, with the most common cases being: VRP with stochastic customers (Bertsimas, 1992), stochastic demand (Dror et al., 1993, Mendoza et al., 2010, Tillman, 1969) and stochastic travel time (Lambert et al., 1993). A survey by Cordeau et al. (2007) covers these SVRP variants. Applications of the SVRP arise in a number of settings such as delivery of meals on wheels (Bartholdi III et al., 1983) delivery of home heating oil (Dror et al., 1985), of sludge disposal (Larson, 1988), forklift truck routing in warehouses (Bertsimas, 1992), money collection in bank branches (Lambert et al., 1993), and general pickup and delivery operations (Hvattum et al., 2006).

2.2.8 Pollution routing problem

The pollution routing problem (PRP) is an extension of the CVRP with a more comprehensive objective function that considers travel distance, the amount of greenhouse gas emissions, fuel, travel times and their costs. Bektaş and Laporte (2011) introduce the PRP with the aim of choosing a vehicle dispatching scheme with less pollution, especially with a reduction of carbon emissions. In the PRP, vehicle load and speed may change from one leg of the route to another, but all other parameters remain constant on a given leg.

The PRP has been extensively studied and quickly extended in the literature. The variants include bi-objectives (Demir et al., 2014), heterogeneous vehicles (Koç et al., 2014), time-dependent travel (Franceschetti et al., 2017), and the Steiner PRP (Raeesi and Zografos, 2019). A recent survey on the green VRP by Lin et al. (2014) covers studies of the PRP during the period 2007 to 2012. Their paper also suggeste some possible future research directions for the green VRP. Asghari et al. (2021) provide a state-of-the-art review of the green VRP. A systematic literature review by Moghdani et al. (2021) covers freight transportation with green VRPs.

2.2.9 VRP in reverse logistics

Any operations related to the reuse of products and materials belong to reverse logistics. Dekker et al. (2013) defined reverse logistics as “The process of planning, implementing and controlling backward flows of raw materials, in process inventory, packaging and finished goods, from a manufacturing, distribution or use point, to a point of recovery or point of proper disposal”. Carter and Ellram (1998) provided an overview of reverse logistics. The VRP in reverse logistics (VRPRL) concerns about the distribution aspects of reverse logistics. Based on the evidence of a large number of publications, reverse logistics has received much attention with large amount of publications over the past two decades. However, there are only a few studies on reverse logistics from the perspective of vehicle routing (Lin et al., 2014).

The VRP can be utilised to formulate reverse logistics problems. Beullens et al. (2004) discuss the collection (reverse) and vehicle routing systems that link the chain with the market, the VRPRL. The majority of VRPRL studies seen in the literature were mainly focus on the recycling waste or end-of-life goods to one or multiple depots for further reprocessing. A review paper of the existing literature of VRPRL is provided by Lin et al. (2014). In their paper, the authors subdivide the problem into four categories; selective pickups with pricing, waste collection, end-of-life goods collection, and simultaneous distribution and collection. A more recent review paper of Sar and Ghadimi (2023) investigates the state-of-the-art by focusing on VRPRL articles published between 2000 and 2022. Both review papers suggest some further research directions in the VPRRL.

2.2.10 Electric VRP

In the electric vehicle routing problem (EVRP) a fleet of electric vehicles (EVs) are used instead of internal combustion engine vehicles (ICEVs). It was first introduced by Conrad and Figliozzi (2011) and has been of interest to organisations, companies and researchers because of the new policies and regulations related to greenhouse gas emission in the transport sector. In fact, many companies started using EVs and their number is steadily increasing (Coplon-Newfield and Park, 2017). Many other companies such as FedEx, UPS, Frito-Lay, AT&T, General Electric, and Coca-Cola started testing or implementing this technology (Suizo, 2013). The enormous rise in interest of shifting from the conventional petroleum-fuel powered vehicles to EVs leads to this fertile area of research, the EVRP, in which a set of routes for a fleet of EVs is to be created. An EV is equipped with a limited-capacity battery that allows 160 to 240 kilometre to be driven before visiting a charging station in between customer visits, thereby allowing the continuation of its route (Van Duin et al., 2013, Young et al., 2013).

Following the introduction of the EVRP, various studies have considered variants of the basic problem. Schneider et al. (2014) introduce the EVRP with time windows and charging stations. Some studies assume that the stations have different chargers (Felipe et al., 2014, Keskin and Çatay, 2018, Li-ying and Yuan-bin, 2015, Sassi et al., 2014). Other papers deal with both location of the charging stations and the routing (Li-ying and Yuan-bin, 2015, Paz et al., 2018, Schiffer and Walther, 2017). A heterogeneous EV fleet is conidered by Lin et al. (2016). Other extensions of the basic EVRP including battery swap stations (BSS) where the low-charge battery is replaced with a fully recharged one (Jie et al., 2019, Liao et al., 2016, Paz et al., 2018), wireless charging systems (WCS) where the battery is recharged while driving (Li et al., 2018), and the EVRP with time windows (EVRPTW) involving time-dependent queuing times at recharging stations (Keskin et al., 2019) are recently introduced in the literature. An overview of solution approaches for solving the EVRP and its variants is presented by Erdelić and Carić (2019). The most recent review paper of Kucukoglu et al. (2021) covers EVRP variants, mathematical formulations, and solution approaches.

2.2.11 The multi-trip VRP

Contrary to the majority of the vehicle routing problems, a vehicle can perform more than a single trip in the multi-trip VRP (MTVRP). The multiple use of vehicles is more realistic in several practical situations. For example, distributing goods in city centres is usually performed by small vehicles and, because of the capacity limitation, they daily perform several short trips. Fleischmann (1990) addresses the problem under the name *Vehicle Routing Problem with Multiple Use of Vehicles*. A significant increase of the number of publications dealing with this subject is noticeable. The development of new distribution schemes in cities is the reason behind this gain of interest (Cattaruzza et al., 2018). The MTVRP appears in the literature under several names. In addition to the already mentioned *VRP with multiple use of vehicles* used by Fleischmann (1990), it has been addressed as *multitrip VRP* (Prins, 2002), *VRP with multiple routes* (Azi et al., 2007), *VRP with multiple trips* (Petch and Salhi, 2003), *VRP with multiple depot returns* (Tsirimpas et al., 2008), and *multiple trip VRP* (Battarra et al., 2009). Taniguchi and Van Der Heijden (2000) allow vehicles to make multiple *traverses*, while the multiple usage of vehicles has been called *recycling of trucks* in Van Buer et al. (1999).

Fleischmann (1990) is the first to address the MTVRP in his working paper in 1990, where he proposes a modification of the Clarke and Wright savings algorithm and the use of a bin packing (BP) heuristic to assign trips to the vehicles. Six years later, Taillard et al. (1996) propose a three-phase algorithm. In the first phase, a large number of good vehicle trips satisfying the VRP constraints are generated. Then, a subset of these trips is selected in the second phase and a MTVRP solution is constructed using a BP heuristic in the third phase. Petch and Salhi (2003) propose a multi-phase algorithm. In the first phase, VRP solutions are generated by the parametrised Yellow's savings algorithm (Yellow, 1970). For each VRP solution, a MTVRP solution is constructed using the same BP heuristic used by Taillard et al. (1996). Then, the MTVRP solutions are improved using 2-opt and 3-opt moves. Later, Salhi and Petch (2007) propose a genetic algorithm in which the plane is divided in circular sectors. Each sector is defined by an angle measured with respect to the depot and the x-axis. Customers are then clustered according to the sector they occupy. In each cluster, the Clarke and Wright savings heuristic is used to solve a smaller VRP problem. Then, a MTVRP solution is generated by packing the resulting trips using a BP heuristic.

Olivera and Viera (2007) used an adaptive memory approach to solve the MTVRP. A memory of trips is initialised by different VRP solutions generated by the sweep algorithm. Then, the algorithm iteratively creates new

VRP solutions by probabilistically selecting trips from the memory. These solutions are improved by applying a tabu search (TS) algorithm and then used to update the memory. At every iteration of the TS, a BP heuristic is used for the aim of producing a tentatively feasible MTVRP solution. The first exact method to solve the MTVRP is designed by Koc and Karaoglan (2011). They propose a branch-and-cut algorithm with several valid inequalities taken from the VRP literature that remain valid for the MTVRP. Mingozzi et al. (2013) propose more sophisticated exact method for the MTVRP based on branch-and-price.

Cattaruzza et al. (2014) propose a memetic algorithm in which each chromosome represents a customer sequence. They first apply a modified version of the split procedure proposed by Prins (2004) for the VRP. The splitting procedure is used to turn chromosomes into MTVRP solutions. They then compute the best MTVRP solution that can be obtained with the trips of this solution. The authors introduce a new local search operator based on a combination of standard VRP moves and swaps between trips. François et al. (2016) propose two adaptive large neighborhood search (ALNS) heuristics for the MTVRP, namely the ALNS with multi-trip operators (ALNSM), and the ALNS combined with BP (ALNSP). Both heuristics work on a relaxed version of the MTVRP where the tour duration constraints are relaxed and overtime is penalised in the objective function.

Several extensions of the MTVRP introduced in the literature. The MTVRP with time widows (MTVRPTW) is addressed in which each customer has an associated time interval during which service should occur. Several exact methods are proposed to solve the MTVRPTW. Azi et al. (2007) propose an exact algorithm that is able to solve to optimality instances with 100 customers and 1 vehicle. Furthermore, instances with 50 customers and 4 vehicles are solved exactly in Hernandez et al. (2014). Other extensions such as *service-dependent loading times*, where there is loading time for a vehicle at the depot that depends on the customers visited during the trip, *limited trip duration*, where there is a time limit on each trip's duration, and *profits* where serving all customers is not mandatory and a profit P_i is associated with serving customer i. Cattaruzza et al. (2018) provide a state-of-the-art survey on the MTVRP and its variants.

2.2.12 The two-echelon VRP

In the classical VRP, vehicles start and end their routes at the depot to serve a set of customers. However, in practice, there are some deliveries need to be undertaken to customers residing in inaccessible areas, e.g., pedestrian zones. Therefore, it is economically beneficial to divide the distribution network into two levels. In the first level, different vehicles, so-called *urban vehicles*, will be delivering parcels from the depot to intermediate facilities called *satellites*. In the second level, vehicles, porters, or cyclists (also known as *city freighters*) will be delivering parcels from satellites to a set of customers (Crainic et al., 2009). This problem is also known as the two-echelon distribution problem (2E-VRP) in the literature. The aim of this problem is to deliver parcels, consolidated through the satellites, to customers while the overall transportation cost is minimised.

The 2E-VRP is an extension to the classical VRP. The first formal definition of the problem is presented by Crainic et al. (2009). In their paper, a rich variant of a 2E-VRP with multiple products and depots, time-dependencies, and vehicle synchronisation is studied. The simplest and most frequently studied problem in the class of the 2E-VRPs, the two-echelon capacitated vehicle routing problem (2E-CVRP), is explicitly examined by a flow-based mathematical model by Perboli et al. (2011). Since then, exact (Baldacci et al., 2013, Santos et al., 2015), and heuristic (Crainic et al., 2011, Hemmelmayr et al., 2012) algorithms proposed in the literature. The 2E-VRP together with two other related problems, namely the two-echelon location-routing problem and the truck-and-trailer routing problem, are reviewed by Cuda et al. (2015). The most recent literature review on 2E-VRPs is by Sluijk et al. (2022). The authors discussed the canonical problem and its real-world inspired variants such as the 2E-VRP with time windows, pick-up and delivery operations, and multiple commodities. The state-of-the-art exact algorithm is the branch-cut-and-price algorithm proposed by Marques et al. (2020). Their algorithm is capable of solving instances with up to 300 customers and 15 satellites.

2.3 Basic models for the VRP

There are three main mathematical programming formulations to model VRPs; vehicle flow formulations, commodity flow formulations and set partitioning formulations.

- **Vehicle flow formulations:**

  Models of this type uses integer variables associated with each arc or edge that count the number of times that the arc or the edge is traversed by a vehicle. This is suitable for cases where the solution cost can be expressed as the sum of the costs associated with the arcs. These models are the most widely used for basic VRPs in the literature, although they cannot be used to formulate many practical variants of the VRP such as when the cost of the solution depends on the type of vehicle assigned to a route. A vehicle flow formulation, for example, is introduced by Li et al. (2019) for a 2E-VRP variant called the two-echelon time-constrained VRP. Another example of the use of this type of formulation is the vehicle flow formulation proposed by Li et al. (2017) for the roll-on roll-off VRP. According to Letchford and Salazar-González (2015), most of the successful exact algorithms for the CVRP are based on this type of formulation (Lysgaard et al., 2004).

- **Commodity flow formulations:**

  In this type of model, a new set of continuous variables are associated with the arcs or edges which represent the flow of the commodities along the paths travelled by the vehicles. Models of this type have only recently been used to find exact solutions of VRPs. A two-commodity flow formulation is proposed by Ramos et al. (2020) for the MDVRP. Letchford and Salazar-González (2015) present two multi-commodity flow formulations (MSF) that dominate (their continuous relaxations yield stronger lower bounds) other MCFs for the CVRP.

- **Set partitioning formulations:**

  For this approach, a set of feasible routes, each starting and ending at the depot, is created. The model associates a binary variable with each of there routes to indicate whether a route is used in the solution. The VRP is then formulated as a set partitioning problem having a solution comprising those routes that satisfy the VRP constraints of circuits with minimum cost. This allows for extremely general route costs, such as the travel cost being vehicle-dependent (Toth and Vigo, 2002b). Many successful exact algorithms for the CVRP are based on a set partitioning formulation (Baldacci et al., 2008b, Fukasawa et al., 2006).

Magnanti (1981) outlines several relationships between these three formulations. Additional formulations for the VRP are provided by Laporte and Nobert (1987). In most of the VRP variants, there are detailed review papers about the methods used to formulate the problem. For example, Kallehauge (2008) present a review paper for the VRPTW. Another example is the review paper by Oyola et al. (2018) for the formulations of SVRPs.

2.4 Exact algorithms for VRPs

In this section, we describe some algorithms that can be used to produce a solution that is guaranteed to be optimal (or can show that there is no feasible solution).

2.4.1 Branch-and-bound algorithms for VRPs

The most effective exact approaches until the late eighties were mainly branch-and-bound algorithms based on elementary combinatorial relaxations such as the assignment problem (AP), the degree-constrained shortest spanning tree (SST), and state-space relaxation. Laporte and Nobert (1987) provide a complete and detailed analysis of these algorithms. By the end of nineties, more sophisticated bounds, such as those based on Lagrangian relaxation or on the additive approach, have been proposed thereby increasing the size of the problems that can be solved to optimality (Toth and Vigo, 2002b).

The general idea behind branch-and-bound algorithms is to recursively decompose a problem into subproblems. To solve integer linear programs (ILPs), for maximisation problems, the method first solves the linear relaxation of the original ILP, using linear programming (LP) solution methods such as the simplex method. If an integer solution is obtained, then the problem is solved. Alternatively, if the solution is non-integer, then we have an upper bound on the objective value of an optimal solution (the lower bound is set to $-\infty$). Then, two new subproblems are created by adding additional constraints to the original problem. This process is known as *branching*. The linear relaxations of the two subproblems are then solved with the two solutions providing upper bounds for the two branches. This process is usually represented in the form of a search tree, with each node corresponding to a different subproblem. The following checks are made for each subproblem:

1. all variables in the solution for the relaxed subproblem are integral. If the objective value is greater than the existing lower bound, it replaces the existing lower bound;

2. the relaxed subproblem is infeasible;

3. the objective value of the fractional solution is below the current lower bound.

When one of these is satisfied, the search tree can be pruned by removing the node corresponding the to subproblem, which is often referred to as fathomed or killed. Toth and Vigo (2002a) present several basic combinatorial relaxations, including better relaxations based on Lagrangian and additive approaches which considerably increase the size of the instances solvable by branch-and-bound. Their book contains the main features (or ingredients) of the algorithm used for the exact solution of asymmetric and symmetric CVRP. A review of the main ingredients of branch-and-bound algorithms for the VRPs proposed by Semet et al. (2014).

2.4.2 Branch-and-cut algorithms for VRPs

Applying a branch-and-bound method that enables cutting planes to be added at any node of the tree is called branch-and-cut. This method has been very successful in solving many combinatorial optimisation problems (Caprara and Fischetti, 1997), but it may perform very poorly for some instances, such as when the number of iterations of the cutting plane phase is too high or the LP becomes unsolvable because of its size (Toth and Vigo, 2002b). For solving ILPs, the algorithm starts by solving the LP relaxation and if the optimal solution is integral, we stop; otherwise a cutting plane algorithm is used to find valid inequalities, which are often called cutting planes or cuts. These cuts are then added to the LP, which is then re-solved so that a better solution. From there, the branch-and-bound algorithm proceeds. The use of branch-and-cut for the VRP is rooted in the exact algorithm of Laporte et al. (1985), who introduce the two-index formulation of the VRP and describe the first branch-and-cut algorithm for its solution. Semet et al. (2014) present the main research works on branch-and-cut algorithms for the symmetric CVRP published between 1980 and 2005. Branch-and-cut algorithms are commonly designed to tackle VRP and its variants. For instance, a branch-and-cut algorithm is designed for the VRP with drones (Tamke and Buscher, 2021), two-dimensional loading constraints (Zhang et al., 2022), and with split delivery and time windows (Bianchessi and Irnich, 2019).

2.4.3 Branch-and-price algorithms for VRPs

A combination of branch-and-bound and column generation methods is used to create a branch-and-price algorithm in which columns might be added to the LP relaxation at each node of the search tree. For problems with many variables, most columns (variables) will be non-basic and their corresponding values equal to zero, thus, making them irrelevant for solving the problem. Considering a small number of columns is beneficial in reducing computational and memory requirements. The algorithm works as follows:

1. reformulate the problem using any technique such as Dantzig-Wolfe

decomposition to create the *master problem* with the aim of obtaining better bounds;

2. after the relaxation is solved, a large number of variables remains and the problem should be formulated as a *restricted master problem* (RMP) which has as a small subset of the columns as possible;

3. solve the LP relaxation of the RMP;

4. solve a sub-problem called the *pricing problem* to find columns with negative reduced cost;

5. if such a column is found, it is added to the RMP and the relaxation is re-optimised. On the other hand, when there is no columns can enter the basis and the solution to the relaxation is not integer, then branching occurs.

The philosophy of branch-and-price is similar to that of branch-and-cut except that the procedure focuses on column generation rather than row generation. Although both techniques have been extensively used with great success in the last few decades, the current best algorithms often belong to the branch-and-cut-and-price family, which is a combination of these methods (Toth and Vigo, 2014). Branch-and-price algorithms are commonly used to solve the VRP and its variants. For example, a branch-and-price algorithm for the VRP with time windows on a road network is designed by Ben Ticha et al. (2019). The two-echelon electric VRP was also tackled by Wu and Zhang (2021) using a branch-and-price algorithm.

2.5 Heuristics for VRPs

Exact algorithms are able to find optimal solutions for relatively small-size instances involving about 100 customer, but they are often extremely time consuming when solving real-world problems where instances are much larger and the computation time is limited. Heuristic techniques are powerful and flexible search methodologies have successfully tackled difficult practical problems. Heuristic algorithms seek to obtain high-quality solutions, but optimality cannot be guaranteed, in reasonable computation times and good enough for practical purposes. While efficient heuristics are required in practice, an enormous number of heuristics have been proposed for VRPs. Heuristics to solve VRPs can be classified as classical heuristics and metaheuristics.

2.5.1 Classical heuristics

Laporte and Semet (2002) classify classical heuristics for the VRP into three categories: constructive heuristics, two-phase methods, and improvement heuristics.

2.5.1.1 Constructive heuristics

The process of building an initial solution from scratch is called a constructive heuristic. There are two fundamental techniques used for constructing VRP solutions: merging existing routes using a saving criterion, and sequentially assigning customers to vehicle routes using an insertion cost to select the next customer together with a route and position for the insertion.

The first and the most widely known heuristic, based on the concept of saving, is the Clarke and Wright savings algorithm (Clarke and Wright, 1964). Because of the simplicity, intuitive appeal and the quality of solutions obtained with the algorithm, it has been widely accepted in the research community. The algorithm naturally applies to problems for which the number of vehicles is a decision variable, and it works equally well for problems defined on directed and undirected graphs. The algorithm starts from a solution in which each customer is visited in a separate tour. For each pair of customers, the saving by connecting these customers directly through merging the two routes is determined whenever this is feasible. The algorithm then creates a saving list by sorting these savings in a non-increasing order.

There are two versions of the Clarke and Wright algorithm, a sequential version where each route is built at a time, and a parallel version where routes are simultaneously built. Various enhancement strategies for the savings approach are proposed in the literature with the aim of improving either its effectiveness or its computational efficiency using better data structures (Golden et al., 1977, Paessens, 1988). Other attempts to improve the effectiveness of the saving method are made by Altinkemer and Gavish (1991), and Wark and Holt (1994).

The second type of constrictive heuristic is based on the sequential insertion of customers. Two sequential insertion algorithms are the Mole and Jameson (1976) sequential insertion heuristic that expands one route at a time, and the Christofides et al. (1979) sequential insertion heuristic that applies, in turn, sequential and parallel route construction procedures. Both heuristics have a 3-opt improvement phase. A detailed description and comparison between the two methods is reported by Toth and Vigo (2002b). They find that the sequential insertion heuristic of Christofides et al. (1979) is more general and effective than the Mole and Jameson (1976) algorithm.

2.5.1.2 Two-Phase methods

In two-phase methods, the VRP solution process is decomposed into two separate subproblems:

1. **clustering:** determine a partition of the customers into groups, each corresponding to a feasible route; and

2. **routing:** the customers in each of these groups are routed.

In cluster-first, route-second methods, customers are first grouped into clusters and then a vehicle route for each cluster is determined. As an example of the cluster-first, route-second approach is the *sweep* algorithm. The first mentions of this algorithm are found in a book by Wren and Carr (1971) and in a paper by Wren and Holliday (1972), but it became more popular as a result of the paper by Gillett and Miller (1974). An extension of the sweep algorithms is the class of so called *petal* algorithms. These generate several routes, called petals (Ryan et al., 1993), and make a final selection by solving a set partitioning model. Another example of this approach is the well-known *Fisher and Jaikumar* algorithm, which solves a generalised assignment problem (GAP) either optimally or heuristically to find clusters of customers, and then determines the final routes by solving a traveling salesman problem (TSP) on each cluster.

In route-first, cluster-second methods, a giant TSP tour is constructed over all customer in a first phase, which is then partitioned into feasible routes in a second phase. This idea applies to problems with an unlimited number of vehicles. Examples of such methods are provided in the literature (Beasley, 1983, Bertsimas and Simchi-Levi, 1996, Haimovich and Rinnooy Kan, 1985), but this approach is generally not competitive with other approaches (Cordeau et al., 2007).

2.5.1.3 Improvement heuristics

Improvement heuristics start with a given solution which is either generated randomly or by constructive heuristics. Local search is one of the improvement methods which tries to improve the solution through simple modifications such as arc exchanges or customer movements to obtain neighbor solutions possibly having, for a minimisation problem, a lower cost. If an improvement occurrs, the solution is updated and the process iterates; otherwise a local minimum has been found. Improvement heuristics can be subdivided into *single route* improvements, if they operate on a single route at a time, and *multiple route* improvements if they consider several routes simultaneously. The most common method of the former type is the k-opt heuristic of Lin (1965) for the TSP, where k edges are removed and replaced by k different edges. In practice, k takes the value 2 or 3. Commonly used methods for the latter type are multiple route improvements, including classical operators such as removing k consecutive customers from their current route and reinserting them elsewhere, so-called *relocate*, swapping customers between different routes, so-called *swap*, or removing two edges from different routes and reconnecting them differently, so-called *2-opt*. An example of an improvement heuristic designed for the period VRP (vehicle routes are planned over several days) is presented by Chao et al. (1995).

2.5.2 Metaheuristics

Unlike classical heuristics, a metaheuristic has the ability to avoid getting trapped at a local optima. This feature explains the wide use of metaheuristics. Braekers et al. (2016) indicate that for VRP publications within the period 2009 and 2015, more than 70% use metaheuristics as a solution method. Metaheuristics can be broadly classified into two classes:

1. **local search** methods explore the solution space by iteratively moving from a solution to another solution in its neighborhood until a stopping criterion is satisfied. These methods include simulated annealing (SA), tabu search (TS), and variable neighborhood search (VNS);

2. **population search** methods evolve a population of solutions which might be combined together in the hope of generating better ones. These including ant colony optimisation (ACO), genetic algorithms (GA), and adaptive memory procedures (AMP).

Combining ideas from different metaheuristic principles yields often to better heuristics, so called *hyper-heuristic*. Due to the large number of metaheuristics published for VRPs in recent years and their level of intricacy, we will concentrate on the basic principles of some local search algorithms since they are the most widely used in the literature (Laporte, 2009). An overview of metaheuristic principles can be found in the book by Gendreau et al. (2010).

2.5.2.1 Tabu search

One of the most effective and popular methods for solving VRPs is tabu search (TS), which is first proposed by Glover (1986). In TS, the solution space is explored by moving from the current solution to the best neighbor. In order to avoid cycling, solutions that were recently examined are forbidden, or tabu, for a number of iterations. To alleviate time and memory requirements, an attribute of tabu solutions is customary recorded rather than the solutions themselves. TS can sometimes successfully solve difficult problems to near optimality, although in most cases additional features such as intensification, and diversification have to be included in the search strategy to enhance its effectiveness. Various features are described by Burke et al. (2005). A large number of implementations of TS are proposed in the literature. A survey for the most important tabu search heuristics for the VRP is given by Cordeau and Laporte (2005). Zachariadis and Kiranoudis (2010) design a TS algorithm for the VRP that provides good results. A more recent TS approach for the VRP is described by Jia et al. (2013).

2.5.2.2 Variable neighborhood search

Variable neighborhood search (VNS) is introduced by Mladenović and Hansen (1997). It works with several *local search operators*, also called *neighborhoods*, which are usually nested. Starting with a given neighborhood, the algorithm iteratively applies these neighborhoods in a descent fashion until no further improvement is possible. After applying the last neighborhood, a new cycle can be started. The algorithm terminates after a predetermined number of cycles or when no further improvement can occur. Several variants of VNS are proposed in the VRP literature. A successful application, for example, is proposed by Kytöjoki et al. (2007). Variants of the VRP are also tackled using a VNS algorithm, such as by Polacek et al. (2004) for the multi-depot VRPTW, Sarasola et al. (2016) for the stochastic and dynamic VRP, Xu and Cai (2018) for the consistent VRP, and Yilmaz and Kalayci (2022) for the electric VRP with simultaneous pickup and delivery. The basic schemes, extensions, more recent developments, and some successful applications of VNS are presented by Hansen et al. (2010).

One of the most successful approaches for the VNS that has led to some of the most successful applications reported in the literature is the *general* VNS. In the general VNS, neighborhoods are used in a deterministic manner. Such procedures are known as variable neighborhood descent (VND). The VND method usually uses the steepest descent direction, or best improvement, heuristic in each of its neighborhoods and it stops when there is no direction of descent (Gendreau et al., 2010). When the order of neighborhoods is selected, the VND can be designed in two different ways: *sequential*, and *nested*. In the sequential VND, neighborhoods are always explored in the given order, whereas in the nested or composite VND, neighborhoods are composed. In the sequential VND, the basic, the pipe, the cyclic, and the union VND appear to be the most representative search methods. The main difference between these methods is the way of implementing the neighborhood change procedure. That is when an improvement is achieved by a neighborhood, the incumbent solution is updated and this is how the search is continued:

- **Basic VND** returns to the first neighborhood in the list;
- **Pipe VND** continues the search in the same neighborhood;
- **Cyclic VND** resumes the search in the next neighborhood of the list;
- **Union VND** continues the search in the same large neighborhood.

The reader can refer to Gendreau et al. (2010) for more details about VND variants. Duarte et al. (2018) discussed typical problems that arise in developing VND heuristic. The authors also performed a comparative analysis of common VND variants when solving TSP. In their analysis, they find that pipe VND is the fastest, but not the best, sequential VND version.

Chapter Three

A branch-and-cut algorithm for the TPRP

This chapter aims at designing and implementing a branch-and-cut algorithm for the truck-porters routing problem (TPRP). Section 3.1 provides a formal description of the problem. Two mathematical formulations are introduced in Section 3.2. Section 3.3 describes a set of families of valid inequalities for the TPRP. A general framework of the branch-and-cut algorithm for the TPRP is presented in Section 3.4, including the separation procedure of the capacity constraints, the set of families of valid inequalities, and the branching technique. Section 3.5 gives details about the method used for generating a set of problem instances, measuring the efficiency of each family of valid inequalities, and reports on the computational results. Conclusions of this chapter are given in Section 3.6.

3.1 Problem description

In the TPRP, there are n customers requiring deliveries from a depot. Each delivery is performed either by one of m porters, each with a capacity of Q_P units, or by a truck with no capacity limit. We define $M = \{1, 2, ..., m\}$ to be the set of porters. Customers may require a truck delivery, a porter delivery, or a delivery by either the truck or a porter; hence they are referred to as *truck* customers, *porter* customers and *unconstrained* customers, respectively. Each customer i requires a delivery of q_i units. The delivery network is represented by a complete directed graph $G = (V, A)$, where $V = V' \cup \{0\}$, V' is a set of vertices corresponding to customer locations, 0 is the vertex corresponding to the depot and A comprises a set of arcs (i, j) between vertex $i \in V$ and vertex $j \in V$ for $i \neq j$. We define V_T to be the set of vertices for truck customers, V_P to be the set of vertices for porter customers and V_U to be the vertices for unconstrained customers, so that $V' = V_T \cup V_P \cup V_U$ and $V = V_T \cup V_P \cup V_U \cup \{0\}$. We also define $V_{PU} = V_P \cup V_U$ and $V_{TU} = V_T \cup V_U$.

The objective of the TPRP is determine a delivery scheme having a minimum total cost. The cost comprises a fixed cost F_P for each porter that is used, a travel cost $\bar{c}_{ij}$ for each arc (i, j) traversed by the truck, and a travel cost c_{ij} for each arc (i, j) traversed by a porter. We assume that c_{ij} is also the travel time for traversing each arc (i, j). Matrices $(\bar{c}_{ij})$ and (c_{ij}) are assumed to satisfy the triangle inequality.

The delivery scheme specifies a route for the truck, the number of porters to be used together, and a route for each of these porters. Each customer must be visited exactly once by the truck or a porter, with the constraints for truck and porter customers satisfied. A truck route starts and ends at the depot, and visits a subset of customers from $V_T \cup V_U$. A porter route may include several trips, where each trip starts and ends at the depot, and visits a subset of customers from $V_P \cup V_U$. Feasibility of a porter trip visiting customers in set S is ensured if $\sum_{i \in S} q_i \leq Q_P$ and feasibility of a porter route traversing arcs in set S' is ensured if $\sum_{(i,j) \in S'} q_i \leq T_P$, where T_P is a time limit on each porter's delivery schedule.

3.2 Mathematical formulation

The formulation below uses the following variables:

$$\bar{x}_{ij} = \begin{cases} 1, & \text{the truck traverses arc } (i, j) \in A; \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ij}^k = \begin{cases} 1, & \text{the porter } k \text{ where } k \in M \text{ traverses arc } (i, j) \in A; \\ 0, & \text{otherwise.} \end{cases}$$

$$z^k = \begin{cases} 1, & \text{the porter } k \text{ where } k \in M \text{ is active;} \\ 0, & \text{otherwise.} \end{cases}$$

The resulting model is:

$$\min \quad F_P \sum_{k \in M} z^k + \sum_{k \in M} \sum_{i,j \in V_{PU} \cup \{0\}} c_{ij} x_{ij}^k + \sum_{i,j \in V_{TU} \cup \{0\}} \bar{c}_{ij} \bar{x}_{ij} \tag{1}$$

$$s.t. \quad \sum_{k \in M} \sum_{i \in V_{PU} \cup \{0\}} x_{ij}^k = 1, \quad \forall j \in V_P \tag{2}$$

$$\sum_{i \in V_{TU} \cup \{0\}} \bar{x}_{ij} = 1, \quad \forall j \in V_T \tag{3}$$

$$\sum_{k\in M}\sum_{i\in V_{PU}\cup\{0\}} x_{ij}^k + \sum_{i\in V_{TU}\cup\{0\}} \bar{x}_{ij} = 1, \quad \forall j \in V_U \tag{4}$$

$$\sum_{i\in V_{PU}\cup\{0\}} x_{is}^k - \sum_{j\in V_{PU}\cup\{0\}} x_{sj}^k = 0, \quad \forall s \in V_{PU}\cup\{0\}, \forall k \in M \tag{5}$$

$$\sum_{i\in V_{TU}\cup\{0\}} \bar{x}_{is} - \sum_{j\in V_{TU}\cup\{0\}} \bar{x}_{sj} = 0, \quad \forall s \in V_{TU}\cup\{0\} \tag{6}$$

$$\sum_{i,j\in V_{PU}\cup\{0\}} c_{ij}x_{ij}^k \le \sum_{i,j\in V_{PU}\cup\{0\}} c_{ij}x_{ij}^{k-1}, \quad \forall k \in M\backslash\{1\} \tag{7}$$

$$x_{ij}^k + x_{ji}^k \le z^k, \quad \forall i,j \in V_{PU},\ \forall k \in M \tag{8}$$

$$x_{0j}^k \le z^k, \quad \forall j \in V_{PU}, \forall k \in M \tag{9}$$

$$\sum_{i,j\in V_{PU}\cup\{0\}} c_{ij}x_{ij}^k \le T_P, \quad \forall k \in M \tag{10}$$

$$\sum_{k\in M}\sum_{i\in S}\sum_{j\in S} x_{ij}^k \le |S| - \left\lceil \frac{Q(S)}{Q_P} \right\rceil, \quad \forall\ S \subseteq V_{PU},\ S \neq \emptyset \tag{11}$$

$$\sum_{i\in S}\sum_{j\in S} \bar{x}_{ij} \le |S| - 1, \quad \forall\ S \subseteq V_{TU},\ S \neq \emptyset \tag{12}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall i,j \in V_{PU}\cup\{0\},\ i \neq j,\ \forall k \in M \tag{13}$$

$$\bar{x}_{ij} \in \{0,1\}, \quad \forall i,j \in V_{TU}\cup\{0\},\ i \neq j \tag{14}$$

$$z^k \in \{0,1\}, \quad \forall k \in M \tag{15}$$

The objective function (1) aims to minimise the total travel time by the truck and the porters and the number of porters used. Constraints (2)–(4) impose that every customer is visited exactly once. Constraints (5) and (6) ensures the connectivity of the routes. Constraints (7) are symmetry breaking constraints: they force the travel time of porter $k-1$ to be at least as much as the travel time for porter k. To include the cost of active porters in the objective function, we have constraints (8) and (9) that define the value of z^k. Constraints (10) ensure that porters do not exceed the predetermined time T_P. Constraints (11) and (12), which are called the capacity cut constraints (CCCs), ensure connectivity of the solution and avoid porter capacity violations. They generalise CCCs for the capacitated vehicle routing problem (CVRP). Constraints (13), (14) and (15) specify the binary nature of the decision variables x_{ij}^k, $\bar{x}_{ij}$ and z^k.

The number of CCCs given by (11) and (12) grows exponentially with n. Thus, in order to overcome this drawback, we rely on separation procedures.

Alternatively, we can replace (11) and (12) by the following:

$$\sum_{k\in M}\Big[\sum_{i\in V_{PU}\cup\{0\}}\Big(y_{ij}^k - y_{ji}^k\Big)\Big] = q_j, \quad \forall j\in V_P \tag{16}$$

$$\sum_{i\in V_{TU}\cup\{0\}}\Big(\bar{y}_{ij} - \bar{y}_{ji}\Big) = q_j, \quad \forall j\in V_T \tag{17}$$

$$\sum_{k\in M}\Big[\sum_{i\in V_{PU}\cup\{0\}}\Big(y_{ij}^k - y_{ji}^k\Big)\Big] + \sum_{i\in V_{TU}\cup\{0\}}\Big(\bar{y}_{ij} - \bar{y}_{ji}\Big) = q_j, \quad \forall j\in V_U \tag{18}$$

$$q_j x_{ij}^k \le y_{ij}^k \le (Q_P - q_i)x_{ij}^k, \quad \forall i,j\in V_{PU}\cup\{0\}, i\neq j, \forall k\in M \tag{19}$$

$$q_j \bar{x}_{ij} \le \bar{y}_{ij} \le (Q_T - q_i)\bar{x}_{ij}, \quad \forall i,j\in V_{TU}\cup\{0\}, i\neq j \tag{20}$$

$$y_{ij}^k \ge 0, \quad \forall i,j\in V, i\neq j, \forall k\in M \tag{21}$$

$$\bar{y}_{ij} \ge 0, \quad \forall i,j\in V, i\neq j \tag{22}$$

where y_{ij}^k and $\bar{y}_{ij}$ are additional continuous variables representing the load after visiting customer i by the porters and by the truck respectively. These constraints govern the commodity flow conservation and capacity restrictions. The advantage of using flow conservation constraints, (16)–(20), is that the model has a polynomial number of constraints in terms of the number of customers. However, the lower bound provided by the linear programming (LP) relaxation of this model is known to be weak in relation to other models (Toth and Vigo, 2002b).

3.3 Valid inequalities

In this section, we introduce several families of valid inequalities for the TPRP. These inequalities can be added to the two formulations introduced in the previous section in order to strengthen their LP relaxations. The impact of each family of valid inequalities is assessed through computational experiments in the last section of this chapter.

The presence of unconstrained customers, V_U, in the TPRP leads us to the majority of families of valid inequalities introduced in this section. For this reason, we shed light on this type of customers at this stage. We know that any customer i, where $i\in V_U$, must be visited either by the truck or by a porter as given by constraints (4), so it is obvious that:

$$\sum_{k\in M}\sum_{i\in V_U}\sum_{j\in V_{PU}\cup\{0\}} x_{ij}^k + \sum_{i\in V_U}\sum_{j\in V_{TU}\cup\{0\}} \bar{x}_{ij} \le |V_U|$$

which means that the total number of outgoing arcs from V_U nodes toward

V, either by the truck or by any porter, is always less than or equal to the number of V_U nodes, $|V_U|$. Therefore, for any subset S such that:

(a) $S \subseteq V_{TU}$, then we can state that

$$\sum_{k\in M}\sum_{i\in V_U}\sum_{j\in V_{PU}\cup\{0\}} x_{ij}^k + \sum_{i\in V_U}\sum_{j\in S}\bar{x}_{ij} \le |V_U|. \tag{23}$$

(b) $S \subseteq V_{PU}$, then we can state that

$$\sum_{k\in M}\sum_{i\in V_U}\sum_{j\in S} x_{ij}^k + \sum_{i\in V_U}\sum_{j\in V_{TU}\cup\{0\}}\bar{x}_{ij} \le |V_U|. \tag{24}$$

We are going to use (23) and (24) to prove the validity of some families of valid inequalities. It is worth mentioning at this stage the way by which the CCCs of (11) are found. We know from the well-known capacity cut constraints CCCs of the CVRP that $\sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - r(S)$, where $r(S)$ is the minimum number of vehicles needed to serve set S. We also know that the CCCs remain valid when we replace $r(S)$ by the trivial *Bin Packing Problem* lower bound $\left\lceil \frac{Q(S)}{C} \right\rceil$, where $Q(S) = \sum_{i\in S} q_i$ and C is the vehicle capacity (Cornuejols and Harche, 1993). Therefore, the CCCs for the porters in the TPRP are given by

$$\sum_{k\in M}\sum_{i\in S}\sum_{j\in S} x_{ij}^k \le |S| - \left\lceil \frac{Q(S)}{Q_P} \right\rceil, \quad \forall\ S \subseteq V_{PU}$$

where $Q_P = C$, which is constraints (11) in the original formulation. Constraints (12) are the well-know subtour elimination constraints (SECs) of the TSP for the truck since the truck has no capacity restriction. The latter constraints, (12), can be strengthen in two different cases. The next two propositions show these cases.

Proposition 1. *For any set $S \subseteq V_{TU}$, if S contains at least one node i such that $i \in V_T$, the subtour elimination inequalities (12) can be strengthened as:*

$$\sum_{i\in S}\sum_{j\in S}\bar{x}_{ij} + \sum_{k\in M}\sum_{m\in S\cap V_U}\sum_{z\in V_{PU}\cup\{0\}} x_{mz}^k \le |S| - 1,$$
$$\forall\ S \subseteq V_{TU},\ S\cap V_T \ne \emptyset. \tag{25}$$

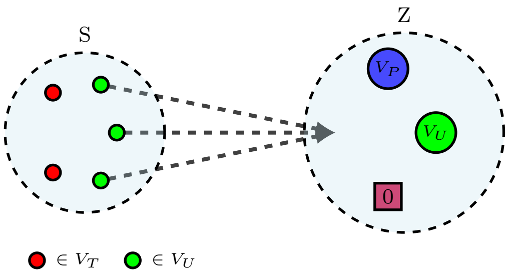

Figure 3.1: An illustrated example of Proposition 1.

Proof. Let N be a subset of S, where $S \subseteq V_{TU}$, such that $N = S \cap V_U$. If $N = \emptyset$, it means that $S \subseteq V_T$ and the summation over m, where $m \in S \cap V_U$, in (25) is equal to 0, which is the original SECs of (12). On the other hand, if $N \neq \emptyset$, then any node in N is visited exactly once, either by the truck or by a porter. By the fact that

$$\sum_{i \in S} \sum_{j \in S} \bar{x}_{ij} = \sum_{i \in S \setminus N} \sum_{j \in S \setminus N} \bar{x}_{ij} + \sum_{i \in N} \sum_{j \in S} \bar{x}_{ij}$$

we can re-write (25) as follows:

$$\sum_{i \in S \setminus N} \sum_{j \in S \setminus N} \bar{x}_{ij} + \sum_{m \in S \cap V_U} \sum_{j \in S} \bar{x}_{mj} + \sum_{k \in M} \sum_{m \in S \cap V_U} \sum_{z \in V_{PU} \cup \{0\}} x^k_{mz} \leq |S| - 1. \quad (26)$$

Therefore, we can replace the first term of (26) by its maximum value which is equal to $(|S| - |N|) - 1$ using the SEC of (12). Also, we know that the sum of the second and the third terms of (26) is less than or equal to $|N|$ as given in (23). As a result, the LHS of (25) is always less than or equal to $|S| - 1$, which is the RHS of (25). Thus, the inequalities (25) are valid. □

Proposition 2. *For any set $S \subseteq V_U$, inequalities (12) can be lifted to yield:*

$$\sum_{i \in S} \sum_{j \in S} \bar{x}_{ij} + \sum_{k \in M} \sum_{z \in V_{PU} \cup \{0\}} x^k_{mz} \leq |S| - 1, \quad \forall\ S \subseteq V_U,\ m \in S. \quad (27)$$

Proof. Consider any feasible solution of the TPRP and any customer m, where $m \in V_U$. Suppose first that customer m is visited by a porter. This implies that the number of outgoing arcs for porter routes from m to the depot and to a node z, where $z \in V_{PU}$, is equal to one because constrains (5) ensure the connectivity of the routes for the porters. In this case, the truck cannot travel from the depot nor from a vertex i, where $i \in V_{TU}$, to m and,

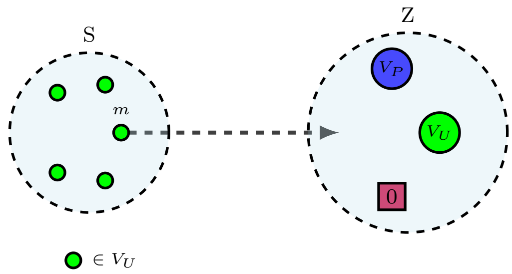

Figure 3.2: An illustrated example of Proposition 2.

thus, the first term of (27) is equivalent to $\sum_{i\in S\setminus m}\sum_{j\in S\setminus m}\bar{x}_{ij}$. From the SEC of (12), this term is always less than or equal to $(|S|-|m|)-1$ which is $|S|-2$, since $|m|=1$. Therefore, the LHS of (27) is less than or equal to $(|S|-2)+1$, which is the RHS of (27).

Alternatively, suppose that customer m is not visited by any porter. This means that customer m must be visited by the truck and the sum over z in (27) is equal to zero. In this case, all nodes in S are visited by the truck and we can simply use the SEC of (12) to obtain the upper bound of the LHS of (27), by ignoring the second term of (27), which is $|S|-1$. Hence, the inequalities (27) are valid. □

Proposition (25) and (27) are not comparable. The former proposition is only applicable when the set of customers S contains at least one truck-customer, that is $S\cap V_T\neq\emptyset$. Whereas, the latter proposition is applicable when the set of customers S consists of V_U customers, that is $S\subseteq V_U$.

Proposition 3. *For any set $S\subseteq V_{PU}$, when S contains at least one node i such that $i\in V_P$, the following inequality is valid for the TPRP:*

$$\sum_{k\in M}\sum_{i\in S}\sum_{j\in S}x_{ij}^k+\sum_{m\in S\cap V_U}\sum_{z\in V_{TU}\cup\{0\}}\bar{x}_{mz}\leq |S|-\left\lceil\frac{Q(S\cap V_P)}{Q_P}\right\rceil, \quad \forall\ S\subseteq V_{PU},\ S\cap V_P\neq\emptyset. \tag{28}$$

Proof. Suppose N is a subset of S, where $S\subseteq V_{PU}$, such that $N=S\cap V_U$. If $N=\emptyset$, it follows that $S\subseteq V_P$, therefore $Q(S\cap V_P)=Q(S)$ and the summation over m, where $m\in S\cap V_U$, in (28) is equal to 0, which is the original CCCs of (11). However, if $N\neq\emptyset$, then any node in N is visited exactly once, either by the truck or by any porter. Because

$$\sum_{k\in M}\sum_{i\in S}\sum_{j\in S}x_{ij}^k=\sum_{k\in M}\sum_{i\in S\setminus N}\sum_{j\in S\setminus N}x_{ij}^k+\sum_{k\in M}\sum_{i\in N}\sum_{j\in S}x_{ij}^k$$

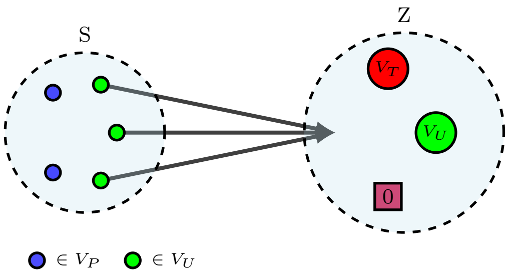

Figure 3.3: An illustrated example of Proposition 3.

we can re-write (28) as

$$\sum_{k\in M}\sum_{i\in S\setminus N}\sum_{j\in S\setminus N} x_{ij}^k + \sum_{k\in M}\sum_{i\in N}\sum_{j\in S} x_{ij}^k + \sum_{m\in S\cap V_U}\sum_{z\in V_{TU}\cup\{0\}} \bar{x}_{mz}$$
$$\leq |S| - \left\lceil \frac{Q(S\cap V_P)}{Q_P} \right\rceil, \qquad \forall\ S\subseteq V_{PU},\ S\cap V_P \neq \emptyset. \tag{29}$$

At any feasible solution of the TPRP, the sum of the second and the third terms of (29) is always less than or equal to $|N|$, according to (24). We also know from (11) that

$$\sum_{k\in M}\sum_{i\in S\setminus N}\sum_{j\in S\setminus N} x_{ij}^k \leq (|S| - |N|) - \left\lceil \frac{Q(S\setminus N)}{Q_P} \right\rceil.$$

Thus, the LHS of (29) is always less than or equal to

$$(|S| - |N|) - \left\lceil \frac{Q(S\setminus N)}{Q_P} \right\rceil + |N|$$

which is the RHS of (29). Hence, inequality (28) holds. □

Proposition 4. *For any set $S \subseteq V_U$, the following inequality is valid for the TPRP:*

$$\sum_{k\in M}\sum_{i\in S}\sum_{j\in S} x_{ij}^k + \sum_{z\in V_{TU}\cup\{0\}} \bar{x}_{mz} \leq |S| - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil,$$
$$\forall\ S\subseteq V_U,\ m\in S. \tag{30}$$

Proof. Consider any feasible solution of the TPRP with any customer m, where $m \in V_U$. Suppose first that customer m is visited by the truck. This implies that the sum of outgoing arcs traversed by the truck from m to the

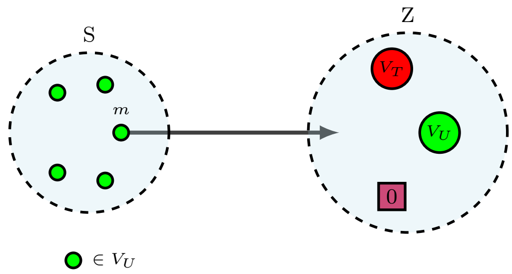

Figure 3.4: An illustrated example of Proposition 4.

depot and from m to a node z, where $z \in V_{TU}$, is equal to one. This is because constrains (6) ensure the connectivity of the route for the truck. In this case, porters cannot travel to m from the depot or from a node i, where $i \in V_{PU}$, thus implying that the first term of (30) is equivalent to $\sum_{k \in M} \sum_{i \in S \setminus m} \sum_{j \in S \setminus m} x_{ij}^k$. From the CCCs of (11), this term is always less than or equal to

$$(|S| - |m|) - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil \tag{31}$$

where $|m| = 1$ and q_m is the demand of customer m. Therefore, the maximum value for the LHS of (30) is equal to (31) + 1, which is the RHS of (30).

Alternatively, suppose that customer m is not visited by the truck. This implies that customer m must be visited by any porter. Using the fact that

$$\sum_{k \in M} \sum_{i \in S} \sum_{j \in S} x_{ij}^k = \sum_{k \in M} \sum_{i \in S \setminus \{m\}} \sum_{j \in S \setminus \{m\}} x_{ij}^k + \sum_{k \in M} \sum_{j \in S} x_{mj}^k$$

we can re-write (30) as

$$\sum_{k \in M} \sum_{i \in S \setminus \{m\}} \sum_{j \in S \setminus \{m\}} x_{ij}^k + \sum_{k \in M} \sum_{j \in S} x_{mj}^k + \sum_{z \in V_{TU} \cup \{0\}} \bar{x}_{mz}$$
$$\leq |S| - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil, \quad \forall \; S \subseteq V_U, \; m \in S. \tag{32}$$

The fist term of (32) is less than or equal to (31), the second term is equal to one, and the third term, the summation over z, is equal to zero. Thus, the LHS of (30) is less than or equal to $|S| - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil$, which is the RHS of (30). Hence, the inequalities (30) are valid. □

Proposition 5. *For any porter k' and set $S \subseteq V_{PU}$, if S contains at least one node i such that $i \in V_P$, the following inequality is valid for the TPRP:*

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k'} + \sum_{m\in S\cap V_U}\ \sum_{f\in V_{PU}\cup\{0\}}\ \sum_{k\in M\setminus\{k'\}} x_{mf}^{k} + \sum_{m\in S\cap V_U}\ \sum_{z\in V_{TU}\cup\{0\}} \bar{x}_{mz}$$
$$\leq |S| - \left\lceil \frac{Q(S\cap V_P)}{Q_P} \right\rceil, \quad \forall k' \in M,\ S \subseteq V_{PU},\ S\cap V_P \neq \emptyset. \tag{33}$$

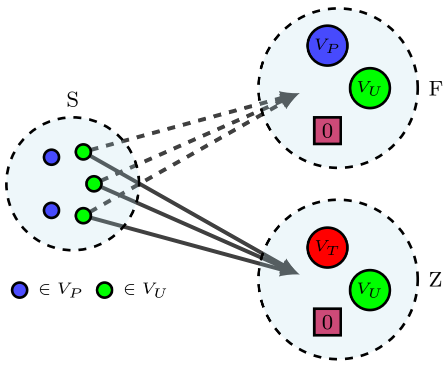

Figure 3.5: An illustrated example of Proposition 5.

Proof. Let N be a subset of S, where $S \subseteq V_{PU}$, such that $N = S \cap V_U$. If $N = \emptyset$, it follows that $S \subseteq V_P$, therefore implying $Q(S\cap V_P) = Q(S)$ and the sum of the second and the third terms in (33) is equal to 0. From the CCCs of (11), it is apparent that the first term of (33) is less than or equal to $|S| - \left\lceil \frac{Q(S)}{Q_P} \right\rceil$, which is the RHS of (33). However, if $N \neq \emptyset$, then any node in N is visited exactly once, either by the truck or by any porter. Using the equation

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k'} = \sum_{i\in S\setminus N}\ \sum_{j\in S\setminus N} x_{ij}^{k'} + \sum_{i\in N}\sum_{j\in S} x_{ij}^{k'}$$

and since

$$\sum_{i\in N}\sum_{j\in S} x_{ij}^{k'} + \sum_{m\in N}\ \sum_{f\in V_{PU}\cup\{0\}}\ \sum_{k\in M\setminus\{k'\}} x_{mf}^{k} + \sum_{m\in N}\ \sum_{z\in V_{TU}\cup\{0\}} \bar{x}_{mz} \leq |N|$$

according to (24), we can re-write (33) as

$$\sum_{i\in S\setminus N}\ \sum_{j\in S\setminus N} x_{ij}^{k'} + |N| \leq |S| - \left\lceil \frac{Q(S\cap V_P)}{Q_P} \right\rceil,$$
$$\forall k' \in M,\ S \subseteq V_{PU},\ S\cap V_P \neq \emptyset$$

Therefore, using (11), the LHS of (33) is always less than or equal to

$$(|S| - |N|) - \left\lceil \frac{Q(S \backslash N)}{Q_P} \right\rceil + |N|,$$

which is the RHS of (33). Hence, the inequalities given in (33) hold. □

Proposition 6. *For any porter k' and set $S \subseteq V_{PU}$, the following inequality is valid for the TPRP:*

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^{k'} + \sum_{f \in V_{PU} \cup \{0\}} \sum_{k \in M \backslash \{k'\}} x_{mf}^{k} + \sum_{z \in V_{TU} \cup \{0\}} \bar{x}_{mz} \leq |S| - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil, \quad \forall k' \in M,\ S \subseteq V_{PU},\ m \in S. \tag{34}$$

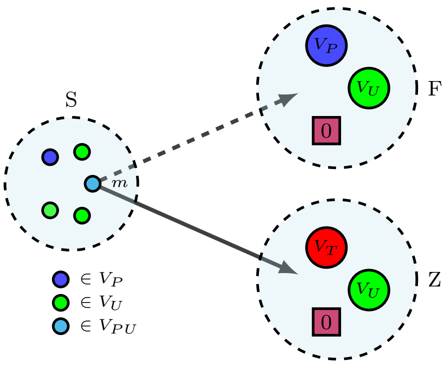

Figure 3.6: An illustrated example of Proposition 6.

Proof. Consider any feasible solution of the TPRP and any customer m, where $m \in V_U$. Suppose first that customer m is visited by the truck. This implies that the summation over z in (34) is equal to one because of constraints (6), that ensure connectivity of the route for the truck. In this case, porters cannot travel from m to the depot or to a node f, where $f \in V_{PU}$, which establishes that the second term of (34) is equal to zero. Finally, the first term is equivalent to $\sum_{i \in S \backslash \{m\}} \sum_{j \in S \backslash \{m\}} x_{ij}^{k'}$ which, by (11), is always less than or equal to

$$(|S| - |\{m\}|) - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil \tag{35}$$

where $|\{m\}| = 1$. Therefore, the maximum value of the LHS of (34) is equal to (35) + 1, which is the RHS of (34).

Alternatively, suppose that customer m is not visited by the truck. This implies that customer m must be visited by a porter and the summation over z in (34) is equal to zero. It is straightforward to establish that

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k'} = \sum_{i\in S\setminus\{m\}}\sum_{j\in S\setminus\{m\}} x_{ij}^{k'} + \sum_{j\in S} x_{mj}^{k'}$$

and we, also, know that the number of outgoing arcs from m to either the depot or to a node f, where $f \in V_{PU}$, by porter k' and all other porters is equal to one. This can be expressed as

$$\sum_{j\in S} x_{mj}^{k'} + \sum_{f\in V_{PU}\cup\{0\}}\sum_{k\in M\setminus\{k'\}} x_{mf}^{k} = 1.$$

Thus, we can re-write (34) as

$$\sum_{i\in S\setminus\{m\}}\sum_{j\in S\setminus\{m\}} x_{ij}^{k'} + 1 \leq |S| - \left\lceil \frac{Q(S) - q_m}{Q_P} \right\rceil, \qquad \forall k' \in M,\ S \subseteq V_{PU},\ m \in S.$$

Therefore, by (11), the LHS of (34) is always less than or equal to $|S| - \left\lceil \frac{Q(S)-q_m}{Q_P} \right\rceil$, which is the RHS of (34). Hence, the inequalities (34) are valid. $\square$

The separation procedures for the CCCs of (11) and (12) together with the six families of valid inequalities introduced in the six propositions of this section are described in the next section. The two inequalities

$$\sum_{k\in M}\sum_{j\in V_{PU}} x_{0j}^{k} \geq \left\lceil \frac{Q(V_P)}{Q_P} \right\rceil \tag{36}$$

and

$$\sum_{j\in V_{TU}} \bar{x}_{0j} \geq 1 \tag{37}$$

have been added to the formulations to strengthen their LP relaxations. Inequality (36) ensures that the minimum number of arcs leaving the depot to V_{PU} customers by porters is greater than or equal to $\left\lceil \frac{Q(V_P)}{Q_P} \right\rceil$. This holds because V_P customers must be served by porters, thus implying that porters must depart from the depot at least $\left\lceil \frac{Q(V_P)}{Q_P} \right\rceil$ times. Moreover, the problem definition for the TPRP specifies that there is at least one V_T customer, so the truck must depart from the depot exactly once, therefore establishing inequality (37).

3.4 The branch-and-cut algorithm

In this section, we describe a branch-and-cut algorithm for the exact solution of the TPRP. After solving the LP relaxation of the problem by relaxing capacity constraints (11)–(12) and integrality constraints (13)–(15), if the solution is integer but not feasible, a violated capacity constraint can be easily identified. However, if the solution is integer and feasible, then an optimal solution of the TPRP has been obtained. For the case of infeasibility, the LP can be strengthen by adding a set of valid inequalities, so-called cutting planes, that are violated by this optimal non-integer solution. This process is repeated until either an integer feasible solution is found (which is an optimal solution for the TPRP) or the separation routine fails to find a valid inequality violated by the current optimal LP solution. In the latter case, a lower bound on the cost of an optimal TPRP solution is obtained and in order to solve the TPRP we need to employ a branching scheme. Thus, a search tree is constructed, and violated valid inequalities are produced at some nodes of this tree. At each node of the tree, the LP is solved and the separation routine is called to find violated constraints. If a violated constraint is identified at a given node, it is added to its LP, and we proceed as before. Otherwise, branching takes place on a variable which has a fractional value. Subsection 3.4.1 describes the separation procedure used to identify violated inequalities. The branching strategy is explained in Subsection 3.4.2.

3.4.1 Separation strategy

Let x^* be any LP solution vector satisfying (2)–(10) and (13)–(15). For this solution, let $A^* = \{(i,j) \in A \ : \ \sum_{k\in M} x_{ij}^k + \bar{x}_{ij} > 0\}$, which produces a support graph $G^* = (V, A^*)$. Note that, arcs (i,j) and (j,i) such that $i \in V_T$ and $j \in V_P$ cannot form part of any feasible solution, so the corresponding variables x_{ij}^k and x_{ji}^k for $k \in M$, $\bar{x}_{ij}$ and $\bar{x}_{ji}$ are not defined. Similarly, variables x_{0j}^k and x_{j0}^k for $k \in M$ and $j \in V_T$, and variables $\bar{x}_{0j}$ and $\bar{x}_{j0}$ for $j \in V_P$ are also undefined.

Given a non-empty subset S, let $\delta^T(S)$ be the set of truck arcs that have one end in S and the other one in $V \backslash S$, that is $\delta^T(S) = \{(i,j) \in A : \ i \in S, \ j \in V \backslash S\}$, and let $x^*(\delta^T(S)) = \sum_{i\in S}\sum_{j\in V\backslash S} \bar{x}_{ij}$. Moreover, let $\delta^P(S)$ denote the set of porter arcs with one end in S and the other one in $V\backslash S$, that is $\delta^P(S) = \{(i,j) \in A : \ i \in S, \ j \in V\backslash S\}$, and let $x^*(\delta^P(S)) = \sum_{i\in S}\sum_{j\in V\backslash S}\sum_{k\in M} x_{ij}^k$. In addition, for any non-empty subset S, let $\gamma^T(S)$ and $\gamma^P(S)$ be the set of arcs with both ends in S for the truck and for the porters respectively. Therefore, $\gamma^T(S) = \{(i,j) \in A : \ i,j \in S\}$

and $\gamma^P(S) = \{(i,j) \in A : \; i,j \in S\}$. Also, let $x^*(\gamma^T(S)) = \sum_{i\in S}\sum_{j\in S}\bar{x}_{ij}$ and $x^*(\gamma^P(S)) = \sum_{i\in S}\sum_{j\in S}\sum_{k\in M} x_{ij}^k$.

Designing an effective procedure to find valid inequalities that are violated by a given LP solution plays an important role in the success of a branch-and-cut algorithm. In our algorithm, we design a simple tabu search procedure for this purpose. The main aim of our heuristic algorithm is to find promising sets S for which $x^*(\delta^T(S))$, $x^*(\delta^P(S))$, $x^*(\gamma^T(S))$, $x^*(\gamma^P(S))$, and the total demand $Q(S)$ in set S, are computed to check if any capacity cut constraint (11) or (12), or any valid inequality, Proposition (1)–(6), is violated. The reason of resorting to a heuristic approach to separate capacity inequalities, (11) and (12), is given in the following proposition.

Proposition 7. *The separation of the capacity cut constraints (CCCs) of the TPRP is an $\mathcal{NP}$-hard problem.*

Proof. The separation of the CCCs for the CVRP is known to be strongly $\mathcal{NP}$-hard (Augerat et al., 1995, Naddef and Rinaldi, 2002). Also, the CCCs of the TPRP generalise the CCCs of the CVRP. Therefore, the separation of the CCCs of the TPRP is an $\mathcal{NP}$-hard problem. □

Our procedure for finding valid violated inequalities starts by computing the connected components of the support graph G^*. For each connected component C, the heuristic starts with a random node $v \in C$, initialises set S by setting $S = \{v\}$. The following iterative process is applied for β iterations, where β is a parameter. At every iteration, a node is added to S, or a node is removed from S in order to maximise $x^*(\gamma^T(S))$ or $x^*(\gamma^P(S))$. The chosen movement is the best among all the possibilities although $x^*(\gamma^T(S))$ or $x^*(\gamma^P(S))$ may increase or decrease from an iteration to the next, which may cause cycling to occur. Thus, to prevent cycling, we use a tabu list $\mathcal{L}$, which has a maximum length of ℓ where ℓ is a parameter.

The algorithm has an **Expansion** phase, a **Removal** phase and an **Addition or Removal** phase. First, the expansion phase is executed with set S, starting with $S = \{v\}$, being enlarged by successively adding a node from C in order to maximise $x^*(\gamma^T(S))$ or $x^*(\gamma^P(S))$, until $|S| = |C|$. At this stage, $S = C$ and the tabu list is empty, i.e., $\mathcal{L} = \emptyset$. Then, the second phase is applied by removing nodes from S. When a node j removed from S, it is declared tabu for ℓ iterations. The removal phase ends when there is a node $i \in C$ such that $i \notin \mathcal{L}$ and $i \notin S$. At this stage, the addition or removal phase is started. At any iteration in this phase, a node i is added to S or a node j is removed from S. In both cases, the added, or removed, node is declared tabu for ℓ iterations. The choice of what to include in the tabu list, as well as setting values of the parameters β and ℓ, is based on experimental results for some test instances. All candidate nodes to be added to, or removed from, S are computed before any movement is made. The best candidate node to be added, or removed, is the one with the highest (lowest) contribution to (in) S. Any candidate node is either belong to $C\backslash\{S \cup \mathcal{L}\}$, which means it can be added, or belong to $S\backslash\mathcal{L}$, which means it can be removed. In the former case, for any candidate i we find

$$i = \operatorname{argmax}\left\{\sum_{j\in S}(x^*_{ij} + x^*_{ji}) + \lambda\left(\left\lceil\frac{Q(S\cup\{i\})}{Q}\right\rceil - \left\lceil\frac{Q(S)}{Q}\right\rceil\right)\right\}$$

which has the highest contribution to S, while in the latter case, for any candidate j we find

$$j = \operatorname{argmin}\left\{-\sum_{i\in S}(x^*_{ji} + x^*_{ij}) - \lambda\left(\left\lceil\frac{Q(S)}{Q}\right\rceil - \left\lceil\frac{Q(S\backslash\{j\})}{Q}\right\rceil\right)\right\}$$

which has the lowest contribution in S. An input parameter called λ, where $0 \leq \lambda \leq 1$, is used to consider the capacity of the candidate node. If the contribution of the best candidate node to be added, i, is bigger than or equal to the contribution of the best candidate node to be removed, j, we add i to the set S and add it to the tabu list $\mathcal{L}$; otherwise, we remove j from the set S and add it to the tabu list $\mathcal{L}$.

For every generated set S, the algorithm checks the CCCs of (11) for the porters, or (12) for the truck. Moreover, valid inequalities are going to be checked if S satisfies their nodes' type and other conditions, if exist. At every run of our separation algorithm, a new cut pool is check whether a proposed cut is already present in the current LP. That is to check if a cut is already added to the current LP or not. If it is, we ignore it, otherwise we add it to the pool, and keep looking for other cuts. At the end of this run, we add all unique cuts, from the cuts' pool, to the LP. We only accept constraints that are violated by at least 0.0001. Although the number of identified cuts might be very large at some iterations of the cutting plane algorithm, there is no limited number of added cuts to the LP at any iteration. Following some experiments, we conclude that a good choice for the parameters is: $\lambda = 0.9$, $\beta = 5|C|$, and $\ell = \left\lceil\frac{|C|}{2}\right\rceil$. These parameters are used in our experiments presented in the following section, Section 3.5. Algorithm 3.1 shows the steps of our separation procedure.

Algorithm 3.1 : Separation routine for (11), (12) and (25)–(34)

find connected components.
for each connected component C **do**
  pick a random $v \in C$, set $S = \{v\}$.
  set $iter = 1$ and $\mathcal{L} = \emptyset$.
  while $iter \leq \beta$ **do**
    add $i \in C \backslash \{S \cup \mathcal{L}\}$, or **remove** $j \in S \backslash \mathcal{L}$ such that

$$i = \operatorname{argmax} \left\{ \sum_{j \in S} (x^*_{ij} + x^*_{ji}) + \lambda \left(\left\lceil \frac{Q(S \cup \{i\})}{Q} \right\rceil - \left\lceil \frac{Q(S)}{Q} \right\rceil \right) \right\}$$

$$j = \operatorname{argmin} \left\{ - \sum_{i \in S} (x^*_{ji} + x^*_{ij}) - \lambda \left(\left\lceil \frac{Q(S)}{Q} \right\rceil - \left\lceil \frac{Q(S \backslash \{j\})}{Q} \right\rceil \right) \right\}.$$

    if $f(i) \geq -f(j)$, **then** set $S = S \cup \{i\}$, $\mathcal{L} = \mathcal{L} \cup \{i\}$.
    else set $S = S \backslash \{j\}$, $\mathcal{L} = \mathcal{L} \cup \{j\}$.
    determine S nodes type, **then** test associated inequalities.
    if a violation is found, **then** add violated constraints.
  end while
end for

3.4.2 Branching

In the branch-and-cut algorithm, branching occurs when the separation algorithm fails to identify at least one cut and the solution is not integer. The standard way of branching is to branch on any variable which has a fractional value, i.e., to select a fractional binary variable x^*_e and create two branches that correspond to setting $x_e = 0$ and $x_e = 1$, respectively. In our experiments, however, the priority is to branch on a fractional z^* variable. That because these variables are "bigger" decisions and branching on them at an early stage of the branch-and-cut tree leads to solving the problem much faster, as confirmed by our experiments.

3.5 Computational experiments

The algorithm was coded in C++ (Visual Studio 2017) using CPLEX Concert Technology (version 20.1.0) and run on the IRIDIS 5.0 High Performance Computing Facility of the University of Southampton, relying on a cluster of compute nodes equipped with dual Inter(R) Xeon(R) Gold 6130 CPUs @ 2.10GHz and 192 GB of DDR2 RAM using a single thread per experiment. This section is organised as follows. The next subsection gives details about the generation of our set of instances. Subsection 3.5.2 shows the effectiveness of each family of valid inequalities introduced in Section 3.3. Subsection 3.5.3 provides our comparative computational results.

3.5.1 Problem instances

In order to test the proposed algorithm, a set of TPRP instances is needed. However, there is no test bed available for TPRP in the literature and, thus, a set of instances has been created. The instances are built by sampling from a real-world instance called "Leuven1" in the VRP literature that is introduced by Arnold et al. (2019). For any instance of the TPRP, there are three types of customers: porter customers V_P, unconstrained customers V_U, and truck customers V_T. Along with the size of the instance, the number of customers of each type plays an important role in determining how difficult the instance is to solve to optimality. Thus, beside creating a set of instances with different sizes, instances with different percentages of customers were considered. The set of instances divided into three groups. The first group, indicated by A, contains 37.5%, 25%, and 37.5% of the nodes as V_P, V_U, and V_T nodes respectively. The second group, denoted as B, has 25%, 50%, 25% of the nodes are V_P, V_U, and V_T nodes respectively. In the third group, indicated by C, 75% of the nodes are V_U, 12.5% of the nodes are V_P, and 12.5% of the nodes are V_T.

Every group of instances contains nine different sizes. The sizes of the created instances, including the depot, are equal to 13, 16, 19, 22, 25, 28, 31, 34, and 37. For each size, five different instances were created. So, the total number of instances of each group is 45, and the total number of instances that have been created for this experiment is equal to 135. Table 3.1 shows the number of customers at each type of customers in every size and group. The process of creating the set of instances can be explained as follows. The original depot of Leuven1 is considered as the depot in all of the created instances. Customers are chosen randomly such that, the set of V_P customers is chosen to be within 300 meters away from the depot. Whereas, the set of V_U customers is chosen to be within 600 meters away from the depot. Unlike V_P and V_U customers, V_T customers can be anywhere in the graph.

The procedure starts by selecting the set of V_P customers since those customers are the hardest to find as the farthest customer of this type can only be 300 meters away from the depot. Once all V_P customers are chosen, we start looking for V_U customers which is the second hardest set of customers to be found. Those customers can be anywhere within 600 meters away from the depot, hence the reason to be the second hardest set of customers. The easiest set of customers to be found is the set of V_T customers. This because

Table 3.1: Number of customers at each type in every size and group.

	group A			group B			group C		
size	$\lvert V_P\rvert$	$\lvert V_U\rvert$	$\lvert V_T\rvert$	$\lvert V_P\rvert$	$\lvert V_U\rvert$	$\lvert V_T\rvert$	$\lvert V_P\rvert$	$\lvert V_U\rvert$	$\lvert V_T\rvert$
13	4	4	4	3	6	3	1	10	1
16	5	5	5	3	9	3	1	13	1
19	6	6	6	4	10	4	2	14	2
22	7	7	7	5	11	5	2	17	2
25	9	6	9	6	12	6	3	18	3
28	10	7	10	6	15	6	3	21	3
31	11	8	11	7	16	7	3	24	3
34	12	9	12	8	17	8	4	25	4
37	13	10	13	9	18	9	4	28	4

of the fact that these customers can be anywhere in the graph. During the procedure, once a customer is chosen, the coordinates and the demand of the chosen customer are considered. Arnold et al. (2019) assigned each customer to a demand of either one, two, or three parcels.

The first group of instances that was created is group C. This group has the most V_U customers and, therefore, instances of group A and B are created by the instances of group C. This means that when an instance of group C is created, as a result, an instance of group A and an instance of group B are created by the use of the instance of group C. This can be achieved by randomly moving some of the V_U customers to V_P and V_T. To decide which customers to remove from V_U, customers that are closer to the depot have higher probabilities to become V_P customers. Whereas customers that are further from the depot have higher probabilities to become V_T customers.

In the TPRP, the distance between any pair of nodes represents the travelling time between them. The time needed to travel between a pair of nodes is calculated by computing the Euclidean distance and, then, converted to time which requires to know the speed of travel. The average walking speed of an adult is between 3 to 3.2 miles per hour, and the average driving speed in a central large city like London is about 8.7 miles per hour (TFL, 2013). In this chapter, it was assumed that the speed of any porter is 3.2 miles per hour and the truck's speed is equal to 12.8 miles per hour. Thus, the cost of travelling from node i to node j, where $i, j \in V_U \cup \{0\}$ and $i \neq j$, by any porter, c_{ij}, is not equal to the cost of travelling from i to j by the truck, $\bar{c}_{ij}$.

Each customer is assigned to a cost which represents the time needed to serve that customer. For any customer i, where $i \in V_{PU}$, the cost of visiting i by a porter is equal to one minute. This cost is added to c_{ji} for any $j \in V_{PU}$ and its called the *service time* of customer i by the porters. If customer i is served directly from the depot, then the service time is equal to two minutes. However, there is no service time assign to the depot. That because porters are allowed to leave the depot multiple times, and the extra time is needed only when leaving the depot (e.g., time to pick some items). On the other hand, if customer i, where $i \in V_{TU}$, is visited by the truck, there is a randomly generated number between 2 and 5 minutes considered to be the service time for customer i by the truck. This number is going be added to $\bar{c}_{ji}$ for any $j \in V_{TU} \cup \{0\}$ and its called the *service time* of customer i by the truck. Note that there is no extra time for the truck to visit the depot nor to depart from it. That because of the fact that the truck has unlimited capacity, which means that there is a single route for the truck and there is no need for extra time to prepare for another route.

There is a single truck and m identical porters to serve customers. It was assumed that porters have the same carrying capacity Q_P, wage cost F_P, and limited working time T_P. In our experiments, we set the carrying capacity of a porter to be equal to 20 units (or parcels), that is $Q_P = 20$. The cost of adding an extra porter, the wage cost, is equal to 1000, that is $F_P = 1000$. And, the maximum porter's working time is 1800 seconds, so $T_P = 1800$. We, also, set the maximum number of porters, N_P, at any instance to be equal to the number of V_{PU} customers in V, that is $N_P = |V_{PU}|$.

3.5.2 Effectiveness of families of valid inequalities

The aim of each family of the proposed valid inequalities is to strengthen the LP relaxation. In order to assess the effectiveness of each family, we executed two experiments. On the first experiment, the separation routine is allowed to add valid inequalities from only one family at a time. On the second experiment, the separation routine is allowed to add valid inequalities from all families except from one at a time. At any experiment, the separation routine is allowed to add violated CCCs of (11) and (12). Both experiments end when the separation routine fails to find any cut at the root node. Computations were conducted on all of the instances of size 37. The average results are given in Table 3.2. We present the results of the first experiment on the first column, called *with one family*, and the results of the second experiment on the second column, called *without one family*.

The first column of Table 3.2, *with one family*, shows the average improvement on the lower bounds by each family of inequalities. Row *LR* shows the average linear programming relaxation of the tested instances without any family of inequalities. Row number i, where $i = \{1, 2, ..., 6\}$, represents the results obtained by separating family i only. We introduced six families of in-

Table 3.2: Average lower bound results at the root node obtained with/without one family of valid inequalities.

	with one family			*without one family*		
i	Impr(%)	time(s)	N cuts	IImpr(%)	time(s)	N cuts
LR	0.00	89.80	71.73	-	-	-
1	0.79	291.81	114.87	-0.60	478.49	945.60
2	0.22	81.83	74.00	-0.10	1074.02	974.60
3	0.73	169.13	118.87	-0.33	1608.85	1003.13
4	1.02	404.34	110.80	-0.37	810.70	1019.60
5	0.53	410.50	524.40	-0.10	463.46	686.13
6	1.11	524.71	689.80	-0.20	352.59	517.73
Full	2.36	921.30	1109.33	-	-	-

equalities in Section 3.3 in which each one represented by a row in this table. That is, family i is the family of inequalities that introduced by Proposition (i). The last row, row *Full*, shows the average improvement when we allow the separation routine to add violated inequalities from all the families. The second column of the table, *without one family*, shows the impact on the lower bounds when family i is not considered in the separation routine. In both columns, we presented the average improvement of lower bounds obtained under (Impr(%)) for the first column, under (IImpr(%)) for the second column, the average computation times in seconds under (time(s)), and the average number of added cuts under (N cuts). The value of (Impr(%)) for a single instance is calculated by Impr $= (LB_i - LB_{LR})/LB_{LR} \times 100$, where LB_i is the lower bound obtained at the root node with family i, and LB_{LR} is the lower bound obtained at the root node by solving the linear relaxation. On the other side, the value of (IImpr(%)) for a single instance is calculated by IImpr $= (LB_i - LB_{Full})/LB_{Full} \times 100$, where LB_i is the lower bound obtained at the root node without family i, and LB_{Full} is the lower bound obtained at the root node with all the families.

From the average results of *with one family*, we can see that the largest improvement on the lower bounds is 1.11% obtained by the sixth family. Family number four, one, and three come next in which they improved the lower bounds by 1.02%, 0.79%, and 0.73% respectively. Other families are not improving the lower bounds by more than 0.53%. On the other hand, the average results of *without one family* shows that without any family, the average lower bounds is not losing more than 0.60%. For more details about the number of add constraints from each family of inequalities in both experiments, see Table A1 and Table A2 in the appendix.

The fact that we are using a heuristic separation procedure can be time-saving, yet it is not efficient to measure the effectiveness of each family of valid inequalities. That because it might misses, or fails, to identify violated cuts at an optimal non-integer solution. From Table 3.2, we can see that when the separation procedure is allowed to check for all the families of inequalities, the average lower bounds increased by 2.36%. However, the average computation time raised by about 930%, and the number of added cuts increased by more than 1446%. So, there is a trade-off between including all the families of inequalities to obtain a slightly better lower bound on the one hand, and excluding all the families of inequalities to save time and to keep the problem's size smaller on the other hand. For this reason, the following section contains the results of the branch-and-cut algorithm with/without all the families of valid inequalities.

3.5.3 Computational results

The algorithm was tested on the 135 generated instances which have been subdivided into three groups, A, B, and C as explained in Subsection 3.5.1. It was mentioned that generated instances are vary in size as well as the number of each type of customers, and they were constructed from a well-known instance in the VRP literature introduced by Arnold et al. (2019). To evaluate our branch-and-cut algorithm, we attempted to solve the set of instances by five different methods:

$B\&C1$: is a branch-and-cut algorithm where the separation routine is separating the CCCs and all families of valid inequalities at every node of the branching tree;

$B\&C2$: is a branch-and-cut algorithm where the separation routine is separating the CCCs only at every node of the branching tree;

$B\&C3$: is the same as $B\&C1$ but we add the MTZ constraints, constraints (16)–(22), at the root node;

$C\&B$: is a branch-and-bound algorithm that call the separation routine to separate the CCCs and all families of valid inequalities at the root node only, so-called cut-and-branch; and

$B\&B$: is a branch-and-bound algorithm that uses the alternative formulation of the TPRP with the MTZ constraints, constraints (16)–(22).

The reason of testing the first two methods, $B\&C1$ and $B\&C2$, was mentioned in Subsection 3.5.2. The last three methods, $B\&C3$, $C\&B$, and $B\&B$ were performed for comparative purposes.

The key to success in a branch-and-cut algorithm is to have an efficient separation algorithm. The efficiency of the separation algorithm can be measured by the number and quality of violated constraints added to the problem. One way to evaluate the quality of added constraints is to observe the improvement obtained in the lower bound. By the fact that our tabu search procedure is not only trying to identify violated valid inequalities from the CCCs of (11) and (12), but it is also looking for any violation occurred by any family of valid inequalities, the efficiency of our separation algorithm is difficult to be measured. However, we compared between different tabu strategies and we decide to use the one that made the highest improvement on the lower bounds. Tabu strategies such as tabu removed and added nodes, tabu removed nodes only, and tabu added nodes only for ℓ iterations were examined. We also tried to use a dynamic tabu list in our comparison in which the size of the tabu list is increasing (decreasing) when we add (remove) a node to (from) any subset S. Our separation procedure which includes the chosen tabu strategy is explained in details in Section 3.4.

In our experiment, each instance was executed five times by each method. The computation time limit was set to two hours. Table 3.3 provides the results obtained by running each algorithm. Column 1 and 2 identify the size and the number of instances for which the codes were executed. The column

Table 3.3: Number of instances solved exactly and average computational time by each method.

		$B\&C1$		$B\&C2$		$B\&C3$		$C\&B$		$B\&B$	
size	inst	opt	time	opt	time	opt	time	opt	time	opt	time
13	15	15	56	15	52	15	236	15	53	15	985
16	15	15	306	15	337	15	540	15	355	13	1660
19	15	11	710	12	1321	11	1896	11	612	5	2563
22	15	8	2639	7	1902	7	2726	8	2161	1	1546
25	15	3	1791	4	3025	3	3856	2	916	0	0
28	15	1	1899	1	846	1	1569	1	1453	0	0
31	15	1	3182	1	2868	0	0	1	5257	0	0
34	15	2	4162	2	1306	0	0	2	2390	0	0
37	15	0	0	0	0	0	0	0	0	0	0

associated with each method shows the number of solved instances and the average computational time, in seconds, rounded to the nearest integer. Each computation time represents the average computational time in seconds to solve optimality-solved instances. The result of our experiments is reported in the appendix. Table A3 provides the column headings used in the following tables. The detailed results of $B\&C1$, $B\&C2$, $B\&C3$, $C\&B$, and $B\&B$ are reported in Table A4, A5, A6, A7, and A8 respectively.

Each row in Table A4–A8 is an average result of five different instances of the same type and size. For example, the first row with "N13_a" is about five instance of size 13 of type A. The five instances are named as "Leuven1_N13_a1", "Leuven1_N13_a2", ..., "Leuven1_N13_a5". In these tables, the best and the average lower bound (LB), upper bound (UB), and gap (gap(%)) values are reported. The number of instances with an optimal solution, out of 5, is given in the column (opt). In addition, the number of explored nodes in the branching tree, computation time of optimality-solved instances in seconds, and the total number of added cuts are reported in column (N nodes), (time(s)), and (N cuts) respectively.

From Table A4–A8, the best time needed to solve the smallest size instances of type A is 0.33 seconds achieved by $C\&B$. However, the best time to solve instances of type B of the same size is 20.81 seconds which was achieved by $B\&C2$. This growth of complexity occurred due to moving one customer from V_P to V_U and one customer from V_T to V_U. The complexity increased sharply with the instances of the same size of type C. The best time to solve these instances is 130.95 seconds and it was obtained by $C\&B$. It can be concluded that the more V_U customers in an instance, the more decision variables in the problem, and so the harder to be solved to optimality.

In order to compare the performance of each method, more details are needed. Table 3.4 gives a summary of running the five methods, where each method represented by a column. In this table, numbers are rounded to the nearest integer. There are 135 small-size instances. For 59 instances, we were able to find an optimal solution in at least one run by at least one of the five methods. The $B\&C2$ found optimal solution for 57 instances, and that is the highest number of instances solved to optimality by a single method. Note that $B\&C2$ explored more nodes than $B\&C1$ and $B\&C3$, thanks for ignoring the set of families of valid inequalities in the separation routine. The $B\&C1$ and $C\&B$ come next as they were able to solve 56 and 55 instances respectively. The $B\&C3$ and $B\&B$ were not able to solve more than 52 and 34 instances to optimality respectively.

As mentioned, each instance was solved five times by each method. This means that the total number of runs of each method is equal to 675. The third row of Table 3.4 shows the total number of runs that were successfully executed by each method. Out of 675 attempts, $B\&C1$ found 268 optimal

Table 3.4: Summary of the experiment.

	$B\&C1$	$B\&C2$	$B\&C3$	$C\&B$	$B\&B$
Number of instances solved to optimality	56	57	52	55	34
Number of unique-solved instances	1	2	0	1	0
Number of runs ended with an optimal solution	268	262	238	257	170
Number of nodes explored in the branching tree	3970	4050	2094	17872	7930
Average time to solve instances solved exactly	949	937	1245	790	1492
Number of cuts generated throughout the branching tree	306	305	207	140	0

solutions within the time limit. The $B\&C2$ comes next with 262 runs ended with an optimal solution. There are 257 runs ended with an optimal solution in $C\&B$. The last two methods, $B\&C3$ and $B\&B$, ended with the two lowest numbers of runs finished with an optimal solution.

In the same table, Table 3.4, the average number of nodes explored in the branch-and-cut tree for each method is reported in the fourth row. In addition, the average needed time to solve the set of instances which were solved to optimality is given in the fifth row. Note that $C\&B$ is the winner in terms of finding optimal solutions in the lowest amount of time. Last row, row number six, shows the average number of cuts added by the separation algorithm in each method. The time needed to solve each optimality-solved instances by each method is reported in Table A9 in the appendix.

Based on this experiment, there is no dominant method. However, we can say that $B\&C3$ and $B\&B$ are not competitive algorithms as they perform worse than other methods. To decide which method performed better than the others, unsolved instances can help. There are 76 unsolved instances, out of 135, in which each instance has five values of lower bound. Each lower bound value is obtained by one of the five methods. Table A10 contains the set of instances that has not been solved to optimality and shows the best lower and upper bound values for each unsolved instance. It also shows the methods that find the best upper and lower bounds. In the same table, the gap between the best lower bound and the best upper bound of each instance in reported in a percentage. The gap can be defined as gap(%) $= ((UB - LB)/LB) \times 100$.

Table A10 shows that the $C\&B$ is the winner in terms of finding the best lower bound for most of the instances that were not solved to optimality. The best lower bound of 59 instances, out of 76, was obtained by $C\&B$. The $B\&C2$ and $B\&C1$ were able to find the best lower bound for 8 and 7 instances respectively. The $B\&C3$ was able to find the best lower bound for only two instances, whereas the $B\&B$ was not able to find any best lower bound. It can be concluded that $C\&B$ perform better than other methods. The $B\&C2$ and $B\&C1$ are the second and the third best methods respectively. The $B\&C3$ comes in the forth place, whereas the branch-and-bound algorithm, $B\&B$, is the worst performance method. The table, Table A10, will be used for comparison purposes in the next chapter.

Exploring more nodes in the branch-and-cut tree can leads to more improvement in the lower bounds. So, it is important to decide when to call the separation algorithm in order to have an efficient branch-and-cut algorithm. In our experiment, we decided to call the separation algorithm at every node of the branch-and-cut tree in $B\&C1$, $B\&C2$, and $B\&C3$ and that is perhaps the reason of being less efficient than $C\&B$. We expect further improvement on our branch-and-cut algorithm performance when a better strategy of calling the separation routine is adopted. Strategy such as calling the separation algorithm every 100 node of the branch-and-cut tree instead of calling it at every node of the branching tree. Another way to improve the overall performance of our branch-and-cut algorithm is to add some advance futures to our tabu search algorithm like the aspiration criteria, that is to allow adding (removing) a node to (from) subset S even though its in the tabu list.

3.6 Conclusions of the chapter

The truck porters routing problem (TPRP) is a combination between driving and walking to serve a set of customers in urban areas. In the TPRP, there is a single truck and a limited number of identical porters available at a depot to visit every customer exactly once in which some customers are allowed to be visited by the truck only, some customers must be served by a porter only, and the remaining customers can be visited either by the truck or by a porter. The problem consists of designing a set of minimum-cost routes such that each route starts and ends at the depot and it must satisfy capacity and travel time constraints. The TPRP can be considered as a vehicle routing problem (VRP) variant. However, the complexity of this problem is beyond most of the known VRP variants. In this problem, porters are not allowed to exceed a specified amount of demand they can carry nor to travel more than a given time. In addition, porters can perform multiple trips. Such constraints add a great amount of complexity and, therefore, only small-size instances of the TPRP can be solved to optimality.

We introduced two mixed-integer programming formulations for this problem and several families of valid inequalities which are used within a branch-and-cut algorithm. A tabu search algorithm is designed and used for the separation procedure. Our branch-and-cut algorithm applied to solve randomly generated instances and it was able to solve to optimality instances with up to 16 nodes within a reasonable amount of computational time. Computational results are reported and it was used to measure the performance of the variable neighborhood search (VNS) heuristic introduced in the next chapter, Chapter 4. Although, the size of optimality solved instances are relatively small, they were useful to measure the efficiency of the proposed VNS algorithm. As most of the VRP variants, the problem can be attacked by more sophisticated exact methods like a branch-cut-and-price algorithm. In addition, different separation techniques may lead to better outcomes for the proposed branch-and-cut algorithm.

Chapter Four

A variable neighborhood search algorithm for the TPRP

The truck-porters routing problem (TPRP) is a generalisation of the traveling salesman problem (TSP) and is thus it is $\mathcal{NP}$-hard. Therefore, it is not possible to obtain an optimal solution for large-size instances within reasonable amount of computation times. Thus, to tackle this complex combinatorial problem, we propose an efficient metaheuristic approach. The variable neighborhood search (VNS)-based metaheuristic algorithms have proved to be successful in solving a variety of hard combinatorial problems. Therefore, a VNS algorithm is designed and implemented to tackle the problem. This chapter is organised as follows. Section 4.1 gives an overview of the proposed algorithm for the TPRP, including its main steps. A detailed explanation of the main steps of the proposed algorithm is given in Section 4.2. Results are reported and discussed in Section 4.3. Conclusions are given in Section 4.4.

4.1 Variable neighborhood search algorithm for the TPRP

The VNS-based metaheuristic algorithms have proved to be successful in solving many hard combinatorial problems. Therefore, a VNS algorithm is designed and implemented to solve large-size TPRP instances. A brief definition of the VNS heuristic is given in Chapter 2. The reader is referred to Hansen et al. (2019) for basic information and new developments of VNS related work including successful applications. In order to construct an initial solution for the TPRP, the problem is decomposed into three subproblems: one for those customers that are served by the truck, so-called truck customers; one for those customers that are served by any porter, so-called porter customers; and the others, denoted by unconstrained customers, which can be served either by the truck or by a porter. Each problem is, then, solved individually.

At the beginning, the first sub-problem, *sub-problem* 1, that contains the set of truck customers is solved. In this problem, a single truck with unlimited carrying capacity requires to visit each truck-customer exactly once, starting and ending at the depot. In another words, we are solving the famous TSP presented by Flood (1956). In the TSP, if the cost of going from customer i to customer j, c_{ij} , is equal to the cost of going from j to i, c_{ji}, for all customer i and customer j, the problem is said to be symmetric; otherwise, it is asymmetric. In our case, for the aim of solving the first problem, we are facing an asymmetric TSP. That because of the assumption in Chapter 3 which assign a cost, so-called a *service time*, for each customer. The service time of a truck-customer is between 2 to 5 minutes, and it represents the amount of time needed to serve that customer by the truck. One of the most successful heuristic algorithms for the TSP is the Lin-Kernighan heuristic (LKH) algorithm (Lin and Kernighan, 1973). The LKH is capable of solving both symmetric and asymmetric TSPs beside other traveling salesman and vehicle routing problems (Helsgaun, 2017). That because of the fact that a asymmetric TSP with n nodes can be transformed into a symmetric TSP with $2n$ nodes (Jonker and Volgenant, 1983). Therefore, the LKH algorithm is used to solve sub-problem 1.

At this stage, the procedure of solving the second sub-problem, *sub-problem* 2, that contains porter customers, is started. There are m identical porters required to visit every porter-customer exactly once. It was assumed that porters are limited by the total weight of items that they can carry and by a total working time constraint. However, a porter can re-visit the depot to collect further items for delivery. It was also assumed that the number of available porters, N_P, is equal to the number of V_{PU} customers at any instance. A well-know variant of the classical vehicle routing problem (VRP) is the multi-trip VRP (MTVRP) that aims at determining a set of trips and an assignment of each route to a vehicle such that the total travel time is minimised and the following conditions are satisfied:

(1) each trip starts and ends at the depot;

(2) each customer is visited exactly once;

(3) the sum of the demands of the customers in any trip does not exceed the predetermined vehicle's capacity;

(4) the sum of the durations of the trips assigned to the same vehicle (route) does not exceed the pre-set time limit.

This means that solving sub-problem 2 is actually solving the MTVRP. The MTVRP was reviewed in Chapter 2. Olivera and Viera (2007) proved that the MTVRP is $\mathcal{NP}$-hard as being a generalisation of the VRP. In their proof, they assumed that an unlimited fleet of vehicles is available in the VRP. However, Cattaruzza removed this assumption and propose a more formal proof in Cattaruzza et al. (2018). One of the most common way to solve the MTVRP is to combine VRP and bin packing (BP) algorithms (Fleischmann, 1990, Olivera and Viera, 2007, Petch and Salhi, 2003, Salhi and Petch, 2007, Taillard et al., 1996). To solve this problem, a two-stages algorithm is implemented and used. In the first stage, the Clarke and Wright (1964) savings algorithm is applied to create a VRP solution. In the second stage, the best fit decreasing (BFD) heuristic proposed by Johnson (1974) is called to assign tripe to the vehicles. A detailed explanation of the proposed two-stages algorithm is given in the next section, Section 4.2.

Algorithm 4.1 : The constructive heuristic for the TPRP

solve the ATSP with truck nodes, and the MTVRP with porter nodes.
repeat
    for each unassigned node i **do**
        for each feasible position for i into existing routes **do**
            find the best feasible position.
        end for
    end for
    insert the node with the lowest additional cost into it is best position.
until all unconstrained customers are inserted.

Once sub-problem 1 and sub-problem 2 are solved, the third sub-problem, *sub-problem* 3, that deals with the set of unconstrained customers is solved. In this problem, we are given a single truck's route and a number of porters' routes obtained from solving sub-problem 1 and sub-problem 2 respectively. Every route consists of at least a single trip. We refer to a *trip* as a sequence of customer services preceded and followed by a visit to a depot. We call a sequence of trips performed by the same porter a *route*. In the literature *trip* and *route* can, for example, be respectively referred to *trip* and *journey* (Cattaruzza et al., 2018), *route* and *schedule* (Mingozzi et al., 2013), or *tour* and *multi-tour* (Aghezzaf et al., 2006). In this problem, the set of unconstrained customers is inserted into existing routes with the lowest possible additional cost. To solve this problem, we apply the following procedure. For every unconstrained-customer i, compute the cost of inserting i into every feasible insertion position in existing routes. The customer with the minimum additional cost is selected to be inserted at the best feasible insertion position, and the procedure is repeated until all unconstrained customers are included into existing routes. During this procedure, inviting a new porter is not allowed. However, porters are able to start a new trip, if possible. A summary of the proposed constructive heuristic is given in Algorithm 4.1.

Algorithm 4.2 : The VNS for the TPRP

define the set of shaking procedures N^k, for $k = 1, ..., k_{\max}$ and the set of local search operators R_l, for $l = 1, ..., l_{\max}$.
construct an initial solution, x, by applying Algorithm 4.1.
apply the VND algorithm to improve the current solution x.
repeat
  set $k = 1$.
  repeat the following steps:
    Shaking: Generate a solution x' at random from the k^{th} shaking procedure of $x(x' \in N_k(x))$.
    Local search: Apply the VND algorithm with x' as initial solution to find the best neighboring solution x''.
    Move: If x'' yields a better quality solution, **then** set $x = x''$, $iter = 1$, $k = 1$, otherwise set $k = k + 1$.
  until $k = k_{\max}$
until the pre-set time limit is reached.

Once unconstrained customers are included into existing routes, an attempt to improve x is carried out by applying local search operators described in details afterwards in this section. The resulted solution, x_{impr}, is then set as the incumbent solution $x_{best} = x_{impr}$. At this moment, we start our VNS algorithm with x_{best}. At every iteration of the VNS, there are three main steps. The first step is the *perturbation step* where a new solution x' is constructed by shaking the incumbent solution x_{best} by one of the *shaking procedures*. The aim of the perturbation step is to escape from local optima as it allows diversification in the search space. In the second step, known as the *descent step*, we apply the *local search procedures*, also called *neighborhoods*, to find the best neighboring solution, x''. Finally, at the third step, we compare the cost obtained in step two, $f(x'')$, with the cost of the incumbent solution, $f(x_{best})$. If $f(x'') < f(x_{best})$ we set $x_{best} = x''$, otherwise the VNS starts the next iteration. The algorithm is terminated once the pre-set time limit is reached. Algorithm 4.2 shows the designed VNS algorithm. Shaking and local search procedures are described in details in the following section.

4.2 Explanation of the main steps

4.2.1 Initial solution

The construction of the initial solution is started by solving the TSP for the set of V_T nodes. The Lin and Kernighan (1973) heuristic (LKH) is known to be one of the state-of-the-art local search algorithms for the TSPs and, therefore, it was used to solve the problem. The LKH algorithm uses k-opt moves to optimise the solutions. The k-opt moves explores the solution space by replacing k edges of the current trip, where k is any integer greater than or equal to 2 and less than the number of nodes (the number of V_T nodes).

Once the TSP is solved, the Clarke and Wright (1964) savings algorithm is applied to create a VRP solution for the set of V_P nodes. Clarke and Wright (1964) saving algorithm is one of the most widely known heuristic for the VRP, and it can be briefly described as follows:

Step 1: create n trips, $0 \to i \to 0$, for $i = 1, ..., n$;

Step 2: compute the savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ for $i, j = 1, ..., n$ and $i \neq j$;

Step 3: order the savings in a non-increasing order;

Step 4: starting from the top of the savings list with s_{ij}, determine whether merging the two separate trips, one contains the arc $(0, j)$ and the other contains arc $(i, 0)$ is feasible. If so, join these trips by removing $(0, j)$ and $(i, 0)$ and introducing (i, j);

Step 5: repeat step 4 until no additional savings can be achieved.

At this stage, every trip of the VRP solution is assigned to a unique porter. To reduce the number of porters, the best fit decreasing (BFD) heuristic proposed by Johnson (1974) is used to assign trips to porters. In the BFD algorithm, items (trips) are sorted in a decreasing order according to their sizes (costs). Then, each item (trip) is placed into the fullest bin (route) in which it fits, without exceeding the bin capacity (porters travel time T_P).

Finally, the best insertion algorithm is used to insert the remaining nodes, V_U nodes, into existing routes. As it was mentioned in Section 4.1, the procedure inserts a single V_U node at a time. At every iteration, the cost of inserting node i, where $i \in V_U$ such that i is not part of the truck route or porters' routes, is computed for every feasible insertion position. The node with the lowest additional cost is selected to be inserted at the best feasible insertion position. The procedure is repeated until every V_U node is included.

4.2.2 Shaking procedures

The VNS escapes from local optima by applying shaking procedures to the current local minimum. The first step of the VNS for the TPRP is to create a neighboring solution to the current local minimum by one of the shaking procedures. There are three neighborhoods used in our VNS, $k_{max} = 3$, namely the *remove-insert procedure*, the *trips initiation procedure*, and the *perturbation mechanism procedure*. These neighborhoods were ordered as follows: the remove-insert procedure is used as N_1, the trips initiation procedure is used as N_2, and the perturbation mechanism procedure is used as N_3. These procedures can be described as:

N_1: the remove-insert procedure is aimed at generating a feasible solution by removing and, then, re-inserting some customers. The procedure works as follows:

1- choose a number of m customers randomly, and list them in L;

2- remove chosen customers from their current positions in the trips;

3- if $L \neq \emptyset$, choose the first customer in L, call it i, otherwise halt;

4- find the best three feasible insertion positions for i into existing trips;

5- if there is no feasible insertion position for i go to (7), else go to (6);

6- insert i at one of the best three insertion positions, and go to (8);

7- invite a new porter to visit i, that is $0 \to i \to 0$;

8- remove i from L, that is $L = L \backslash \{i\}$, and go to (3).

In this procedure, a customer might be inserted at the first, second, or third best feasible position in step (6). The probability of choosing an insertion position depends on the cost of insertion. The position with less additional cost is more likely to be chosen. Therefore, the cheapest insertion position is always have the highest chance to be selected.

N_2: the trips initiation procedure aims at generating a feasible solution by greedily initiating k trips, where $1 \leq k \leq \lceil \frac{t}{2} \rceil$ and t is the number of trips in the current local minimum. The procedure works as:

1- pick a customer randomly. Call it i;

2- start a new trip, $\hat{T}$, to visit i such that $\hat{T} : 0 \to i \to 0$;

3- remove i from its original trip;

4- find the three closest V_{PU} customers to i, and list them in L;

5- if $L \neq \emptyset$ go to (6), otherwise go to (1) until k trips are created;

6- insert j where $j \in L$ after i in $\hat{T}$. Set $i = j$ and $L = \emptyset$, and go to (3).

In a greedy manner, a number of k trips are created in this procedure. Once a customer, i, is chosen to be visited, one of the three closest V_{PU} customers to i that are reachable by the porter who serves i is going to be visited. The closer customer to i, the higher chance to be visited after i.

N_3: the perturbation mechanism procedure is a scheme that was initially developed by Salhi and Rand (1987) for the VRP. In this procedure, three trips are considered simultaneously. The idea is to systematically take a customer from a trip and relocate it into another trip without considering capacity and time constraints in the receiver trip. A customer from this receiver trip is then shifted to the third trip if both capacity and time constraints for the second and the third trips are not violated. In our experiments, the procedure is repeated n times every time this procedure is executed.

4.2.3 Local search operators

In the VNS, local search operators, also called *neighborhoods*, are applied once the current local minimum is perturbed by one of the shaking procedures for the aim of improving the current solution x. In this step, neighborhoods are applied sequentially in a deterministic way. Such a method is called a variable neighborhood descent (VND) algorithm. In a VND algorithm, the way of ordering the set of neighborhoods plays an important role in the algorithm, therefore different orders have been tested. The most typical VND variants that traverse the list of neighborhoods in a sequential way are briefly described in Chapter 2. The reader can refer to Gendreau et al. (2010) for more details about VND variants. Any VND procedure starts from a given solution x and stops when there is no improvement with respect to any of the considered neighborhoods. For the TPRP, after some experiments, a good choice of VND variant is the pipe VND (PVND). The PVND stops the search in a neighborhood when there is not any improvement detected, otherwise its continue the search in the same neighborhood. The procedure stops when there is no improvement with respect to any of the neighborhoods.

Our VND method uses fourteen neighborhoods which are briefly described here. There are three different types of procedures: *intra-trip*, *intra-route*, and *inter-trip*. The first type, intra-trip, procedures are applied to each trip individually. The second type, intra-route, procedures are applied to each pair of trips belong to the same route. Whereas the third type, the inter-trip procedures, is applied for each pair of trips that do not belong to the same route. These neighborhoods are:

The 1-insertion procedure (intra-trip, intra-route, inter-trip): These neighborhoods try to reduce the total cost of the current solution by removing a customer from its position and re-insert it at the best feasible position. When a customer is removed from a trip, we check every possible insertion position: into the same trip in the intra-trip procedure; into other trips within the same route in the intra-route procedure; into other trips in different routes in the inter-trip procedure.

The 2-insertion procedure (intra-trip, intra-route, inter-trip): This is similar to the 1-insertion procedure except that we consider two consecutive customers instead of one in all of the procedures.

The swap procedure (intra-trip, intra-route, inter-trip): In these neighborhoods, we look for an improved solution by swapping a pair of customers. In the intra-trip procedure, customers belong to the same trip. In the intra-route procedure, customers belong to the same route, but not the same trip. Finally, customers belong to different routes in the inter-trip procedure.

The 2-opt procedure (intra-trip, intra-route, inter-trip): These neighborhoods aim at reducing the total cost by selecting, removing, and replacing two non-adjacent arcs by other two arcs. Selected arcs belong to: the same trip in the 2-opt intra-trip procedure; different trips in the same route in the 2-opt intra-route procedure; different routes in the 2-opt inter-trip procedure. The 2-opt intra-trip procedure is the

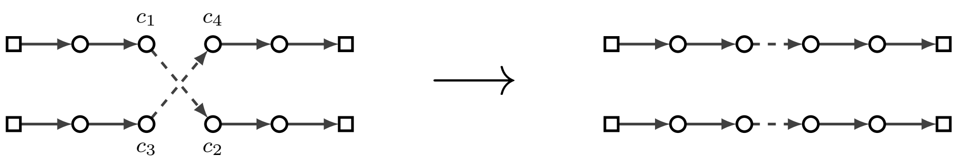

Figure 4.1: Illustration of the implemented 2-opt intra-route and inter-trip procedures. A 2-opt move where two arcs from two different trips are replaced with another two arcs within the route, in intra-route procedure, or with different routes, in inter-trip procedure. We only consider the exchange of arcs if the removal and addition of indicated arcs results in an improvement, i.e., $C_{c_1c_2}+C_{c_3c_4}-C_{c_1c_4}-C_{c_3c_2} > 0$.

2-opt algorithm propose by Croes (1958). It works by replacing two arcs with other two new arcs and reverting the direction of one of the resulting two sub-paths. In the 2-opt intra-route and inter-trip procedures, we consider two trips, x and y, simultaneously. The idea is to remove an arc from each trip to create four sub-paths x_h, x_t, y_h, and y_t such that $x = x_h \rightarrow x_t$ and $y = y_h \rightarrow y_t$. Then, connect the x_h with y_t and y_h with x_t, that is to make $x_h \rightarrow y_t$ and $y_h \rightarrow x_t$. The resulting trips might be shorter than the original trips and the total cost is therefore reduced. Any pair of trips belong to the same route will be checked by the 2-opt intra-route procedure, otherwise the 2-opt inter-trip procedure will handle it. Figure 4.1 illustrates the 2-opt intra-route and inter-trip procedures.

The best-fit-decreasing heuristic: The best fit decreasing (BFD) heuristic is used to assign trips to porters. The BFD heuristic among with other heuristics to solve the bin packing problem (BPP) are proposed by Johnson (1974). In the BFD algorithm, items (trips) are sorted in a decreasing order according to the size (trip's cost). Then, we assign each item (trip) to the fullest bin (route) in which it fits. If an item (trip) does not fit in any bin (route), then start a new bin (route).

The route-destruction procedure: The route-destruction procedure tries to reduce the number of invited porters by distributing some (or hopefully all) customers from a route into other routes. A summary of the procedure is given in Algorithm 5.1.

Algorithm 4.3 : The route-destruction procedure

1: **sort** the routes according to their lengths in an increasing order.
2: **for each** route R **do**
3:     **name** the current solution without R as $\mathcal{S}$.
4:     **repeat**
5:         **measure** the efficiency, e, of each node in R, where $e_i = l_R - l_{R \setminus i}$.
6:         **sort** the nodes according to the efficiency in a decreasing order.
7:         **remove** the node i which has the highest efficiency from R and insert it at the cheapest position in $\mathcal{S}$.
8:     **until** there is no node in R can be removed.
9:     **for each** route $\bar{R} \in \mathcal{S}$ **do**
10:         **if** $l_R + l_{\bar{R}} \leq T$, **then** assign R to route $\bar{R}$'s porter. Go to line 1.
11:     **end for**
12: **end for**

In our implementation, neighborhoods are placed in a list with a given order and always explored in that order. Neighborhoods are explored in the following order: the 1-insertion (inter-trip) procedure as R_1, the 2-insertion (inter-trip) procedure as R_2, the swap (inter-trip) procedure as R_3, the 2-opt (inter-trip) procedure as R_4, the 1-insertion (intra-trip) procedure as R_5, the 2-insertion (intra-trip) procedure as R_6, the swap (intra-trip) procedure as R_7, the 2-opt (intra-trip) procedure as R_8, the BFD procedure as R_9, the route-destruction procedure as R_{10}, the 1-insertion (intra-route) procedure as R_{11}, the 2-insertion (intra-route) procedure as R_{12}, the swap (intra-route) procedure as R_{13}, the 2-opt (intra-route) procedure as R_{14}.

The process starts by exploring the first neighborhood R_1. Once an improvement has been detected, we continue the search in the same neighborhood. Otherwise, the next neighborhood, R_2, is explored and the procedure is repeated. If an improvement is found in neighborhood R_k, where $2 \leq k \leq 8$, the search starts over and return to the first neighborhood in the list. Going to the top of the list happens when an improvement is found by R_k and there is no further improvement is detected by the same neighborhood. Once the first eight neighborhoods are explored without finding any improvement, the BP heuristic is called to assign trips to porters in R_9. Once the BP heuristic is applied, the solution is now consists of a truck route and a set of porters' routes. The aim now is to reduce the number of porters as well as improving the solution quality. Therefore, it is more reasonable to place the route-destruction procedure and intra-route procedures at this stage. Note that, once we reach this stage, returning to the first eight neighborhoods is not allowed. In addition, intra-route procedures, R_{11}–R_{14}, are only needed when the route-destruction procedure obtained an improvement. Indeed, intra-route procedures are not going to detect any improvement when R_{10} fails to improve the current solution, thanks for intra-trip procedures.

4.3 Computational results

Our VNS-based algorithm was coded in C++, compiled with Visual Studio 2017 and run on the IRIDIS 5.0 High Performance Computing Facility of the University of Southampton, relying on a cluster of compute nodes equipped with dual Inter(R) Xeon(R) Gold 6130 CPUs @ 2.10GHz and 192 GB of DDR2 RAM using a single thread per experiment. Three computational experiments are carried out in this chapter. In any experiment, the algorithm terminates when a pre-set time limit is reached. The time limit of solving an instance with n customers is set to be equals to $4n$ seconds. In addition, for comparison purposes, each instance is solved five times. The order of the local search operators as well as the order of the shaking procedures are fixed and used, as described in Section 4.2, in our experiments. We set m equals to 30% in N_1 and $n = 5$ in N_3. These values are good choices to ensure the intensification and diversification of the search as our preliminary experiments confirmed. For more convenience, tables of this section are reported in the appendix. This section is organised as follows. Subsection 4.3.1 contains the result of solving small-size instances created in Chapter 3. The result of solving the MTVRP benchmark instances is reported in Subsection 4.3.2. Subsection 4.3.3 shows the result of solving a set of large-size instances that is created in the same way of creating the set of small-size instances.

4.3.1 Solving small-size instances

In the first experiment, the algorithm is tested on the set of small-size instances created in Chapter 3. The aim of this experiment is to measure the performance of our VNS algorithm. As it was mentioned, the set of instances is sampled from a well-known instance in the VRP literature called "Leuven1" which is introduced by Arnold et al. (2019). These instances vary in size from 13 to 37 nodes, and they can be divided into three groups, namely A, B, and C. Instances of group A contains 37.5%, 25%, and 37.5% of the nodes as V_P , V_U , and V_T nodes respectively. Instances of the second group, group B, has 25%, 50%, 25% of the nodes as V_P , V_U, and V_T nodes respectively. Finally, instances of group C contain 75% of the nodes as V_U nodes, whereas 12.5% of the nodes as V_P , and 12.5% of the nodes as V_T.

There are 59, out of the 135, small-size instances solved to optimality by our branch-and-cut algorithm in Chapter 3. The proposed VNS algorithm was able to find the optimal solution for all optimality-solved instances. Table A11 shows the result obtained in this experiment, where the column instance and optimal show the instance name and the objective function value of the optimal solution respectively. The column gap(%) reports the best, worst, and average gap values over the runs yielding a feasible solution. The best, worst, and average gap value is obtained by comparing the optimal value of the considered instance with the best, worst, and average solution value obtained by our heuristic algorithm respectively. Each value is computed as $\frac{heuristic-optimal}{optimal} \times 100$, where $heuristic$ is the solution value obtained by our algorithm and $optimal$ is the optimal solution value. Column fs reports the number of runs ended with a feasible solution.

The algorithm is also tested on the remaining 76 small-size instances that are not solved to optimality by our branch-and-cut algorithm within the time limit. The algorithm is tested against the best lower and upper bounds found in the previous chapter. Table A12 contains the results of this experiment. The fist column represents the instance name. Column (2) and (6) show the best lower and upper bounds obtained in Chapter 3 respectively. In column (3), (4), and (5) the best, worst, and average gap values are reported. The best, worst, and average gap value is obtained by comparing the best lower bound value of the considered instance with the best, worst, and average value obtained by our heuristic algorithm respectively. Each value is computed as $\frac{heuristic-best_{LB}}{best_{LB}} \times 100$, where $heuristic$ is the solution value obtained by our heuristic algorithm and $best_{LB}$ is the best lower bound value found in the previous chapter. In a similar fashion, a value in column (7), (8), and (9) calculated as $\frac{heuristic-best_{UB}}{best_{UB}} \times 100$, where $heuristic$ is the solution value obtained by our heuristic algorithm and $best_{UB}$ is the best upper bound value found in the previous chapter. Our VNS algorithm was able to find, or mostly beat, the best upper bound found by our exact algorithm in with an average improvement of 6.70% on the upper bound values.

4.3.2 Solving the MTVRP instances

This experiment conducted on the set of the multi-trip vehicle routing problem (MTVRP) benchmark instances. The MTVRP (with unlimited number of vehicles) is a special case of the TPRP when the number of V_{TU} customers is zero, $|V_{TU}| = 0$. The MTVRP is reviewed in Chapter 2.

The MTVRP instances are introduced by Taillard et al. (1996) and constructed from the instances 1–5 and 11–12 proposed by Christofides et al. (1979) and from the instances 11–12 created by Fisher (1994) for the VRP. The instances are named as CMT1–CMT5, CMT11–CMT12, and F11–F12 respectively in the MTVRP literature. Table 4.1 shows the characteristics of the MTVRP benchmark instances. Column name, n, m, Q, and z^* show the name of the instance, number of nodes, number of available vehicles, vehicle's capacity, and the solution cost of the original VRP instances obtained by Rochat and Taillard (1995) respectively. For each VRP instance,

Table 4.1: Characteristics of the MTVRP benchmark instances.

name	n	m	Q	z^*
CMT1	50	1,...,4	160	524.61
CMT2	75	1,...,7	140	835.26
CMT3	100	1,...,6	200	826.14
CMT4	150	1,...,8	200	1028.42
CMT5	199	1,...,10	200	1291.44
CMT11	120	1,...,5	200	1042.11
CMT12	100	1,...,6	200	819.56
F11	71	1,...,3	30,000	241.97
F12	134	1,...,3	2,210	1162.92

instances for the MTVRP are generated with different values for the number of available vehicles m and two different values for the maximum time duration of a vehicle T_H, given by $T_H^1 = \left[\frac{1.05z^*}{m}\right]$ and $T_H^2 = \left[\frac{1.1z^*}{m}\right]$. There are, in total, 104 different instances. For 42 of them, the optimal solution is known and provided by Mingozzi et al. (2013). Cattaruzza et al. (2014) classified them in a group named as $G1$. For 57 instances, the optimal solution is not known but they have a known feasible solution. These instances belong to the second group, denoted by $G2$, and the state-of-the-art results is presented in Cattaruzza et al. (2018). Several best known solutions are found by François et al. (2016) that, however, are not included in their survey. In our comparison, we consider the best known solutions as reported in either papers. For the remaining 5 instances, there is no known feasible solution and they form the third group, group $G3$.

Table A13 and A14 report the results obtained by solving group $G1$ and $G2$ instances respectively. In both tables, the first column indicates the instance name. Column m and T_H show the number of available vehicles and the maximum time duration of each vehicle respectively. Optimal values are reported in column *optimal* in table A13, whereas best known solutions values are reported in column $best_{UB}$ in table A14. Column $best$, $worst$, and av report the best, worst, and average values over the runs yielding a feasible solution. In table A13, each value is expressed as a percentage and computed by comparing the solution value obtained by our heuristic algorithm and the optimal value of the considered instance. A percentage, or a gap, is calculated as $\frac{heuristic - optimal}{optimal} \times 100$, where *heuristic* is the value obtained by our heuristic algorithm and *optimal* is the optimal value of considered instance. In table A14, each value is also expressed as a percentage, or a gap, and calculated as $\frac{heuristic - best_{UB}}{best_{UB}} \times 100$, where *heuristic* is the value obtained by our algorithm and $best_{UB}$ is the best known value. Column opt in table A13 shows the number of runs ended with an optimal solution. In both tables, column fs represents the number of runs ended with a feasible solution.

For the 42 optimality-solved instances, the algorithm is able to find a feasible solution for all instances except for one, namely the $CMT1\_T_H^2\_4$. In all the 210 runs, feasible solutions are found 205 times and optimal solutions found 130 times. On average, the best, worst, and average gap value is 0.05%, 0.41%, and 0.20% respectively. For the 57 instances of group $G2$, the algorithm is able to find a feasible solution in at least one time, out of five times, on 52 instances with an average gap value equals to 0.87%. Nevertheless, the proposed algorithm is able to find five new feasible solutions. Table 4.2 shows new feasible solution values for improved instances. It also shows gap values of the new feasible solutions. These values are reported in bold in table A14. Table A15 contains the new feasible solutions for the five instances where v, t, τ_t, and l_t indicate the vehicle, the trip, its travelling time and its load. A fair comparison between the performance of our algorithm and the state-of-the-art algorithms can be performed if $CMT1\_T_H^2\_4$ is solved by our algorithm within the time limit. Cattaruzza et al. (2014) obtained feasible solution for all instances of group $G1$ with an average best-gap equal to 0.03% by the memetic algorithm that uses combined local search (MA+CLS) proposed in his paper. Whereas, the average best-gap of our algorithm is equal to 0.05%, however there is one instance without a feasible solution. Our VNS algorithm

Table 4.2: New feasible solution values.

instance name	previous best known	new best known	gap(%)
$CMT4\_T_H^1\_1$	1031.00	1028.43	-0.25
$CMT4\_T_H^1\_5$	1029.65	1029.16	-0.05
$CMT4\_T_H^2\_1$	1031.07	1029.65	-0.14
$CMT4\_T_H^2\_4$	1031.07	1028.78	-0.22
$CMT4\_T_H^2\_5$	1030.86	1029.65	-0.12

is not particularly designed to tackle the MTVRP, hence the reason of not obtaining feasible solutions for some MTVRP benchmark instances that are known to be feasible within the time limit.

4.3.3 Solving large-size instances

The third experiment carried out on a set of large-size instances. The set of large-size instances is sampled from the well-known instance "Leuven1" introduced by Arnold et al. (2019). To create the set of large-size instances, we use the same method used in Chapter 3 to create the set of small-size instances. However, the size of the instances is equal to 100, 200, or 300. Also, for the aim of providing a more comprehensive experiment, the new set of instances is created by four different methods of customer positioning. The four methods can be briefly described as follows:

Random (R): customers are chosen randomly;

Clustered ($C1$): an instance of this type is created by clustering V_U customers. First, the number of clusters c is determined using a uniform discrete distribution $UD\left[\frac{|V_U|}{20}, \frac{|V_U|}{10}\right]$, where $|V_U|$ is the number of V_U customers in the instance. Once c is determined, a number of nodes equals to k, where $k = c$, are chosen randomly. Every chosen node is considered as a cluster and the following procedure is repeated until the total number of V_U nodes reaches $|V_U|$. Considering a cluster at a time, find the nearest node to any V_U nodes belong to the current cluster. Once a node i is found, we set $V_U = V_U \cup \{i\}$ and then go to the next cluster. Then, V_P and V_T customers are chosen randomly;

Clustered ($C2$): each group of customers is clustered independently. In this type, each type of customers is clustered in the same way of clustering V_U nodes in $C1$;

Clustered (C3): customers are clustered in k clusters, where k is chosen from an uniform discrete distribution $UD\left[\frac{n}{20}, \frac{n}{10}\right]$ and n is the total number of customers needed in the instance. Note that, a cluster of this type may contains more than one type of customers.

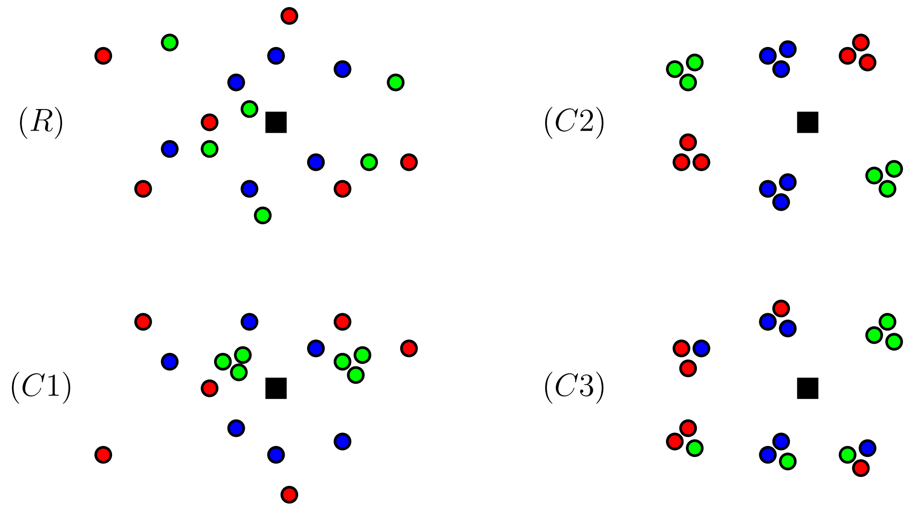

Figure 4.2: Illustration of the four methods used to create the set of large-size instances. In all figures, the square is a depot, the red dots are truck customers, the green dots are unconstrained customers, and the blue dots are porter customers. In the top-left figure, customers are chosen randomly. The bottom-left figure shows the first type of clustered instances, $C1$, where V_U customers are clustered and other types of customers are chosen randomly. The top-right and the bottom-right figures illustrate $C2$ and $C3$ respectively.

In any method, V_P customers are chosen to be within 300 meters away from the depot, V_U customers are chosen to be within 600 meters away from the depot, and V_T customers can be anywhere in the graph. Regardless of the size of the instances and the ways of customer positioning, instances are grouped intro three groups, namely A, B, and C. The difference between these groups is the number of customers of each type of customers as explained earlier in this section. The size and the number of customers of each type of customers are reported in Table 4.3. For each customer positioning method, there are

Table 4.3: Number of customers at each type in every size and group.

	group A			group B			group C		
size	$\lvert V_P\rvert$	$\lvert V_U\rvert$	$\lvert V_T\rvert$	$\lvert V_P\rvert$	$\lvert V_U\rvert$	$\lvert V_T\rvert$	$\lvert V_P\rvert$	$\lvert V_U\rvert$	$\lvert V_T\rvert$
100	37	26	37	25	50	25	12	76	12
200	75	50	75	50	100	50	25	150	25
300	112	76	112	75	150	75	37	226	37

45 instances in which there are 5 instances of each size and group. The total number of large-size instances is equal to 180. The experiment, indicated as $TPRP_1$, aims at testing the proposed heuristic algorithm by solving these instances. Another aim of this experiment is to measure the benefit of having some flexible customers which can be either visited by the truck or by a porter, the set of V_U customers. One possible way is to assign V_U customers to the truck at the one hand, denoted as $TPRP_2$. On the other hand, V_U customers are assigned to the porters, denoted as $TPRP_3$. Then, a comparison can be performed to compute the additional cost of each experiment when compared to the $TPRP_1$. Each instance in this experiment is solved with eight different parameter settings, or scenarios. At every scenario, porter carrying capacity Q_P, travel time duration T_P, and wage cost F_P are taking a unique combination. First three columns of Table 4.4 show the eight combinations. Column V_U-nodes shows the percentage of V_U customers that are served in $TPRP_1$ by porters. The average number of invited porters to solve the original instances, $TPRP_1$, is given in column P-number. Column $TPRP_2$ and $TPRP_3$ show the additional cost, in percentages, of assigning V_U customers to the truck and to the porters respectively. Last column shows the table's name in which the result is reported in the appendix.

Table 4.4: Summary of results.

Q_P	T_P	F_P	V_U-nodes	P-number	$TPRP_2$	$TPRP_3$	table
20	1800	100	95.41	11.49	22.89	0.72	A16
		1000	74.20	8.65	9.72	4.73	A16
	3600	100	96.15	5.40	24.06	0.60	A17
		1000	89.14	4.82	17.15	2.11	A17
40	1800	100	97.26	9.27	26.24	0.51	A18
		1000	84.94	8.01	14.60	3.56	A18
	3600	100	98.95	5.38	29.80	0.10	A19
		1000	96.15	4.56	22.50	0.58	A19

It can be seen from Table 4.4 that a smaller value for wage cost, F_P, leads to invite more porters and, as a result, more V_U customers are served by porters. Increasing the travel time duration for porters, T_P, drives porters to perform longer trips and, thus, the possibility of visiting more V_U customers increase. The results confirm that increasing the capacity of a porter, Q_P, raise the number of V_U customers visited by porters while the number of needed porter is decreased. This means that porters are getting busier as the carrying capacity increase. In case that $(Q_P, T_P, F_P) = (20, 1800, 1000)$, where porters are given a small carrying capacity and travel time duration, 25.80% of V_U customers are assigned to the truck. In the contrary, only 1.05% of V_U customers are assigned to the truck when porters have larger carrying capacity, travel time duration, and inviting an additional porter is not expensive, that is when $(Q_P, T_P, F_P) = (40, 3600, 100)$.

The majority of V_U customers are served by porters in $TPRP_1$. In $TPRP_2$, the set of V_U customers is assigned to the truck. As a result, the additional cost of solving the set of instances in $TPRP_2$ is more than 22% in most cases. In $TPRP_3$, the set of V_U customers is assigned to the porters. However, since most of the V_U customers are served by porters in $TPRP_1$, the additional cost of solving these instances in $TPRP_3$ is less than 1% on average. Indeed, the higher the number of V_U customers served by porters in $TPRP_1$, the higher additional cost in $TPRP_2$ and the lower additional cost in $TPRP_3$. In real-life, delivery companies aim at maximising the profit and the presence of V_U customers is helpful as proved in this experiment.

4.4 Conclusions of the chapter

A VNS-based metaheuristic is designed and implemented to solve the TPRP. Three experiments are carried out in this chapter. In the first experiment, the algorithm was tested on the set of small-size instances created in Chapter 3. The algorithm was able to find optimal solution for all optimality-solved instances. For small-size instances that were not solved to optimality using our branch-and-cut algorithm (within the time limit) but known to be feasible, the VNS was able to improve their upper bounds (obtained by our exact algorithm) by 6.70% on average. In the second experiment, the algorithm was tested on the set of the MTVRP benchmark instances. Our heuristic performs competitively as it was able to find feasible solutions for most of the benchmark instances known to be feasible. In addition, it produced several new feasible solutions for benchmark instances. In the last experiment, a set of large-size instances is created and solved. Our experiments show that most of the unconstrained customers are visited by porters which confirm the benefit of having this type of customers instead of forcing them to be visited by the truck or by porters. There are many possible research directions to extend this work. The problem can be, for example, extended by considering porters with different carrying capacity, travel time duration, speed, or wage. Also, like any combinatorial optimisation problem, different heuristic algorithms can be designed to overcome our algorithm.

Chapter Five

TPRP with satellites

In this chapter, the truck-porters routing problem with satellites (TPRPS) is introduced. A variable neighborhood search (VNS) algorithm is designed and implemented to compute a good-quality solution for the TPRPS. An introduction is given in Section 5.1. Section 5.2 describes the problem in detail. The constructive heuristic for the TPRPS is presented in Section 5.3. The designed VNS algorithm is described in Section 5.4. Section 5.5 shows the computational experiments carried out in this chapter. Conclusions and possible future works are reported in Section 5.6.

5.1 Introduction

The growth of urbanisation and e-commerce sales in urban areas considerably increases causing environmental pollution and traffic congestion. Nowadays, more than 57% of the world's population lives in major cities, according to The World Bank (2022), and the process of urbanisation is foreseen to rise up further reaching 70% or more by 2050 (Bretzke, 2013). In addition, e-commerce sales grow rapidly and they were forecasted to grow by 56% over the next years (Statista, 2022). These factors worsen road traffic and burden the existing infrastructure in urban areas by increasing the number of delivery vehicles entering the city centres. Other factors aggravate the situation including the high purchasing power of the people living in urban areas and the increase of the number of trailers offering same-day delivery. Such factors rise the quantity and diversity of goods ordered and shipped to customers in urban areas and, therefore, inducing a much higher number of delivery vehicles to enter urban areas. Other challenges such as the reduction of road network capacity and the increase of road taxes in urban areas. These challenges causing a lack of available parking locations and, so, impacting the total cost of a product. According to Van Goor (1980), transportation costs often form a considerable part of the total cost of a product and represent up to 10% of the final price (Coyle et al., 1996).

Logistic activities related to delivering parcels to customers in urban areas are known as last-mile delivery operations. Last mile logistics is the least efficient stage of the supply chain that causes up to 28% of the total transportation costs (Rodrigue et al., 2016). The genuine willingness of retails to deliver parcels quickly and efficiently leads to existing last-mile delivery concepts such as parcel delivery by cargo bikes, drones, or delivery robots. Other last-mile delivery options have been promoted during the recent years like the combined transport of people and parcels in private transport such as taxis (Chen and Pan, 2016, Li et al., 2014) or public transport like buses, undergrounds, or trains (Ghilas et al., 2018, 2016). Boysen et al. (2021) discussed 27 distinct last-mile delivery concepts treated by existing research. In their paper, the authors discussed some future ideas for last-mile delivery concepts such as Amazon's patent for flying warehouses, i.e., airships circling over city centres from where drones are launched (Berg et al., 2016).

The interest of using electric cargo bikes (E-cargo bikes) for last-mile deliveries is rapidly growing (Narayanan and Antoniou, 2021). A new study finds that E-cargo bikes deliver about 60% faster than vans in city centres (Carrington, 2021). This because of the fact that E-cargo bikes can reach customers residing in restricted areas, e.g., pedestrian zones, and they can avoid dense car traffics which often occur in urban areas (Arnold et al., 2018). A comprehensive review consolidating the studies in the growing field of E-cargo bikes is introduced by Narayanan and Antoniou (2021).

The efficient use of cargo bikes in two-echelon distribution schemes can lead to successful applications. Anderluh et al. (2019) describe some successful applications in some European cities like, for example, Heavy Pedals and DPD in Vienna, Haijtas Pajtas in Budapest, and By-Expressen in Copenhagen. In 2017, the same authors develope a two-echelon city distribution scheme with temporal and spatial synchronisation between cargo bikes and vans in Anderluh et al. (2017). In their scheme, there is a van-depot and a bike-depot. From the van-depot, which is located on the outskirts of the city of Vienna, vans perform the delivery to the so-called *van-customers*, a type of customers that must be supplied by vans, and to the set of satellites (also called as micro-consolidation centres or micro-depots). In their paper, they used mobile satellites without storage facilities. So, a satellite is transshipment point where vans and cargo bikes can meet and it can be anywhere like, for example, in a car park. Similar to the vans, cargo bikes start and end their routes at the bike-depot located in the city centre.

The process of delivering parcels in urban areas can be efficiently organised by consolidating the transport requirements of different stakeholders and using environmentally friendly transport modes. Indeed, the main motivation for the distribution scheme developed by Anderluh et al. (2017) is to help logistics companies to design more efficient distribution plans with less costs and less emissions to serve customers located in the city centre of Vienna. They considered two companies chosen from two different sectors, namely pharmacy wholesale and distributors of vegetable boxes, and they were operating as follows: goods from different suppliers are consolidated at a depot on the outskirts of the city and then delivered by vans to customers. Most of customers are residing in inaccessible areas by vans and, hence, the need for more new sustainable distribution concepts.

In this chapter, we introduce a similar distribution concept using real-world data collected by a famous delivery company called Evri (previously known as Hermes). The provided data consists of a single depot located in Eastleigh and a set of satellites located in Romsey (both towns in England). It also contains information about 365 customers including their locations (most of them located in the centre of Romsey), and their parcels (size and weight). Our distribution concept is designed based on Evri's future plan to efficiently organise the distribution of parcels consolidated at a depot at the outskirts of the city and delivered to the customers located in urban areas using environmentally friendly transport modes.

To model such a situation, we introduce the truck-porters routing problem with satellites (TPRPS). In the TPRPS, parcels are stored in a depot located on the outskirts of the city. There is a set of satellites located on the outskirts of the city centre. Each satellite has a limited storage facility and it has a fixed location on the map and, so, it cannot be re-allocated. In this problem, a satellite can be a garage in a multi-story car park, the loading dock of a shop, or a trailer parked in a car park. In the TPRPS, there is a single truck (or van) and a limited number of porters. The task is to design an efficient delivery plan to deliver each parcel located in the depot by the truck either to the customer or to one of the satellites. Parcels delivered to the satellites by the truck must be then delivered to customers by porters.

In the TPRPS, the truck is assumed to be electric and each porter is assumed to be using an E-cargo bike for delivery. In this chapter, an efficient variable neighborhood search (VNS) algorithm is designed and implemented to find a good-quality solution for the problem. Our algorithm is designed to produce a good-quality solution for any TPRPS instance as well as it can be easily used by any logistic company who owns a fleet of E-cargo bikes, at least one electric truck, and has a usable set of satellites.

5.2 Problem description

The TPRPS is a generalisation of the truck-porters routing problem (TPRP) introduced earlier in this thesis. The TPRP is tackled by a branch-and-cut algorithm and an efficient VNS algorithm in Chapter 3 and Chapter 4 respectively. In the TPRP, there is a truck and a trailer available at a depot and loaded with parcels need to be delivered to customers located in an urban area. The trailer is attached to the truck before the truck departs from the depot towards a specific parking location (also called as a transshipment location) where the attached trailer is parked and a known number of porters arrived. The truck and the porters are then going to depart from the transshipment location to deliver all the parcels such that some parcels are delivered by the truck, some parcels are delivered by any porter, and the rest of parcels are delivered either by the truck or by porters. Once all parcels are delivered, the truck-driver has to return to transshipment location to re-attach the trailer and return to the depot.

In the TPRP, it was assumed that porters are identical and they must start and end their routes at the transshipment location. It was also assumed that the truck has sufficient capacity for a single trip. Whereas porters are limited by the amount of demand that they can carry and by the total time they can travel. However, porters can perform more than a single trip. In the TPRP, the transshipment location is referred to the depot. That is because of the assumption that visiting a customer is not permitted when the truck is pulling the trailer. In other words, the problem consists of one depot that both the truck and the porters depart and return to.

An extension of the TPRP is the TPRPS. In the TPRPS, there is a single depot and a set of satellites. Parcels are stored in a depot located on the outskirts of a city. The set of satellites is located on the outskirts of the city centre. In this problem, there is a single truck and a limited number of porters. The task is to design an efficient delivery plan to deliver each parcel located in the depot by the truck either to the customer or to one of the satellites. Any parcel delivered to a satellite must be then delivered to the customer by a porter. So, the total amount of demand delivered to the set of satellites by the truck is equivalent to the total demand delivered from the satellites to the customers by porters.

The distribution scheme, the TPRPS, we seek to introduce and solve consists of two levels. In the first level, some parcels will be delivered by the truck to the set of satellites. These parcels are then delivered by porters in the second level. The remaining parcels will be delivered by the truck on its way to supply some, or possibly all, satellites and before returning to the depot. An example of the TPRPS is given in Figure 5.1. Such a two-level distribution problem is often called a two-echelon distribution problem in the literature. Crainic et al. (2009) introduced the first formal definition of a famous class of two-echelon routing problems named two-echelon vehicle routing problems (2E-VRPs). A brief review of 2E-VRPs is given in Chapter 2. A recent review paper on the 2E-VRPs can be found in Sluijk et al. (2022).

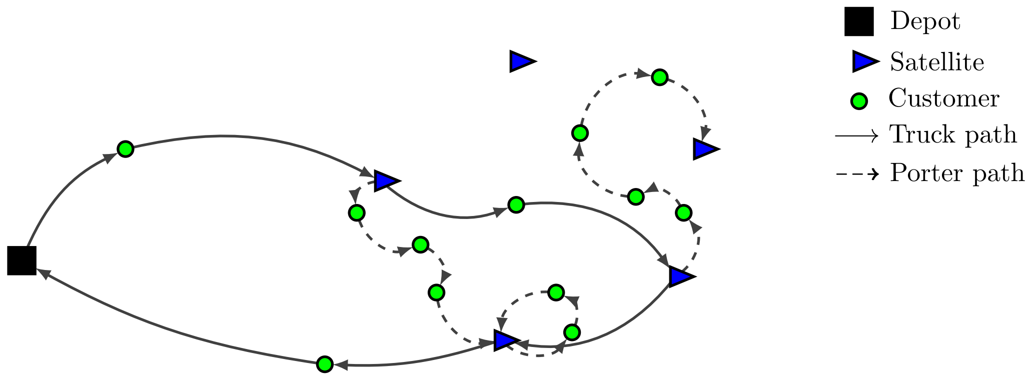

Figure 5.1: Illustration of the TPRPS with five satellites and twelve customers. Note that, only used satellites, where at least a porter departs from, are visited by the truck.

The 2E-VRP and the TPRPS aim at minimising the total delivery cost with the lowest possible number of delivery vehicles entering central cities by consolidating parcels in urban distribution centers (UDCs) located in cities outskirts (Savelsbergh and Van Woensel, 2016). In the 2E-VRP, customers are supplied from a UDC through satellites. First-echelon (FE) vehicles transport goods from the UDC to the satellites, from which second-echelon (SE) vehicles collect the goods and deliver them to the customers. So, customers are exclusively served by the fleet of SE vehicles in the 2E-VRP. In the TPRPS, however, this assumption is removed and, so, FE vehicles have the option of serving some customers. Taking an advantage of FE vehicles travelling from a satellite to another satellite or from a satellite to the depot (or vice versa) increases the efficiency of delivery plans.

The TPRPS can be formally defined as follows. Let $G = (V, A)$ be a directed graph, where $V = \{0, 1, ..., N\}$ is the set of vertices and $A = \{(i, j)|\ i, j \in V\}$ is the set of arcs. The set of vertices is partitioned into the depot $D = \{0\}$, the set of satellites $S = \{|D| + 1, ..., |D| + |S|\}$, and the set of customers $C = \{|D| + |S| + 1, ..., |D| + |S| + |C|\}$. Each arc $(i, j) \in A$ leads to a non-negative routing cost $\bar{c}_{ij}$ when traversed by the truck, and a non-negative routing cost c_{ij} when traversed by any porter. Each customer $i \in C$ requires a supply of $q_i = (q_i^v, q_i^w)$ and must be visited exactly once. The demand of satellite $s \in S$ equals to the sum of the demand of the customers served from s by porters. Satellites are assumed to be identical and, so, they have the same storage (size) capacity S^v. There is a single truck available at the depot, and m porters available at the needed satellites with $M = \{1, 2, .., m\}$. The truck is assumed to have sufficient capacity for a single trip (e.g., $Q_T^v = Q_T^w = \infty$). It is also assumed that porters are limited by the amount of demand (size Q_P^v and weight Q_P^w) that they can carry and by the total time they can travel T_P. However, porters can perform more than a single trip. We refer to a *trip* as a sequence of customer services preceded and followed by a visit to a satellite. We call a sequence of trips performed by the same porter a *route*.

In the TPRPS, the truck is assumed to be electric. In addition, every porter is assumed to be using an E-cargo bike for delivery. This means that each porter needs an E-cargo bike at the starting point (at a satellite). The objective of this problem is to minimise the total distribution cost under the following restrictions:

1. each customer must be visited exactly once;
2. if the truck or a porter visits a customer, it must depart from it;
3. the truck must start and end its route at the depot;
4. each porter can start from any satellite and return to the same or to a different satellite;
5. satellites storage (size) capacity S^v cannot be exceeded;
6. porters cannot travel more than T_P hour;
7. porters cannot exceed the carrying capacity (size Q_P^v and weight Q_P^w);
8. the maximum number of porters, m, can not be exceeded;
9. the truck must visit any satellite used by porters.

In this problem, we assumed that:

1. the truck-driver and the porters get paid the same amount of money equals to W_{cost} pounds per hour (we used the national minimum wage in 2022 in the UK (LPC, 2022));
2. the cost of electricity is fixed and its equal to E_{cost} pounds per kilowatt (we used the domestic electricity rate in the UK in 2022 (BEIS, 2022));
3. there is no cost associated with using any satellite;

4. any satellite may or may not be used;

5. E-cargo bikes are identical (have the same capacity and speed).

The parameters $\bar{c}_{ij}$ and c_{ij} represent the driving and cycling costs between i and j, where $i, j \in V$. In order to compute them, driving and cycling distances between i and j are needed. Let us denote the driving distance between i and j by $\bar{d}_{ij}$ and the cycling distance by d_{ij}. The time needed to travel from i to j can be computed using the following formula:

$$time = \frac{distance}{speed}. \tag{38}$$

The time of travelling from node i to node j by the truck and by any porter is denoted by $\bar{t}_{ij}$ and t_{ij} respectively. In this problem, however, $\bar{t}_{ij}$ and t_{ij} do not represent the actual time to go from i to j as there is an additional time called *service time*. The service time can be either a handling time or a loading/unloading time. As a result, the new travel time from i to j, which includes the service time, is denoted $\bar{t'}_{ij}$ and t'_{ij}. Additional times are reported in the computational experiments section, Section 5.5. To compute the cost of travelling from i to j by the truck and by porters, W_{cost}, E_{cost}, $\bar{t'}_{ij}$, and $\bar{d}_{ij}$ are needed. Costs of travel from i to j, where $i, j \in V$, can be expressed as:

$$\bar{c}_{ij} = W_{cost} \cdot \bar{t'}_{ij} + E_{cost} \cdot \bar{d}_{ij} \tag{39}$$

and

$$c_{ij} = W_{cost} \cdot t'_{ij} + E_{cost} \cdot d_{ij} \tag{40}$$

by the truck and by porters respectively. Thus, the objective function can be expressed as:

$$\min \quad \sum_{i \in V} \sum_{j \in V} \bar{c}_{ij} \bar{x}_{ij} + \sum_{k \in M} \sum_{i \in S \cup C} \sum_{j \in S \cup C} c_{ij} x_{ij}^k \tag{41}$$

where $\bar{x}_{ij}$ is a binary variable that takes value 1 if and only if the truck traverses arc $(i, j) \in A$, and x_{ij}^k is a binary variable that takes value 1 if and only if porter $k \in M$ traverses arc $(i, j) \in A$. A mathematical formulation for the TPRPS can be found in the appendix. It can be said that the TPRPS is a TPRP when there is only one satellite, the satellite location is exactly the same location as the depot, and the storage facility of the satellite is sufficiently large to store all the parcels. Therefore, the TPRPS is a generalisation of the TPRP and is thus it is $\mathcal{NP}$-hard.

5.3 Constructive heuristic for the TPRPS

In order to construct an initial solution for the TPRPS, a two phase algorithm is designed and implemented. In the first phase, a modified version of the nearest neighbour (NN) algorithm is applied to build porters routes. The NN algoithm is one of the first algorithms used to solve the traveling salesman problem (TSP) heuristically. The main idea of the NN algorithm is to always visit the nearest customer. Within time complexity of $O(n^2)$ where n is the number of customers, the algorithm normally finds fairly close route to the optimal route in the TSP (Karkory and Abudalmola, 2013). The process of our first-phase algorithm can be briefly described as follows:

Step 1: if inviting a new porter is possible, go to **Step 2**. Else, halt;

Step 2: find the closest customer i to a satellite S_k, where $k = 1, ..., k_{max}$ and k_{max} is the number of satellites. Go to **Step 4**;

Step 3: find the closest customer, i to the satellite *head*. Set $S_k = head$;

Step 4: if there is enough time for the porter to go from S_k to i, go to **Step 5**. Otherwise, go to **Step 1**.

Step 5: start a trip from S_k to i and set $tail = S_k$;

Step 6: find the nearest node to i. Call it j;

Step 7: find the nearest satellite to j. Call it S'_k;

Step 8: if going from i to j and then to S'_k is possible (that is $i \to j \to S'_k$), go to **Step 9**. Otherwise, close the current trip by going from j to *tail*, set $head = tail$, and go to **Step 3**;

Step 9: go from i to j, set $tail = S'_k$, set $i = j$, and go to **Step 6**.

Figure 5.2 illustrates the implemented NN algorithm with two satellites (blue triangles) and five customers (green circles). In this example, the porter starts from satellite S_1 to visit node 1. This means that

$$D_{S_1 \to 1} = \min\{D_{S_k \to i},\ \forall k \in S,\ i \in C\}$$

where S is the set of satellites and C is the set of customers. The trip continues to grow by visiting the nearest customer to the last visited customer in the trip until either all customers are served or visiting a customer is not possible. In the latter case, the porter must return to the nearest satellite to

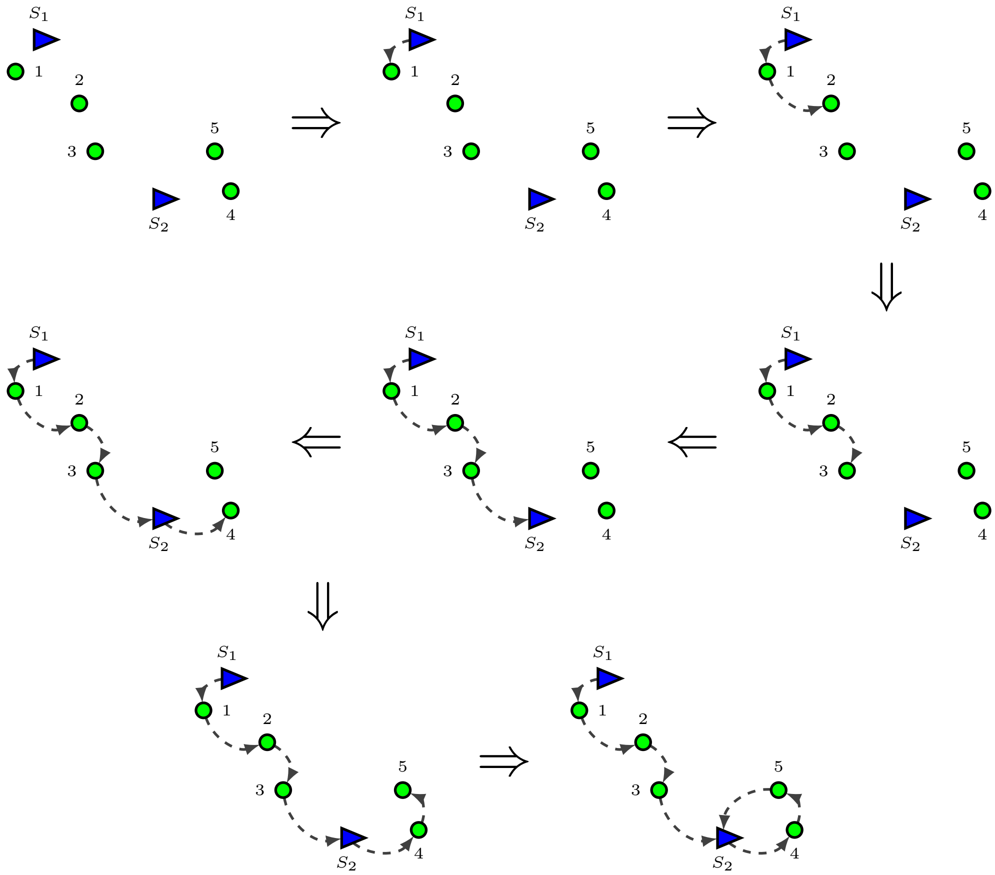

Figure 5.2: Illustration of the implemented nearest neighbour algorithm. In these figuers, blue triangles are satellites and green circles are customers. Starting from the top-left figure, the closest node to a satellite is node 1, so $i = 1$ and $S_k = S_1$. A porter trip starts from S_1 to node 1, 2, 3, and then to S_2 ($tail = S_2$). At this moment, the porter is able to depart from S_2 to visit the node 4. Thus, the porter visits node 4 from S_2. From node 4, the porter is able to visit node 5 before returning to the nearest satellite, S_2, without violating capacity restrictions or exceeding the travel time. The porter ends up taking the following route $S_1 \to 1 \to 2 \to 3 \to S_2 \to 4 \to 5 \to S_2$.

finish the current trip and to start, if possible, a new trip. Otherwise a new porter, who can starts from any satellite, is invited to make a new route. The process is repeated until either all customers are served or adding a new porter is not possible. At this moment, the second phase of the designed algorithm is started. In this phase, the truck departs from the depot, visits the set of used satellites and the set of the remaining customers, and then returns to the depot. In another words, we are solving the famous TSP presented by Flood (1956). One of the most successful heuristic algorithms for the TSP is the Lin-Kernighan heuristic (LKH) algorithm (Lin and Kernighan, 1973). The LKH algorithm uses k-opt moves to optimise the solutions. The k-opt moves explores the solution space by replacing k edges of the current trip, where k is any integer greater than or equal to 2 and less than the number of nodes. Therefore, the LKH algorithm is called to solve this problem.

5.4 Variable neighborhood search algorithm for the TPRPS

5.4.1 An overview

The VNS algorithm starts with an initial solution created by the constructive heuristic explained in the previous section. At the beginning, our algorithm attempts to improve the current solution by applying *local search operators*, also called *neighborhoods*, to find the best neighboring solution. Local search operators are applied sequentially in a deterministic way. Such a method is known as the variable neighborhood descent (VND) algorithm. The VND algorithm stops when there is no improvement with respect to any of the considered neighborhoods. Within the VND algorithm, there are different types of search procedures such as pipe, cyclic, and union VND. In our algorithm, we use the pipe VND procedure where the search in a neighborhood stops when there is no improvement, otherwise it continues the search in the same neighborhood. Algorithm 5.1 shows the proposed VND algorithm. We refer the reader to Gendreau et al. (2010) for more details about VND variants.

Algorithm 5.1 : The VND algorithm used within the VNS algorithm

define the set of local search operators R_k, for $k = 1, ..., k_{\max}$. Set $k = 1$.
while $k \leq k_{max}$ **do**
    repeat
        find and then apply the best move in R_k.
    until R_k fails to find an improvement.
    call the current local optima of R_k as x'.
    if $f(x') < f(x)$ **then**
        set $x = x'$ and $k = 1$.
    else
        set $k = k + 1$.
    end if
end while

The improved solution, x, obtained by the sequential VND is a local optimum with respect to all neighborhoods and, therefore, set as the incumbent solution, $x_{incumbent} = x$. The algorithm is then repeated within the following procedure, which consists of three steps, until the pre-set time limit is reached. The first step is the *perturbation step*, also called *shaking step*. In the perturbation step, we construct a new solution, x', generated from shaking the incumbent solution using one of the *shaking procedures*.

Algorithm 5.2 : The VNS algorithm for the TPRPS

define the set of shaking procedures N^k, for $k = 1, ..., k_{\max}$ and the set of local search operators R_l, for $l = 1, ..., l_{\max}$.
apply the constructive heuristic to create an initial solution x.
apply the VND algorithm, Algorithm 5.1, to improve x.
repeat
  set $k = 1$.
  repeat the following steps:
    Shaking: Generate a solution x' at random from the k^{th} shaking procedure of $x(x' \in N_k(x))$.
    Local search: Apply the VND algorithm with x' as initial solution to find the best neighboring solution x''.
    Move: If x'' yields a better quality solution, **then** set $x = x''$, $iter = 1$, $k = 1$, otherwise set $k = k + 1$.
  until $k = k_{\max}$.
until the pre-set time limit is reached.

The aim of the perturbation step is to escape from local optima as it allows diversification in the search space. The second step is known as the *improvement step*. In the improvement step, the VND algorithm is applied to find the best neighboring solution, x''. Finally, at the third step, we compare the cost obtained in step two, $f(x'')$, with the cost of the incumbent solution, $f(x_{incumbent})$. If $f(x'') < f(x_{incumbent})$ we set $x_{incumbent} = x''$, otherwise the algorithm starts the next iteration. Algorithm 5.2 describes the proposed VNS algorithm. Shaking and local search procedures used within our VNS algorithm are described in details in the following section.

5.4.2 Explanation of the main steps

Shaking procedures

The VNS escapes from local optima by applying shaking procedures to the current local minimum. The first step of the VNS is to create a neighboring solution to the current local minimum by one of the shaking procedures. There are three shaking procedures used in our VNS algorithm, $k_{max} = 3$, the *remove-insert procedure*, the *satellite eliminator procedure*, and the *route initiation procedure*. These shaking procedures are ordered as follows: the remove-insert procedure is used as N_1, the satellite eliminator procedure is used as N_2, and the route initiation procedure is used as N_3. These procedures are described as the following:

N_1: the remove-insert procedure is aimed at generating a feasible solution by removing and, then, re-inserting some customers. The procedure works as follows:

1. choose a number of m customers randomly, and list them in L;
2. remove chosen customers from their current positions in the trips;
3. if $L \neq \emptyset$, choose the first customer in L, call it i, otherwise halt;
4. find the best three feasible insertion positions of i into existing trips;
5. insert i at one of the best three insertion positions;
6. remove i from L, $L = L \backslash \{i\}$, and go to (3).

In the removal phase of this procedure, the chance of selecting a customer increases when the parcel-size of the customer increases. In addition, in the insertion phase of this shaking procedure, a customer might be inserted at the first, second, or third best feasible position in step (5). The probability of choosing an insertion position depends on the cost of insertion. The position with less additional cost is more likely to be chosen. Therefore, the cheapest insertion position is always have the highest chance to be selected.

N_2: the satellite eliminator procedure aims at generating a feasible solution by removing a satellite that is used in $x_{incumbent}$. Letting m be the number of porters in $x_{incumbent}$, the procedure works as follows:

1. pick a used satellite randomly, s', and remove it from the truck route;
2. remove all the trips starting from s' and list their nodes in L;
3. if the number of porters is greater than or equal to m, go to (6);
4. choose a satellite s'' randomly such that $s'' \neq s'$;
5. apply the NN algorithm to build a new route by a new porter starting from s'', and go to (3);
6. if $L \neq \emptyset$, extend current porters' routes by starting new trips to serve some of the remaining customers if possible;
7. insert the remaining nodes into existing routes.

In this shaking procedure, any satellite can be eliminated. In step (4), however, the chance of selecting a satellite is increased when the demand of the satellite decreasing. The demand of a satellite is equal to the sum of customers' demand served from it by porters. In step (5) and step (6), the NN algorithm is working slightly differently than the NN algorithm explained earlier in Section 5.3. In the current NN algorithm, visiting the nearest customer by porter is optional as the porter has the chance to visit the first, second, or third nearest customer. The probability of choosing the next customer depends on the cost of visiting that customer. The customer with less cost is more likely to be chosen. In the last step, step (7), we follow the same strategy used in N_1 to insert customers into exiting routes.

N_3: the route initiation procedure is aiming at generating a feasible solution by inviting one more porter, if possible, in addition to the set of porters that is already involved in the incumbent solution. The procedure starts by randomly selecting a satellite. The chance of choosing a satellite increases when the total amount of parcels (size) stored at the satellite decreases. The new porter departs from the selected satellite to serve some customers which are part of other porters routes. In other words, the new porter is going to steal some customers from other routes in order to build their own route. We use a modified version of the NN algorithm used in our constructive heuristic described in Section 5.3 to build this route. In this procedure, instead of always visiting the closet customer, the new porter can go to the first, second, or third closest customer. The probability of choosing a customer depends on the cost of visiting that customer. The closer customer is more likely to be selected.

Local search operators

Local search operators are applied once the current local minimum is perturbed by one of the shaking procedures. The aim of applying these neighborhoods is to improve the perturbed solution. In our algorithm, the pipe VND procedure is used to find the local minimum with respect to all neighborhoods described in this section. In the pipe VND, neighborhoods are explored in a deterministic way such that the search in a neighborhood stops when there is no improvement has been found, otherwise it continues the search in the same neighborhood. The procedure stops when there is no improvement with respect to any of the considered neighborhoods. Thus, the solution obtained by our VND algorithm is a local optimum with respect to all neighborhoods. Algorithm 5.1 presented in Subsection 5.4.1 illustrates the designed VND algorithm. There are three local search operators have been used inside our VND algorithm, the *1-insertion procedure*, the *swap procedure*, and the *2-opt intra-trip* procedure. These neighborhoods can be briefly described as follows:

The 1-insertion procedure: This neighborhood tries to reduce the total cost of the current solution by removing a customer from its position in a trip and insert it elsewhere while maintaining feasibility. The removed customer can be inserted into a different position in the same trip, into another trip within the same route, or into another trip in a different route. The procedure is repeated and the one that yields the largest improvement is selected.

The swap procedure: In this neighborhood, we look for an improved solution by swapping a pair of customers. The two customers are selected, removed from their trips, and then inserted such that the first customer is inserted at the best insertion position of the second customer's trip and the second customer is inserted at the best insertion position of the first customer's trip, if feasible. The two customers may belong to the same trip, the same route, or different routes. The move is repeated and the one that yields the largest improvement is chosen.

The 2-opt intra-trip procedure: This neighborhood aims at reducing the total cost by selecting, removing, and replacing two non-adjacent arcs by other two arcs within the trip. The 2-opt intra-trip procedure is the 2-opt algorithm proposed by Croes (1958). It works by replacing two arcs with other two new arcs and reverting the direction of one of the resulting two sub-paths. This neighborhood is applied to each trip individually. In this procedure, we always select the move that produces the largest improvement to the considered trip.

In our implementation, neighborhoods are placed in a list with a given order and always explored in that order. In our VND algorithm, neighborhoods are explored in the following order: the 1-insertion procedure as R_1, the swap procedure as R_2, and the 2-opt intra-trip procedure as R_3.

5.5 Computational experiments

In our computational experiments we use the designed VNS algorithm explained in Section 5.4 on a real-world instance. The instance consists of one depot (located in Eastleigh, a small town in England), five satellites (located on the outskirts of the city centre of Romsey, another small town in Englan), and 365 customers (distributed across Romsey) as described earlier in the description section, Section 5.2. To solve this instance, driving and cycling distances between each pair of nodes are computed using the Python toolkit OSMnx (version 1.2.3) developed by Geoff Boeing (Boeing, 2017). The free open-source OSMnx package interacts with OpenStreetMap APIs to calculate the shortest path for any pair of nodes beside other things such as retrieving, constructing, analysing, and visualising street networks. We use OSMnx to calculate the driving and cycling distance matrices for the truck and the porters respectively.

Table 5.1: Assumed values for the Romsey instance.

description	value (unit)	
allowed porter carrying capacity (weight)	150	kg
the storage capacity of a satellite (size)	3000	litre
allowed porter carrying capacity (size)	1500	litre
allowed porter travel time duration (time)	4.50	hours
time to serve a customer by porters (time)	3	minutes
time for loading at a satellite by porters (time)	6	minutes
time to serve a customer by the truck (time)	5	minutes
time for unloading at a satellite by the truck (time)	15	minutes
the speed of the truck (mile per hour)	25	mph
the speed of porters (mile per hour)	10	mph
the wage cost of the truck-diver and porters (per hour)	9.50	£
the electricity cost when travel by the truck (per mile)	0.57	£
the electricity cost when travel by a porter (per mile)	0.12	£

Our VNS-based algorithm was coded in C++, compiled with Visual Studio 2017 and run on the IRIDIS 5.0 High Performance Computing Facility of the University of Southampton, relying on a cluster of compute nodes equipped with dual Inter(R) Xeon(R) Gold 6130 CPUs @ 2.10GHz and 192 GB of DDR2 RAM using a single thread per experiment. Table 5.1 contains information about assumed values for the Romsey instance. Every row in the table contains an assumed value described in the first column, column *description*, and given in column *value*. At any experiment carried out in this section, the algorithm terminates when a pre-set time limit is reached. The time limit to solve the current problem is set to be equal to one hour. In addition, all instances are solved five times and the average value for each instance is reported in all tables. Table 5.2 provides the column headings for tables presented in this section. This section is organised as follows. An effective configuration of the designed VNS algorithm is presented in Subsection 5.5.1. An evaluation of the proposed distribution concept, the TPRPS, is carried out in Subsection 5.5.2. The impact of changing satellites storage capacity is tested in Subsection 5.5.3. In Subsection 5.5.4 and 5.5.5, various carrying capacities and speeds for porters are experimented. Finally, the impact of changing the electricity cost on the total distribution cost is tested in Subsection 5.5.6.

Table 5.2: Column headings for next tables.

column heading	description
S_{cap}	storage capacity of each satellite
P_{num}	the number of invited porters
$\|trip\|$	the total number of porters' trips
total cost	the total cost of the solution including the truck cost
T_{nodes}	the number of customers visited by the truck
cost	the average porter trips' costs
time	the average working time for each porter trip
length	the average distance travelled in each trip
weight	the average weight of items carried by a porter in a single trip
size	the average size of items carried by a porter in a single trip
P_{cap}	the maximum size of items a porter can carry in a single trip
P_{speed}	the speed of porters (mph)
T_{cost}	the charging cost of the truck per mile
B_{cost}	the charging cost of E-cargo bikes per mile

5.5.1 Algorithm configuration

Designing an efficient heuristic to compute a high-quality solution for this complex problem is challenging. In many traveling salesman problem (TSP) and vehicle routing problems (VRP)s efficient heuristics, such as the Lin and Kernighan (1973) heuristic, only promising moves are considered. Such approach is used to reduce the computational complexity of the problem and it is known as *heuristic pruning*. Heuristic pruning attempts to reduce the size of the explored neighborhoods by only considering promising moves. The more restricted the pruning strategy is, the faster the neighborhoods. However, as a result, neighborhoods explore less moves which might prevent them from finding improvements. So, there is a trade-off between runtime and solution quality. A conmon pruning strategy is to only consider those moves that involve short edges (Toth and Vigo, 2003). This means that a node can only be connected to its nearest C nodes, where $0 \leq C \leq n$ and n is the number of customers. Arnold et al. (2019) find that the best results of solving VRP instances with up 1000 customers are obtained with $C = 30$. After some experiments with different values of C, $C = \{10, 20, 30, 40, 50\}$, we observed that the best results with $C = 30$ and, thus, we decide to use heuristic pruning with $C = 30$ in our VNS algorithm.

In the improvement phase of our VNS algorithm, neighborhoods are applied within the pipe VND algorithm described in Section 5.4. One of the most popular and effective strategies to explore a neighborhood is the best improvement, also called the *steepest descent*. In this strategy, the associated neighborhood is completely explored by a fully deterministic procedure, performing the best associated move (Gendreau et al., 2010). Such a strategy has a higher computational complexity than other strategies like, for example, the first improvement. However, the best improvement method leads to better results for our problem as our preliminary experiments confirm and, thus, it has been adapted in our pipe VND algorithm.

The use of heuristic pruning fastens the process of exploring neighborhoods. However, with the use of the best improvement approach within our pipe VND algorithm, the VND algorithm takes long time to be executed. Thanks to the 1-insertion and swap procedures which consist of intra-trip, intra-route, and inter-route procedures. To make our VND algorithm more effective, as we observed, we adopt a different way to apply the VND procedure. In our VND algorithm, we only consider the 2-opt intra-trip procedure until no improvement has been detected for a specific number of iterations, I. Once the VNS fails to find any improvement for I iterations in a row, other neighborhoods, namely the 1-insertion and the swap procedures, are included in the VND until either an improvement is detected or the pre-determined time limit is reached. In the former case, we repeat the process until the time limit is reached. This method with $I = 500$ leads to the best results for this problem and, therefore, it was adopted in our metaheuristic algorithm.

In any VNS algorithm, shaking procedures are called according to their orders. However, we have adopted a different approach in which the shaking procedures are called in our VNS algorithm. In our approach, the chance of selecting a procedure depends on the number of times the shaking procedure, followed by the improvement procedure, leads to a new incumbent solution. We observed that the first shaking procedure was able to lead to a new incumbent solution most of the times and, so, it has the highest chance to be selected. Therefore, we set the chance of selecting the first, second, and third shaking procedures as 70%, 15%, and 15% respectively.

The adopted shaking procedures have been tested to ensure the intensification and diversification of the search in our VNS algorithm. In the first shaking procedure, we decided to remove and then insert only 10% of the customers. Shaking procedures are explained in details in Subsection 5.4.2.

The quality of the proposed VNS algorithm can be measured by comparing the cost of the initial solution constructed by the constructive heuristic and the cost of the solution obtained by the VNS. Our VNS algorithm is capable of improving the initial solution by more than 39% in some cases such as when the carrying capacity of porters is small. The smaller carrying capacity of porters, the more trips to be performed and, therefore, the harder to obtain a good-quality solution by the designed constructive heuristic.

5.5.2 Concept evaluation

To evaluate the proposed distribution concept, a comparison between the real operating cost conducted by Evri and the approximate cost computed by our VNS algorithm to deliver all the parcels in the current instance. However, the real operating cost was not shared with us. In addition, we have no information about the method used nor the number of vehicles that were used in the delivery operation at that day. Large delivery companies often use a fleet of vehicles to deliver their parcels. This problem is known as the vehicle routing problems (VRP) which was introduced by Dantzig and Ramser (1959). In the VRP, the goal is to optimally design routes for multiple vehicles to visit a set of customers such that each customer must be visited exactly once and the set of vehicles must start and end their routes at a depot. Important VRP variants are reviewed in Chapter 2. The latest taxonomic review of VRP literature can be found in Braekers et al. (2016).

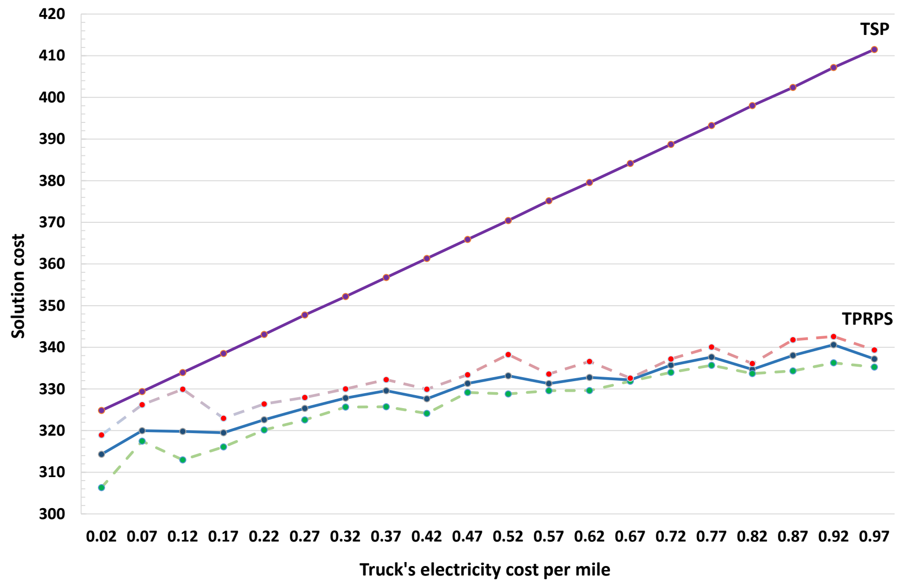

Figure 5.3: Illustration of the comparison between the TSP (purple line) and the TPRPS (red dots are the highest costs, green dots are the lowest costs, and the blue dots refer to the average cost of five runs).

To overcome this problem, we compare the cost of our distribution concept with the lowest possible, but unrealistic, method to deliver the 365 parcels in the current instance. Such a method is known as the traveling salesman problem (TSP) in the literature. In the TSP, a salesman is required to travel between a number of cities with the lowest possible cost.

The salesman requires to visit every city exactly once and returns to the starting point. The TSP was first studied as a mathematical problem in the 1930s by Karl Menger in Vienna (Applegate et al., 2011). In our case, solving the TSP means that the truck must start and end its route at a depot such that each customer must be visited exactly once. Note that, in the current problem, satellites must not be visited by the truck as there is no need to visit them. The aim of solving the TSP of the current problem is to compute the lowest possible cost to deliver all the parcels in the current instance. In this chapter, as it was mentioned, the cost, c, of the shortest path between each pair of nodes is computed and used to compute the cost function. This means that for all vertices $a, b, c \in V$ we have:

$$c(a,\ c) \leq c(a,\ b) + c(b,\ c) \tag{42}$$

and, therefore, the cost function satisfies the triangle inequality. Haimovich and Rinnooy Kan (1985) introduces the following lemma:

Lemma 1. *The cost c(TSP) of an optimal tour is a lower bound on the cost of the VRP.*

Proof. Given an optimal solution of the VRP, merging all sub-tours into a single tour is possible and it leads to lesser or equal cost to the cost of the VRP because of the triangle inequality. □

From Lemma 1, we can say that the cost of delivering the 365 parcels using a fleet of vehicles, as most of delivery companies do, is at least equal to the cost of solving the TSP of this problem, hence the reason of comparing the cost of solving the TSP with the cost of solving the TPRPS. In this comparison, on the one hand, we used the LKH algorithm to solve the TSP (Lin and Kernighan, 1973). On the other hand, we used our VNS algorithm to solve the TPRPS. One of the key elements to compute the cost of travel between two vertices is the electricity cost. The electricity cost of travelling from i to j, where $i, j \in V$, depends on the distance between i and j. In this experiment, we used different values of electricity cost. Figure 5.3 contains the results of solving both problems with different values of electricity cost. The x-axis represents the values of the truck's electricity costs whereas the y-axis indicates the total solution cost. Clearly, the growth of the cost of the TSP is much faster than the growth of the cost of the TPRPS. This means that the proposed distribution concept is not only reducing the total distribution cost, but it also a sustainable distribution concept that can be applied in any city or town.

5.5.3 Satellites storage capacity

As it was assumed that satellites are identical and limited in size. For the aim of finding a reasonable satellites capacity that maximise the profit the most, we carried out this experiment. We tried ten different storage capacities while maintaining other values assumed in Table 5.1. Table 5.3 shows the ten different values in the first column, column S_{cap}. In this table, the size of

Table 5.3: The result of changing satellites storage capacity. See Table 5.2 for column headings description.

					average porter trip				
S_{cap}	P_{num}	$\|trip\|$	*total cost*	T_{nodes}	*cost*	*time*	*length*	*weight*	*size*
1500	8.40	10.40	353.18	12.80	30.05	3.00	12.56	64.17	689.36
2000	8.40	8.80	340.96	8.00	35.36	3.54	14.44	78.12	848.11
2500	8.00	8.00	332.17	5.80	37.90	3.80	15.02	85.98	931.37
3000	8.00	8.00	332.44	7.20	37.84	3.79	15.05	85.52	921.11
3500	8.20	8.20	333.34	5.40	37.31	3.74	14.89	83.95	912.98
4000	8.00	8.20	333.52	5.80	37.37	3.74	14.97	83.73	903.00
4500	8.00	8.00	333.53	6.20	38.10	3.82	15.23	85.67	927.88
5000	8.20	8.20	332.43	6.60	36.84	3.69	14.52	84.04	910.07
5500	8.00	8.20	331.34	6.00	36.89	3.70	14.53	83.87	909.29
6000	8.20	8.20	330.09	5.60	36.81	3.69	14.44	83.78	910.20

satellites starts from 1.5 to 6 cubic meters. As it can be seen from the table that the smaller the satellite storage capacity, the higher number of customers to be served by the truck and, thus, the higher the cost of delivery. It can be argued that a good storage capacity of satellites, at least for the current instance, can be anywhere between 2.5 and 6 cubic meters. Obviously, the larger the storage capacity, the less delivery cost.

5.5.4 Porters carrying capacity

In the TPRPS, porters are limited by the total demand (weight and size) that they can carry and by a total working time constraint. However, a porter can re-visit any satellites to collect further items for delivery. In the real world, the size of parcels usually causes a problem. Indeed, retail companies usually pack small items in large cardboard boxes (see, e.g., an article about Amazon packaging written by Statz, 2018). Such action yields to carry large size, but small weight, items. Therefore, different carrying capacities (size) are tested. The aim of this experiment is to measure the influence of the

Table 5.4: The result of changing porters carrying capacity. See Table 5.2 for column headings description.

					average porter trip				
P_{cap}	P_{num}	$\|trip\|$	*total cost*	T_{nodes}	*cost*	*time*	*length*	*weight*	*size*
100	11.00	37.80	431.12	103.00	7.59	0.75	3.56	9.58	89.68
250	10.80	30.00	412.43	24.40	12.10	1.20	5.82	20.41	216.70
500	10.00	18.40	383.90	7.20	19.19	1.91	8.84	37.12	401.18
750	8.00	12.80	352.65	11.20	24.78	2.48	10.42	52.34	564.11
1000	8.00	10.40	337.18	6.80	29.39	2.95	11.69	65.82	714.17
1250	8.00	9.40	334.91	7.20	32.57	3.26	13.03	73.23	792.05
1500	8.20	8.20	332.70	6.00	37.20	3.73	14.82	84.12	912.55
1750	8.20	8.20	331.38	5.00	37.37	3.74	14.92	84.08	913.46
2000	8.00	8.20	330.45	5.20	37.15	3.72	14.73	83.97	910.35

capacity on the total working time of porters and so the total cost of this problem. Table 5.4 shows that the more carrying size capacity of a porter, the more customers to be served by porters and, thus, the smaller the cost of delivery. To maximise the benefit gained from E-cargo bikes and, so, the profit, it can be suggested that E-cargo bikes volume should be larger than or equal to one cubic meter. The larger the size of E-cargo bikes is, the smaller the delivery cost.

5.5.5 Porters speed

Cycling can be faster than driving in some city centres. For example, the average driving speed in central London is between 7.1 – 8.7 mph (TFL, 2013). Whereas, the average cycling speed in central London is 13.98 mph according to one of the largest tracking physical exercise company called Strava (Woodman, 2015). The current data, however, is taken from a town in England and, so, we cannot use the average speeds of central London. So, we set the speed of driving according to the average driving speed on all roads in Great Britain in 2014 (Statista, 2015). In addition, the speed of E-cargo bikes is set to be equal to the average speed of thirty different cycle courier operators across Europe (McLeod et al., 2020). The default cycling and driving speeds are set to be equal to 10 mph and 25 mph respectively as presented in Table 5.5. In this experiment, the speed of driving remains constant. However, the speed of porters driving E-cargo bikes is changing. Starting from 3 mph to 15 mph, thirteen different porter speeds are considered. This experiment

Table 5.5: The result of changing porters speed. See Table 5.2 for column headings description.

					average porter trip				
P_{speed}	P_{num}	$\|trip\|$	*total cost*	T_{nodes}	*cost*	*time*	*length*	*weight*	*size*
3	10.40	10.40	467.28	131.60	27.82	2.86	5.08	43.62	460.83
4	10.80	11.00	442.05	77.00	29.53	3.02	6.64	51.45	540.83
5	10.80	10.80	434.92	39.20	33.34	3.39	9.16	57.65	624.31
6	10.00	10.00	411.28	16.20	36.93	3.74	11.67	66.46	718.72
7	9.00	9.00	378.82	11.60	37.79	3.82	12.62	74.83	811.72
8	9.00	9.00	369.23	7.00	37.49	3.77	13.84	76.01	822.01
9	8.00	8.00	344.59	6.80	39.47	3.96	15.06	85.97	925.64
10	8.00	8.00	332.90	6.00	38.13	3.82	15.25	85.89	927.35
11	7.40	7.40	321.39	6.20	40.06	4.01	16.70	92.17	1000.53
12	7.00	7.00	314.46	5.40	41.16	4.11	17.83	97.86	1062.66
13	7.00	8.00	305.60	4.80	35.05	3.49	15.47	86.18	934.54
14	7.00	8.00	297.01	4.40	34.11	3.40	15.28	86.21	932.87
15	7.00	7.00	288.27	4.40	37.95	3.78	17.22	98.53	1063.74

confirms the importance of having porters even if the speed of E-cargo bikes is way less than what it is originally assumed.

5.5.6 Electricity cost

Electricity cost, or charging cost, has a significant impact on the solution of this problem since both the truck and bikes are electric. This experiment aims at measuring the impact of changing the charging cost of the truck only, E-cargo bikes only, and both the truck and E-cargo bikes. The electricity cost depends on many factors including the battery size and the cost of electricity. Nowadays, the domestic electricity rate in the UK is about 0.34 pound per kWh (BEIS, 2022). We assumed that the electric truck is consuming 420 kWh to do 250 miles, which means that it costs 0.57 pound per mile. It was, also, assumed that E-cargo bikes are costing 0.12 pound per mile (Brown, 2013). In Table 5.6, the cost of charging the truck is variable whereas the cost of charging E-cargo bikes is constant. Table 5.6 shows that the higher charging cost of the truck, the smaller the number of nodes to be visited by the truck. It also shows the need of the porters as they, in all cases, were responsible to deliver most of the parcels. The other way round where the cost of charging the truck is constant and the cost of charging E-cargo bikes is variable confirms the importance of the porters as shown in Table

Table 5.6: The result of changing the electricity charging cost for the truck. See Table 5.2 for column headings description.

					average porter trip				
T_{cost}	P_{num}	$\|trip\|$	*total cost*	T_{nodes}	*cost*	*time*	*length*	*weight*	*size*
0.03	8.00	8.00	317.22	16.00	35.70	3.59	13.53	83.39	895.58
0.12	8.20	8.20	319.37	16.80	34.86	3.50	13.22	81.21	871.58
0.21	8.20	8.20	324.49	7.80	36.57	3.67	14.33	83.39	905.88
0.30	8.20	8.20	325.69	10.00	36.20	3.63	14.11	83.10	898.45
0.39	8.20	8.20	330.98	7.20	37.05	3.71	14.74	83.45	906.44
0.48	8.20	8.40	334.36	6.20	36.59	3.67	14.71	81.82	890.38
0.57	8.00	8.00	337.28	5.20	38.81	3.88	15.84	85.92	933.13
0.66	8.00	8.00	335.07	5.00	38.42	3.85	15.47	85.83	933.25
0.75	8.00	8.00	336.67	5.20	38.50	3.86	15.55	85.90	930.27
0.84	8.00	8.00	338.60	5.20	38.52	3.86	15.57	86.02	930.39
0.93	8.20	8.20	338.92	5.00	37.56	3.76	15.11	84.11	911.21

Table 5.7: The result of changing the electricity charging cost for the E-cargo bikes. See Table 5.2 for column headings description.

					average porter trip				
B_{cost}	P_{num}	$\|trip\|$	*total cost*	T_{nodes}	*cost*	*time*	*length*	*weight*	*size*
0.03	8.00	8.00	322.03	5.20	37.04	3.85	15.48	86.10	932.61
0.12	8.20	8.20	332.42	6.00	37.22	3.73	14.84	84.02	913.48
0.21	8.20	8.20	342.95	7.40	37.96	3.68	14.40	83.53	906.57
0.30	8.00	8.00	353.92	7.80	40.09	3.76	14.71	85.61	928.03
0.39	8.00	8.00	364.19	10.00	41.24	3.74	14.67	84.63	920.50
0.48	8.00	8.00	371.31	8.20	42.14	3.71	14.30	85.04	920.70
0.57	8.00	8.00	384.94	10.20	43.68	3.72	14.55	84.53	912.28
0.66	8.00	8.00	394.21	14.80	44.00	3.65	14.10	83.48	897.57
0.75	8.00	8.00	405.73	15.20	45.54	3.67	14.27	83.35	899.47
0.84	8.00	8.00	414.53	17.60	46.37	3.63	14.10	82.38	892.69
0.93	8.00	8.00	418.98	26.20	45.01	3.47	12.97	81.52	882.12

5.7. Indeed, the majority of parcels were delivered from satellites by porters in Table 5.6 and Table 5.7. However, the impact of the latter case, when the cost of charging the E-cargo bikes is variable, on the total delivery cost is more significant. This means that, the charging cost of E-cargo bikes is more important than the cost of charging the truck. In the last part of this experiment, the cost of charging the truck and the cost of charging E-cargo

Table 5.8: The result of changing the electricity charging cost for the truck and E-cargo bikes. See Table 5.2 for column headings description.

						average porter trip				
T_{cost}	B_{cost}	P_{num}	$\|trip\|$	*total cost*	T_{nodes}	*cost*	*time*	*length*	*weight*	*size*
0.03	0.93	8.00	8.00	360.89	85.40	32.83	2.64	8.37	67.41	708.23
0.12	0.84	7.80	7.80	383.11	54.40	39.71	3.18	11.33	76.84	816.36
0.21	0.75	8.20	8.20	377.64	45.20	38.06	3.12	11.17	74.09	790.02
0.30	0.66	8.20	8.20	381.59	27.20	40.02	3.35	12.37	78.41	849.15
0.39	0.57	8.00	8.00	377.74	21.80	41.06	3.52	13.28	81.53	881.93
0.48	0.48	8.20	8.40	369.71	12.80	39.64	3.49	13.40	80.66	870.62
0.57	0.39	8.00	8.00	365.15	9.60	41.26	3.74	14.67	84.72	914.68
0.66	0.3	8.00	8.00	354.31	6.60	40.24	3.77	14.76	85.67	930.59
0.75	0.21	8.00	8.00	346.51	5.20	39.50	3.82	15.20	86.10	933.77
0.84	0.12	8.00	8.00	337.94	5.40	38.39	3.85	15.46	85.46	925.07
0.93	0.03	8.00	8.00	326.22	5.20	36.81	3.83	15.26	85.80	930.85

bikes are variable. Table 5.8 contains the costs of charging the truck and E-cargo bikes in the first and the second column, column T_{cost} and B_{cost}, respectively. In the first row of Table 5.8, the charging cost of the truck is the lowest while the charging cost of the E-cargo bikes is the highest. As a result, the truck was delivering most of the parcels.

5.6 Conclusions of the chapter

The TPRPS is an innovative two-echelon distribution concept that combines driving and cycling to deliver parcels in urban areas. The concept was designed based on a genuine desire of a well-known delivery company to find an applicable, profitable, and eco-friendly idea to deliver parcels consolidated at a depot located at the outskirts of a city to the customers living in the city centre with the minimum operating cost and the lowest possible harm to the environment. The aim of the TPRPS is to design an efficient delivery plan to deliver every parcel consolidated at the depot by the truck either directly to the customer or to one of the satellites, also called micro-depots, located at the outskirts of the city centre. Parcels delivered by the truck to the satellites are delivered to customers by porters, using E-cargo bikes, whom can help to quickly reach customers residing in urban areas with access restriction, e.g., pedestrian zones, and they can avoid dense car traffics which often occur in urban areas (Arnold et al., 2018).

An effective VNS algorithm is designed and implemented to tackle this problem in reasonable computation times. In our computational experiments, we use a real-world instance provided by Evri. A comparison between the proposed distribution concept with TSP solutions of this instance is provided, for the aim of supporting decision makers to evaluate the use of our distribution concept. Our experiments evaluate the impact on the total operating cost of our distribution concept when there are different storage capacities of satellites, various carrying size capacities of porters, multiple speeds of porters, and different costs of electricity. The need for inviting porters as part of last-mile deliveries in urban areas was confirmed in our experiments. Future research directions will include the evaluation of this distribution concept with the use of heterogeneous E-cargo bikes. It would be also interesting to consider satellites with different storage capacities, heterogeneous satellites. Such suggestions are more likely to be part of future real-world problems related to the proposed distribution concept.

Chapter Six

Conclusion

The growth of the population and e-commerce sales in urban areas because of the process of urbanisation has been steeply increasing the number of delivery vans entering the city centres leading to environmental pollution and traffic congestion. Nowadays, more than 57% of the world's population live in major cities, according to The World Bank (2022), and the process of urbanisation is foreseen to rise up further reaching 70% or more by 2050 (Bretzke, 2013). In addition, e-commerce sales are growing rapidly and have been forecasted to grow by 56% over the next years (Statista, 2022). Therefore, finding sustainable distribution concepts for last-mile delivery operations is needed. The aim of this thesis is to find a sustainable distribution concept that is not only applicable, but it is also profitable and environmentally friendly.

Chapter 2 of this thesis covered important vehicle routing problem (VRP) variants. It described the three main formulations to model VRPs, exact methods, and commonly-used heuristics for the VRP and its variants. The aim of this chapter was to give the reader the basic knowledge needed before start discussing a more complicated distribution concept introduced in this thesis which was named the truck porters routing problem (TPRP).

The TPRP arises when undertaking deliveries within urban areas where vehicle access to some customers is impossible. Thus, some of the deliveries are undertaken by porters who walk to the customers, while a truck is driven to perform deliveries to the other customers. In the TPRP, a single truck and a limited number of identical porters are available at the depot. For the customers, some must be visited by the truck, some must be served by a porter, and the remainder can be visited either by the truck or by a porter. Porters are limited by the total weight of items that they can carry and by a total working time constraint. However, a porter can revisit the depot to collect further items for delivery. The TPRP problem consists of designing a set of minimum-cost routes where each route starts and ends at the depot and satisfies capacity and travel time constraints. Chapter 3 of this thesis introduced the TPRP. It also contained two mathematical programming formulations and several families of valid inequalities for the TPRP. In the same chapter, a branch-and-cut algorithm was designed and implemented for the TPRP. Our experiments of this chapter were carried out on a set of small-size instances sampled from a real-world instance.

In Chapter 4 of this thesis, an effective variable neighborhood search (VNS) algorithm for the TPRP was designed and implemented. Our VNS algorithm was able to find an optimal solution for every instance solved to optimality by our branch-and-cut algorithm. In addition, our VNS algorithm was used to compute high-quality solutions for a set of large-size instances sampled from the same instance used to create the set of small-size instances. In our computational experiments of this chapter, our VNS algorithm was tested on the set of multi-trip vehicle routing problem (MTVRP) benchmark instances and it was able to find several new feasible solutions.

Chapter 5 of this thesis dealt with an extension of the TPRP named the TPRP with satellites (TPRPS). In the TPRPS, there is a truck-driver and a limited number of porters. The truck is assumed to be electric and each porter is assumed to be using an electric cargo bike (E-cargo bikes) for delivery. The truck departs from a depot located at the outskirts of the city, loaded with parcels that need to be delivered to customers mostly residing in urban areas. Each parcel in the truck is either directly shipped to the customer or to one of the satellites located at the outskirts of the city centre. In the latter case, when a parcel is delivered to a satellite, the parcel must be then delivered to the customer by a porter. In the TPRPS, it was assumed that E-cargo bikes are limited by the total weight and total size of parcels that they can carry. However, porters can return to any satellite to replenished their E-cargo bikes multiple times during the day.

The TPRPS is a two-echelon distribution concept that combines driving and cycling to deliver parcels in urban areas. The distribution concept was designed based on a true desire of a well-known delivery company, called Evri, to find an applicable, profitable, and eco-friendly way to deliver parcels consolidated at a depot located at the outskirts of the city to the customers living in the city centre with the minimum operating cost and the lowest possible harm to the environment. The TPRPS was tackled by an effective VNS algorithm. In our computational experiments we used a real-world instance provided by Evri. We provided a comparison between the proposed distribution concept with a traveling salesman problem (TSP) solution of this instance for the aim of supporting decision makers to evaluate the use of our distribution concept, which has shown the advantages offered by the solution concept we proposed.

Bibliography

Aghezzaf, E.-H., Raa, B., and Van Landeghem, H. (2006). Modeling inventory routing problems in supply chains of high consumption products. *European Journal of Operational Research*, 169(3):1048–1063.

Aleman, R. E. and Hill, R. R. (2010). A tabu search with vocabulary building approach for the vehicle routing problem with split demands. *International Journal of Metaheuristics*, 1(1):55–80.

Allen, J., Piecyk, M., Cherrett, T., McLeod, F., Oakey, A., Piotrowska, M., Bates, O., Bektas, T., Wise, S., Cheliotis, K., et al. (2020). Combining on-foot porters with vans for last-mile parcel deliveries: results of a study in central london. *World Review of Intermodal Transportation Research*.

Allen, J., Piecyk, M., Piotrowska, M., Cherrett, T., McLeod, F., Oakey, A., Bates, O., Friday, A., Cheliotis, K., Wise, S., et al. (2021). Combining on-foot porters with vans for last-mile parcel deliveries: results of a study in central london. *World Review of Intermodal Transportation Research*, 10(1):65–85.

Altinkemer, K. and Gavish, B. (1991). Parallel savings based heuristics for the delivery problem. *Operations Research*, 39(3):456–469.

Ambrosino, D. and Sciomachen, A. (2007). A food distribution network problem: a case study. *IMA Journal of Management Mathematics*, 18(1):33–53.

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2017). Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, 25(2):345–376.

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2019). Sustainable logistics with cargo bikes—methods and applications. In *Sustainable Transportation and Smart Logistics*, pages 207–232. Elsevier.

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2011). The traveling salesman problem. In *The Traveling Salesman Problem*. Princeton university press.

Archetti, C., Savelsbergh, M. W., and Speranza, M. G. (2006a). Worst-case analysis for split delivery vehicle routing problems. *Transportation Science*, 40(2):226–234.

Archetti, C., Savelsbergh, M. W., and Speranza, M. G. (2008). To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):114–123.

Archetti, C. and Speranza, M. G. (2004). Vehicle routing in the 1-skip collection problem. *Journal of the Operational Research Society*, 55(7):717–727.

Archetti, C. and Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19(1-2):3–22.

Archetti, C., Speranza, M. G., and Hertz, A. (2006b). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73.

Arnold, F., Cardenas, I., Sörensen, K., and Dewulf, W. (2018). Simulation of b2c e-commerce distribution in antwerp using cargo bikes and delivery points. *European Transport Research Review*, 10(1):1–13.

Arnold, F., Gendreau, M., and Sörensen, K. (2019). Efficiently solving very large-scale routing problems. *Computers & Operations Research*, 107:32–42.

Asghari, M., Al-e, S. M. J. M., et al. (2021). Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, 231:107899.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 949-M, Universite Joseph Fourier, Grenoble, France.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2007). An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178(3):755–766.

Baker, B. M. and Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800.

Baldacci, R., Bartolini, E., Mingozzi, A., and Roberti, R. (2010). An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7(3):229–268.

Baldacci, R., Battarra, M., and Vigo, D. (2008a). Routing a heterogeneous fleet of vehicles. In Golden, B. L., Raghavan, S., and Wasil, E. A., editors, *The vehicle routing problem: latest advances and new challenges*, pages 3–27. Springer, New York.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008b). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385.

Baldacci, R. and Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120:347–380.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research*, 61(2):298–314.

Bartholdi III, J. J., Platzman, L. K., Collins, R. L., and Warden III, W. H. (1983). A minimal technology routing system for meals on wheels. *Interfaces*, 13(3):1–8.

Battarra, M., Monaci, M., and Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36(11):3041–3050.

Beasley, J. E. (1983). Route first—cluster second methods for vehicle routing. *Omega*, 11(4):403–408.

BEIS (2022). Energy price guarantee. Department for Business, Energy & Industrial Strategy website, `https://www.gov.uk/government/publications/energy-bills-support/energy-bills-support-factsheet-8-september-2022`. Accessed on January 2023.

Bektaş, T. and Laporte, G. (2011). The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250.

Ben Ticha, H., Absi, N., Feillet, D., Quilliot, A., and Van Woensel, T. (2019). A branch-and-price algorithm for the vehicle routing problem with time windows on a road network. *Networks*, 73(4):401–417.

Benavent, E. and Martínez, A. (2013). Multi-depot multiple tsp: a polyhedral study and computational results. *Annals of Operations Research*, 207:7–25.

Berbotto, L., García, S., and Nogales, F. J. (2014). A randomized granular tabu search heuristic for the split delivery vehicle routing problem. *Annals of Operations Research*, 222(1):153–173.

Berg, P. W., Isaacs, S., and Blodgett, K. L. (2016). Airborne fulfillment center utilizing unmanned aerial vehicles for item delivery. United States Patent 9(305):280.

Berger, J. and Barkaoui, M. (2003). A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 54(12):1254–1262.

Bertsimas, D. J. (1992). A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585.

Bertsimas, D. J. and Simchi-Levi, D. (1996). A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44(2):286–304.

Beullens, P., Van Oudheusden, D., and Van Wassenhove, L. N. (2004). Collection and vehicle routing issues in reverse logistics. In Dekker, R., Fleischmann, M., Inderfurth, K., and van Wassenhove, L. N., editors, *Reverse Logistics, Quantitative Models for Closed-Loop Supply Chains*, pages 95–134. Springer, Berlin.

Bianchessi, N. and Irnich, S. (2019). Branch-and-cut for the split delivery vehicle routing problem with time windows. *Transportation Science*, 53(2):442–462.

Boeing, G. (2017). Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139.

Bouzaiene-Ayari, B., Dror, M., and Laporte, G. (1993). Vehicle routing with stochastic demand and split deliveries. *Foundations of Computing and Decision Sciences*, 18:63–69.

Boysen, N., Fedtke, S., and Schwerdfeger, S. (2021). Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43(1):1–58.

Braekers, K., Caris, A., and Janssens, G. K. (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, 67:166–186.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Brajevic, I. (2011). Artificial bee colony algorithm for the capacitated vehicle routing problem. In *Proceedings of the European computing conference (ECC'11)*, pages 239–244.

Brandão, J. (2009). A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 195(3):716–728.

Brandão, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research*, 38(1):140–151.

Brandão, J. (2016). A deterministic iterated local search algorithm for the vehicle routing problem with backhauls. *Top*, 24(2):445–465.

Brar, G. S. and Saini, G. (2011). Milk run logistics: literature review and directions. In *Proceedings of the world congress on engineering*, volume 1, pages 797–801, London.

Bretzke, W. R. (2013). Global urbanization: a major challenge for logistics. *Logistics Research*, 6(2):57–62.

Brown, M. (2013). Electric bike running costs. ebikeshop website, `https://www.e-bikeshop.co.uk/blogs/info/electric-bike-running-costs`. Accessed on January 2023.

Burke, E. K., Kendall, G., et al. (2005). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.* Springer, New York. ISBN: 978-1-4614-6939-1.

Caprara, A. and Fischetti, M. (1997). Branch-and-cut algorithms. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 45–63. Wiley, Chichester.

Caramia, M. and Guerriero, F. (2010). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, 61(7):1168–1180.

Carrington, D. (2021). Cargo bikes deliver faster and cleaner than vans, study finds. The Guardian website, `https://www.theguardian.com/world/2021/aug/05/cargo-bikes-deliver-faster-and-cleaner-than-vans-study-finds`. Accessed on January 2023.

Carter, C. R. and Ellram, L. M. (1998). Reverse logistics: a review of the literature and framework for future investigation. *Journal of Business Logistics*, 19(1):85–102.

Cattaruzza, D., Absi, N., and Feillet, D. (2018). Vehicle routing problems with multiple trips. *Annals of Operations Research*, 271(1):127–159.

Cattaruzza, D., Absi, N., Feillet, D., and Vidal, T. (2014). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236(3):833–848.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51.

Chao, I.-M., Golden, B. L., and Wasil, E. (1995). An improved heuristic for the period vehicle routing problem. *Networks*, 26(1):25–44.

Chen, C. and Pan, S. (2016). Using the crowd of taxis to last mile delivery in e-commerce: a methodological research. In *Service orientation in holonic and multi-agent manufacturing*, pages 61–70. Springer.

Chen, P., Huang, H.-k., and Dong, X.-Y. (2010). Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications*, 37(2):1620–1627.

Chen, Q., Li, K., and Liu, Z. (2014). Model and algorithm for an unpaired pickup and delivery vehicle routing problem with split loads. *Transportation Research Part E: Logistics and Transportation Review*, 69:218–235.

Chen, S., Golden, B., and Wasil, E. (2007). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks: An International Journal*, 49(4):318–329.

Choi, E. and Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7):2080–2095.

Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. In Christofides, N., Mingozzi, A., Toth, P., and Sandi, C., editors, *Combinatorial Optimization*, pages 315–339. Wiley, Chichester.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Conrad, R. G. and Figliozzi, M. A. (2011). The recharging vehicle routing problem. In *Proceedings of the 2011 industrial engineering research conference*, page 8, Nevada.

Coplon-Newfield, G. and Park, S.-J. (2017). Corporate Fleets Making the Switch to Electric Vehicles. `https://www.ecowatch.com/japan-seafood-sustainability-olympics-2639921287.html/`. [Online; accessed 21-August-2019].

Cordeau, J.-F. and Laporte, G. (2005). Tabu search heuristics for the vehicle routing problem. In Rego, C. and Alidaee, B., editors, *Metaheuristic Optimization via Memory and Evolution*, pages 145–163. Springer, USA.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Vehicle routing. In Barnhart, C. and Laporte, G., editors, *Handbooks in Operations*

Research and Management Science: Transportation, volume 14, pages 367–428. Elsevier, Amsterdam.

Cornuejols, G. and Harche, F. (1993). Polyhedral study of the capacitated vehicle routing problem. *Mathematical Programming*, 60(1):21–52.

Coyle, J. J., Bardi, E. J., Langley, C. J., et al. (1996). *The Management of Business Logistics*, volume 6. West publishing company, USA.

Crainic, T. G., Mancini, S., Perboli, G., and Tadei, R. (2011). Multi-start heuristics for the two-echelon vehicle routing problem. In *Evolutionary Computation in Combinatorial Optimization: 11th European Conference, EvoCOP 2011, Torino, Italy, April 27-29, 2011. Proceedings 11*, pages 179–190. Springer.

Crainic, T. G., Ricciardi, N., and Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation Science*, 43(4):432–454.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Deif, I. and Bodin, L. (1984). Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. In *Proceedings of the Babson conference on software uses in transportation and logistics management*, pages 75–96, USA.

Dekker, R., Fleischmann, M., Inderfurth, K., and van Wassenhove, L. N. (2013). *Reverse logistics: quantitative models for closed-loop supply chains*. Springer, Berlin.

Demir, E., Bektaş, T., and Laporte, G. (2014). The bi-objective pollution-routing problem. *European Journal of Operational Research*, 232(3):464–478.

Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing-problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). The vehicle routing problem with time windows. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 119–159. SIAM.

Dondo, R. and Cerdá, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507.

Dondo, R. G. and Cerdá, J. (2009). A hybrid local improvement algorithm for large-scale multi-depot vehicle routing problems with time windows. *Computers & Chemical Engineering*, 33(2):513–530.

Drexl, M. et al. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Revista de Metodos Cuantitativos para la Economia y la Empresa*, 12:5–38.

Dror, M., Ball, M., and Golden, B. (1985). A computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research*, 4(1):1–23.

Dror, M., Laporte, G., and Louveaux, F. V. (1993). Vehicle routing with stochastic demands and restricted failures. *Zeitschrift für Operations Research*, 37(3):273–283.

Dror, M., Laporte, G., and Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239–254.

Dror, M. and Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, 23(2):141–145.

Duarte, A., Mladenovic, N., Sánchez-Oro, J., and Todosijević, R. (2018). Variable neighborhood descent.

Elbert, R. and Friedrich, C. (2020). Urban consolidation and cargo bikes: a simulation study. *Transportation Research Procedia*, 48:439–451.

Erdelić, T. and Carić, T. (2019). A survey on the electric vehicle routing problem: Variants and solution approaches. *Journal of Advanced Transportation*, 2019:1–48.

Ergun, Ö., Orlin, J. B., and Steele-Feldman, A. (2006). Creating very large scale neighborhoods out of smaller ones by compounding moves. *Journal of Heuristics*, 12(1-2):115–140.

Euchi, J. and Chabchoub, H. (2010). A hybrid tabu search to solve the heterogeneous fixed fleet vehicle routing problem. *Logistics Research*, 2:3–11.

Felipe, Á., Ortuño, M. T., Righini, G., and Tirado, G. (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111–128.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642.

Fleischmann, B. (1990). The vehicle routing problem with multiple use of vehicles. Technical report, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Hamburg, Germany.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, 4(1):61–75.

Franceschetti, A., Demir, E., Honhon, D., Van Woensel, T., Laporte, G., and Stobbe, M. (2017). A metaheuristic for the time-dependent pollution-routing problem. *European Journal of Operational Research*, 259(3):972–991.

François, V., Arda, Y., Crama, Y., and Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, 255(2):422–441.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511.

Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290.

Gendreau, M., Laporte, G., and Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12.

Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of Metaheuristics*, volume 2. Springer, USA.

Gerdessen, J. C. (1996). Vehicle routing problem with trailers. *European Journal of Operational Research*, 93(1):135–147.

Ghilas, V., Cordeau, J.-F., Demir, E., and Woensel, T. V. (2018). Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. *Transportation Science*, 52(5):1191–1210.

Ghilas, V., Demir, E., and Van Woensel, T. (2016). The pickup and delivery problem with time windows and scheduled lines. *INFOR: Information Systems and Operational Research*, 54(2):147–167.

Gillett, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349.

Giosa, I., Tansini, I., and Viera, I. (2002). New assignment algorithms for the multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 53(9):977–984.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Goetschalckx, M. and Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42(1):39–51.

Golden, B., Assad, A., Levy, L., and Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66.

Golden, B. L., Magnanti, T. L., and Nguyen, H. Q. (1977). Implementing vehicle routing algorithms. *Networks*, 7(2):113–148.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998). Metaheuristics in vehicle routing. In Crainic, T. G. and Laporte, G., editors, *Fleet Management and Logistics*, pages 33–56. Springer, USA.

Gulczynski, D., Golden, B., and Wasil, E. (2010). The split delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):612–626.

Gulczynski, D., Golden, B., and Wasil, E. (2011). The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering*, 61(3):794–804.

Haimovich, M. and Rinnooy Kan, A. H. (1985). Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542.

Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2010). Variable neighborhood search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, pages 61–86. Springer.

Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2019). Variable neighborhood search. In *Handbook of Metaheuristics*, pages 57–97. Springer.

Helsgaun, K. (2017). An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, Roskilde, Denmark.

Hemmelmayr, V. C., Cordeau, J.-F., and 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.

Hernandez, F., Feillet, D., Giroudeau, R., and Naud, O. (2014). A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4OR: A Quarterly Journal of Operations Research*, 12(3):235–259.

Ho, S. C. and Gendreau, M. (2006). Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1-2):55–72.

Ho, S. C. and Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research*, 31(12):1947–1964.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., and Løkketangen, A. (2010). Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research*, 37(12):2041–2061.

Hvattum, L. M., Løkketangen, A., and Laporte, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438.

Irnich, S., Schneider, M., and Vigo, D. (2014). Four variants of the vehicle routing problem. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 241–271. SIAM.

Jia, H., Li, Y., Dong, B., and Ya, H. (2013). An improved tabu search approach to vehicle routing problem. *Procedia-Social and Behavioral Sciences*, 96:1208–1217.

Jie, W., Yang, J., Zhang, M., and 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.

Jin, M., Liu, K., and Bowden, R. O. (2007). A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Journal of Production Economics*, 105(1):228–242.

Jin, M., Liu, K., and Eksioglu, B. (2008). A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters*, 36(2):265–270.

Johnson, D. S. (1974). Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272–314.

Jonker, R. and Volgenant, T. (1983). Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2(4):161–163.

Juan, A. A., Faulin, J., Ruiz, R., Barrios, B., and Caballé, S. (2010). The sr-gcws hybrid algorithm for solving the capacitated vehicle routing problem. *Applied Soft Computing*, 10(1):215–224.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330.

Karkory, F. A. and Abudalmola, A. A. (2013). Implementation of heuristics for solving travelling salesman problem using nearest neighbour and minimum spanning tree algorithms. *International Journal of Computer and Information Engineering*, 7(10):1524–1534.

Keskin, M. and Çatay, B. (2018). A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172–188.

Keskin, M., Laporte, G., and Çatay, B. (2019). Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research*, 107:77–94.

Knight, K. and Hofer, J. (1968). Vehicle scheduling with timed and connected calls: A case study. *Journal of the Operational Research Society*, 19(3):299–310.

Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G. (2014). The fleet size and mix pollution-routing problem. *Transportation Research Part B: Methodological*, 70:239–254.

Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G. (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, 249(1):1–21.

Koc, C. and Karaoglan, I. (2011). A branch and cut algorithm for the vehicle routing problem with multiple use of vehicles. In *41st International Conference on Computers & Industrial Engineering*, pages 554–559.

Koç, Ç. and Laporte, G. (2018). Vehicle routing with backhauls: Review and research perspectives. *Computers & Operations Research*, 91:79–91.

Kucukoglu, I., Dewil, R., and Cattrysse, D. (2021). The electric vehicle routing problem and its variations: A literature review. *Computers & Industrial Engineering*, 161:107650.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757.

Lambert, V., Laporte, G., and Louveaux, F. (1993). Designing collection routes through bank branches. *Computers & Operations Research*, 20(7):783–791.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

Laporte, G., Mercure, H., and Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46.

Laporte, G. and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. In Martello, S., Laporte, G., Minoux, M., and Ribeiro, C., editors, *Surveys in Combinatorial Optimization*, volume 132, pages 147–184. Elsevier.

Laporte, G., Nobert, Y., and Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073.

Laporte, G. and Semet, F. (2002). Classical heuristics for the capacitated vrp. In Toth, P. and Vigo, D., editors, *The vehicle routing problem*, pages 109–128. SIAM.

Larson, R. C. (1988). Transporting sludge to the 106-mile site: An inventory/routing model for fleet sizing and logistics system design. *Transportation Science*, 22(3):186–198.

Lee, C.-G., Epelman, M. A., White III, C. C., and Bozer, Y. A. (2006). A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research Part B: Methodological*, 40(4):265–284.

Lenstra, J. K. and Rinnooy Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Letchford, A. N. and Salazar-González, J.-J. (2006). Projection results for vehicle routing. *Mathematical Programming*, 105(2-3):251–274.

Letchford, A. N. and Salazar-González, J.-J. (2015). Stronger multi-commodity flow formulations of the capacitated vehicle routing problem. *European Journal of Operational Research*, 244(3):730–738.

Li, B., Krushinsky, D., Reijers, H. A., and Van Woensel, T. (2014). The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research*, 238(1):31–40.

Li, C., Ding, T., Liu, X., and Huang, C. (2018). An electric vehicle routing optimization model with hybrid plug-in and wireless charging systems. *IEEE Access*, 6:27569–27578.

Li, F., Golden, B., and Wasil, E. (2005). Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179.

Li, H., Bai, M., Zhao, Y., and Dai, C. (2019). Vehicle flow formulation for two-echelon time-constrained vehicle routing problem. *Journal of Management Science and Engineering*, 4(2):75–90.

Li, H., Chang, X., Zhao, W., and Lu, Y. (2017). The vehicle flow formulation and savings-based algorithm for the rollon-rolloff vehicle routing problem. *European Journal of Operational Research*, 257(3):859–869.

Li-ying, W. and Yuan-bin, S. (2015). Multiple charging station location-routing problem with time window of electric vehicle. *Journal of Engineering Science & Technology Review*, 8(5):190–201.

Liao, C.-S., Lu, S.-H., and Shen, Z.-J. M. (2016). The electric vehicle touring problem. *Transportation Research Part B: Methodological*, 86:163–180.

Lin, C., Choy, K. L., Ho, G. T., Chung, S. H., and Lam, H. (2014). Survey of green vehicle routing problem: past and future trends. *Expert Systems with Applications*, 41(4):1118–1138.

Lin, J., Zhou, W., and Wolfson, O. (2016). Electric vehicle routing problem. *Transportation Research Procedia*, 12:508–521.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

Lin, S.-W., Lee, Z.-J., Ying, K.-C., and Lee, C.-Y. (2009a). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications*, 36(2):1505–1512.

Lin, S.-W., Vincent, F. Y., and Chou, S.-Y. (2009b). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692.

Lin, S.-W., Vincent, F. Y., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244–15252.

Liu, S., Huang, W., and Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, 45(3):434–445.

LPC (2022). The national minimum wage in 2022. Gov.uk website, `https://www.gov.uk/government/publications/the-national-minimum-wage-in-2022`. Accessed on January 2023.

Lysgaard, J., Letchford, A. N., and Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445.

Madsen, O. (1976). Optimal scheduling of trucks-A routing problem with tight due times for delivery. In Strobel, H., Genser, R., and Etschmaier, M., editors, *Optimization Applied to Transportation Systems*, pages 129–136. International Institute for Applied System Analysis, Luxemburg.

Magnanti, T. L. (1981). Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. *Networks*, 11(2):179–213.

Marinakis, Y., Marinaki, M., and Dounias, G. (2010). Honey bees mating optimization algorithm for large scale vehicle routing problems. *Natural Computing*, 9(1):5–27.

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

Martinez-Sykora, A., McLeod, F., Lamas-Fernandez, C., Bektaş, T., Cherrett, T., and Allen, J. (2020). Optimised solutions to the last-mile delivery problem in london using a combination of walking and driving. *Annals of Operations Research*, pages 1–49.

McLeod, F., Cherrett, T., Bektas, T., Allen, J., Martinez-Sykora, A., Lamas-Fernandez, C., Bates, O., Cheliotis, K., Friday, A., Piecyk, M., et al. (2020). Quantifying environmental and financial benefits of using porters and cycle couriers for last-mile parcel delivery. *Transportation Research Part D: Transport and Environment*, 82:102311.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, 37(11):1886–1898.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329.

Min, H., Current, J., and Schilling, D. (1992). The multiple depot vehicle routing problem with backhauling. *Journal of Business Logistics*, 13(1):259.

Mingozzi, A., Giorgi, S., and Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33(3):315–329.

Mingozzi, A., Roberti, R., and Toth, P. (2013). An exact algorithm for the multi-trip vehicle routing problem. *INFORMS Journal on Computing*, 25(2):193–207.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Moghdani, R., Salimifard, K., Demir, E., and Benyettou, A. (2021). The green vehicle routing problem: A systematic literature review. *Journal of Cleaner Production*, 279:123691.

Mole, R. and Jameson, S. (1976). A sequential route-building algorithm employing a generalised savings criterion. *Journal of the Operational Research Society*, 27(2):503–511.

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

Mullaseril, P. A., Dror, M., and Leung, J. (1997). Split-delivery routeing heuristics in livestock feed distribution. *Journal of the Operational Research Society*, 48(2):107–116.

Naddef, D. and Rinaldi, G. (2002). Branch-and-cut algorithms for the capacitated vrp. In Toth, P. and Vigo, D., editors, *The vehicle routing problem*, pages 53–84. SIAM.

Nagata, Y. and Bräysy, O. (2009). Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks: An International Journal*, 54(4):205–215.

Nagy, G. and Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1):126–141.

Narayanan, S. and Antoniou, C. (2021). Electric cargo cycles-a comprehensive review. *Transport Policy*.

Nguyen, T., Bektaş, T., Cherrett, T. J., McLeod, F. N., Allen, J., Bates, O., Piotrowska, M., Piecyk, M., Friday, A., and Wise, S. (2019). Optimising parcel deliveries in london using dual-mode routing. *Journal of the Operational Research Society*, 70(6):998–1010.

Olivera, A. and Viera, O. (2007). Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1):28–47.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451.

Osman, I. H. and Wassan, N. A. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, 5(4):263–285.

Oyola, J., Arntzen, H., and Woodruff, D. L. (2018). The stochastic vehicle routing problem, a literature review, part i: models. *EURO Journal on Transportation and Logistics*, 7(3):193–221.

Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34(3):336–344.

Paradiso, R., Roberti, R., Laganá, D., and Dullaert, W. (2020). An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, 68(1):180–198.

Paz, J., Granada-Echeverri, M., and Escobar, J. (2018). The multi-depot electric vehicle location routing problem with time windows. *International Journal of Industrial Engineering Computations*, 9(1):123–136.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

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

Petch, R. J. and Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133(1-3):69–92.

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

Poggi, M. and Uchoa, E. (2014). New exact algorithms for the capacitated vehicle routing problem. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 59–86. SIAM.

Polacek, M., Hartl, R. F., Doerner, K., and Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):613–627.

Prins, C. (2002). Efficient heuristics for the heterogeneous fleet multitrip vrp with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms*, 1(2):135–150.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

Pullen, H. and Webb, M. (1967). A computer application to a transport scheduling problem. *The Computer Journal*, 10(1):10–13.

Queiroga, E., Frota, Y., Sadykov, R., Subramanian, A., Uchoa, E., and Vidal, T. (2020). On the exact solution of vehicle routing problems with backhauls. *European Journal of Operational Research*, 287(1):76–89.

Raeesi, R. and Zografos, K. G. (2019). The multi-objective steiner pollution-routing problem on congested urban road networks. *Transportation Research Part B: Methodological*, 122:457–485.

Ramos, T. R. P., Gomes, M. I., and Póvoa, A. P. B. (2020). Multi-depot vehicle routing problem: a comparative study of alternative formulations. *International Journal of Logistics Research and Applications*, 23(2):103–120.

Reimann, M., Doerner, K., and Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591.

Rochat, Y. and Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167.

Rodrigue, J.-P., Comtois, C., and Slack, B. (2016). *The geography of transport systems*. Routledge, New York.

Ropke, S. and Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775.

Russell, R. A. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions. *Operations Research*, 25(3):517–524.

Ryan, D. M., Hjorring, C., and Glover, F. (1993). Extensions of the petal method for vehicle routeing. *Journal of the Operational Research Society*, 44(3):289–296.

Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6-7):971–983.

Salhi, S., Imran, A., and Wassan, N. A. (2014). The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. *Computers & Operations Research*, 52:315–325.

Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10):1034–1042.

Salhi, S. and Petch, R. (2007). A ga based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, 6(4):591–613.

Salhi, S. and Rand, G. K. (1987). Improvements to vehicle routeing heuristics. *Journal of the Operational Research Society*, 38(3):293–295.

Salhi, S. and Sari, M. (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, 103(1):95–112.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2015). A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Transportation Science*, 49(2):355–368.

Sar, K. and Ghadimi, P. (2023). A systematic literature review of the vehicle routing problem in reverse logistics operations. *Computers & Industrial Engineering*, page 109011.

Sarasola, B., Doerner, K. F., Schmid, V., and Alba, E. (2016). Variable neighborhood search for the stochastic and dynamic vehicle routing problem. *Annals of Operations Research*, 236:425–461.

Sassi, O., Cherif, W. R., and Oulamara, A. (2014). Vehicle routing problem with mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs. *Int. J. Math. Comput. Stat. Nat. Phys. Eng*, 9(3):148–158.

Savelsbergh, M. and Van Woensel, T. (2016). 50th anniversary invited article—city logistics: Challenges and opportunities. *Transportation Science*, 50(2):579–590.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33(4):894–909.

Schiffer, M. and Walther, G. (2017). The electric location routing problem with time windows and partial recharging. *European Journal of Operational Research*, 260(3):995–1013.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520.

Semet, F., Toth, P., and Vigo, D. (2014). Classical exact algorithms for the capacitated vehicle routing problem. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 37–57. SIAM.

Sierksma, G. and Tijssen, G. A. (1998). Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research*, 76:261–286.

Simsir, F. and Ekmekci, D. (2019). A metaheuristic solution approach to capacitied vehicle routing and network optimization. *Engineering Science and Technology, an International Journal*, 22(3):727–735.

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

Song, S. H., Lee, K. S., and Kim, G. S. (2002). A practical approach to solving a newspaper logistics problem using a digital map. *Computers & Industrial Engineering*, 43(1-2):315–330.

Statista (2015). Average speed on roads in great britain in 2014, by road and vehicle type. Statista website, `https://www.statista.com/statistics/303443/average-speed-on-different-roads-in-great-britain-by-vehicle-type/`. Accessed on January 2023.

Statista (2022). Retail e-commerce sales worldwide from 2014 to 2026. Statista website, `https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/`. Accessed on October 2022.

Statz, A. (2018). Why does amazon pack small items in huge boxes? this question plagues its customers. simplemost website, `https://www.simplemost.com/amazon-pack-small-items-big-boxes/`. Accessed on January 2023.

Suizo, G. (2013). Large and Recharged: Top Electric Fleets. `https://www.greenfleetmagazine.com/155563/large-recharged-top-electric-fleets/`. [Online; accessed 21-August-2019].

Szeto, W. Y., Wu, Y., and Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, 215(1):126–135.

Taillard, É. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673.

Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO-Operations Research*, 33(1):1–14.

Taillard, E. D., Laporte, G., and Gendreau, M. (1996). Vehicle routeing with multiple use of vehicles. *Journal of the Operational Research Society*, 47(8):1065–1070.

Tamke, F. and Buscher, U. (2021). A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 144:174–203.

Taniguchi, E. and Van Der Heijden, R. E. (2000). An evaluation methodology for city logistics. *Transport Reviews*, 20(1):65–90.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research*, 32(9):2309–2327.

Tarantilis, C. D. and Kiranoudis, C. T. (2002). Boneroute: An adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115(1-4):227–241.

Tavakkoli-Moghaddam, R., Safaei, N., Kah, M., and Rabbani, M. (2007). A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. *Journal of the Franklin Institute*, 344(5):406–425.

TFL (2013). How does the road network perform in terms of speed, congestion and journey time reliability? `https://tfl.gov.uk/corporate/publications-and-reports/rtf-supporting-documents`. [Online; accessed 11-August-2022].

The World Bank (2022). Urban population. The World Bank website, `http://https://data.worldbank.org/indicator/SP.URB.TOTL`. Accessed on October 2022.

Tillman, F. A. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204.

Toth, P. and Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31(4):372–385.

Toth, P. and Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113(3):528–543.

Toth, P. and Vigo, D. (2002a). Branch-and-bound algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 29–51. SIAM.

Toth, P. and Vigo, D. (2002b). *The Vehicle Routing Problem.* SIAM, Philadelphia.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on Computing*, 15(4):333–346.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications.* SIAM, Philadelphia.

Tsirimpas, P., Tatarakis, A., Minis, I., and Kyriakidis, E. (2008). Single vehicle routing with a predefined customer sequence and multiple depot returns. *European Journal of Operational Research*, 187(2):483–495.

Van Buer, M. G., Woodruff, D. L., and Olson, R. T. (1999). Solving the medium newspaper production/distribution problem. *European Journal of Operational Research*, 115(2):237–253.

Van Duin, J., Tavasszy, L. A., and Quak, H. (2013). Towards E (lectric)-urban freight: first promising steps in the electric vehicle revolution. *European Transport - Trasporti Europei*, 54(9).

Van Goor, A. R. (1980). *Distributie en logistiek.* Stenfert Kroese, Leiden.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2011). A grasp with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38(9):1319–1334.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244.

Wang, H., Du, L., and Ma, S. (2014). Multi-objective open location-routing model with split delivery for optimized relief distribution in post-earthquake. *Transportation Research Part E: Logistics and Transportation Review*, 69:160–179.

Wark, P. and Holt, J. (1994). A repeated matching heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, 45(10):1156–1167.

Wasner, M. and Zäpfel, G. (2004). An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. *International Journal of Production Economics*, 90(3):403–419.

Woodman, O. (2015). London crowned as strava's most active cycling city. bikeradar website, `https://www.bikeradar.com/news/london-crowned-as-stravas-most-active-cycling-city/`. Accessed on January 2023.

Wren, A. and Carr, J. (1971). *Computers in Transport Planning and Operation.* Ian Allan, London.

Wren, A. and Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Journal of the Operational Research Society*, 23(3):333–344.

Wu, T.-H., Low, C., and Bai, J.-W. (2002). Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29(10):1393–1415.

Wu, Z. and Zhang, J. (2021). A branch-and-price algorithm for two-echelon electric vehicle routing problem. *Complex & Intelligent Systems*, pages 1–16.

Xu, Z. and Cai, Y. (2018). Variable neighborhood search for consistent vehicle routing problem. *Expert Systems with Applications*, 113:66–76.

Yellow, P. (1970). A computational modification to the savings method of vehicle scheduling. *Journal of the Operational Research Society*, 21(2):281–283.

Yilmaz, Y. and Kalayci, C. B. (2022). Variable neighborhood search algorithms to solve the electric vehicle routing problem with simultaneous pickup and delivery. *Mathematics*, 10(17):3108.

Young, K., Wang, C., Wang, L. Y., and Strunz, K. (2013). Electric vehicle battery technologies. In Garcia-Valle, R. and Lopes, J. A. P., editors, *Electric Vehicle Integration into Modern Power Networks*, pages 15–56. Springer, New York.

Yu, B., Yang, Z.-Z., and Yao, B. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1):171–176.

Zachariadis, E. E. and Kiranoudis, C. T. (2010). A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, 37(12):2089–2105.

Zachariadis, E. E. and Kiranoudis, C. T. (2012). An effective local search approach for the vehicle routing problem with backhauls. *Expert Systems with Applications*, 39(3):3174–3184.

Zhang, X., Chen, L., Gendreau, M., and Langevin, A. (2022). A branch-and-cut algorithm for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 302(1):259–269.

Appendix

Table A1: Average number of added constraints at the root node with one family of inequalities.

i	CCCs (11)	SECs (12)	1	2	3	4	5	6	total
LR	60.07	11.67	-	-	-	-	-	-	71.73
1	55.87	10.13	48.87	-	-	-	-	-	114.87
2	59.13	12.33	-	2.53	-	-	-	-	74.00
3	66.60	12.07	-	-	40.20	-	-	-	118.87
4	67.67	13.00	-	-	-	30.13	-	-	110.80
5	61.60	11.73	-	-	-	-	451.07	-	524.40
6	68.60	12.53	-	-	-	-	-	608.67	689.80
Full	79.40	11.73	54.73	4.87	42.13	32.00	328.80	555.67	1109.33

Table A2: Average number of added constraints at the root node without one family of inequalities.

i	CCCs (11)	SECs (12)	1	2	3	4	5	6	total
1	81.40	11.27	-	3.73	38.27	28.00	304.40	478.53	945.60
2	77.40	10.33	50.40	-	40.73	30.40	301.47	463.87	974.60
3	71.13	11.00	56.40	4.33	-	32.67	337.53	490.07	1003.13
4	80.07	10.27	55.00	3.80	52.93	-	288.47	529.07	1019.60
5	74.27	11.07	44.33	3.80	39.33	31.60	-	481.73	686.13
6	78.27	10.33	44.53	3.27	40.93	28.73	311.67	-	517.73

Table A3: Column headings.

column heading	description
instance	name of the problem instance
LB	the value of the lower bound at the termination
UB	the value of the upper bound at the termination, if feasible
gap(%)	optimality gap in percentage, that is Gap = $\frac{UB-LB}{UB} \times 100$
opt	the number of instances solved to optimality
N nodes	total number of nodes explored in the branch-and-cut tree
time(s)	total computation time in seconds
N cuts	number of generated cuts

Table A4: The result of $B\&C1$. See table A3 for column headings description.

	LB		UB		gap (%)					
instance	best	av	best	av	best	av	opt	N nodes	time(s)	N cuts
N13_a	3198.60	3198.60	3198.60	3198.60	0.00	0.00	5	4.64	0.37	8.52
N16_a	3642.20	3642.20	3642.20	3642.20	0.00	0.00	5	1022.56	156.53	40.84
N19_a	4341.60	4341.60	4341.60	4341.60	0.00	0.00	5	5665.40	827.01	86.36
N22_a	4552.19	4544.43	4563.80	4564.00	0.26	0.43	4	5786.72	922.63	142.44
N25_a	5487.63	5459.26	5518.80	5536.36	0.54	1.32	3	10651.64	1791.01	231.84
N28_a	5969.26	5942.14	6243.40	6595.64	4.60	8.22	1	7449.36	1899.33	357.32
N31_a	6375.84	6357.22	6698.80	7145.11	4.87	9.69	1	7062.92	3181.65	412.4
N34_a	6763.02	6750.21	7015.00	7310.72	3.52	6.70	2	4906.76	4162.31	485.6
N37_a	7396.95	7388.48	7726.60	7847.51	4.35	5.79	0	3815.80		536.68
N13_b	3088.80	3088.80	3088.80	3088.80	0.00	0.00	5	25.92	34.86	16.44
N16_b	3447.80	3447.80	3447.80	3447.80	0.00	0.00	5	118.40	287.38	31.56
N19_b	4129.99	4112.60	4180.20	4199.20	1.22	1.87	3	7714.64	255.27	159.6
N22_b	4600.42	4552.64	4686.60	4766.36	1.79	4.38	3	11186.80	5202.70	305.84
N25_b	5014.04	4989.00	5416.00	5604.44	7.41	10.56	0	7179.52		502.04
N28_b	5402.47	5388.27	5937.75	6568.23	6.06	12.39	0	5466.76		582.28
N31_b	5820.72	5803.16	6671.20	7001.82	12.77	16.76	0	4167.88		567.12
N34_b	6147.80	6134.18	7945.00	9138.92	16.04	23.51	0	2292.12		541.44
N37_b	6706.81	6692.99	11826.00	11826.00	16.20	16.20	0	1621.44		407.04
N13_c	2889.20	2889.20	2889.20	2889.20	0.00	0.00	5	343.40	133.66	35.2
N16_c	3269.40	3269.40	3269.40	3269.40	0.00	0.00	5	774.60	473.58	48.12
N19_c	3955.31	3928.98	3998.20	4118.00	0.99	3.91	3	4659.24	969.80	190.96
N22_c	4030.34	4014.19	4428.20	4648.80	8.77	12.76	1	4671.56	1813.01	314.32
N25_c	4547.04	4523.06	5416.00	6340.56	15.70	25.94	0	4481.52		545.56
N28_c	5009.05	4994.68	7095.80	7686.00	28.25	33.35	0	2986.84		571.2
N31_c	5386.71	5358.87	9073.00	9928.67	30.75	35.80	0	1553.24		399.04
N34_c	5626.03	5604.67	9282.33	9322.33	24.80	24.96	0	1002.32		409.2
N37_c	6002.96	5986.61	11181.25	11181.25	37.03	37.03	0	576.20		337.84

Table A5: The result of $B\&C2$. See table A3 for column headings description.

instance	LB best	LB av	UB best	UB av	gap (%) best	gap (%) av	opt	N nodes	time(s)	N cuts
N13_a	3198.60	3198.60	3198.60	3198.60	0.00	0.00	5	3.40	0.34	8.76
N16_a	3642.20	3642.20	3642.20	3642.20	0.00	0.00	5	1311.72	278.55	43.72
N19_a	4341.60	4341.60	4341.60	4341.60	0.00	0.00	5	8184.72	1269.35	95.76
N22_a	4549.60	4534.99	4563.80	4568.44	0.31	0.73	4	4548.40	911.53	128.96
N25_a	5490.65	5447.20	5518.80	5554.60	0.53	1.85	4	12126.88	3025.17	234.64
N28_a	5960.62	5952.56	6243.00	6335.08	4.51	5.95	1	7181.68	846.12	376.52
N31_a	6374.29	6353.77	6675.60	6997.12	4.58	8.55	1	6861.84	2867.80	410.2
N34_a	6762.93	6737.50	7006.20	7535.40	3.51	8.82	2	4239.16	1306.43	459.2
N37_a	7394.39	7387.65	9597.00	9837.10	19.38	21.73	0	4008.44		582.72
N13_b	3088.80	3088.80	3088.80	3088.80	0.00	0.00	5	25.76	20.81	16.76
N16_b	3447.80	3447.80	3447.80	3447.80	0.00	0.00	5	130.64	265.63	32.32
N19_b	4148.70	4124.27	4176.00	4190.64	0.61	1.44	4	7950.16	1625.44	144.64
N22_b	4593.77	4544.04	4683.20	4764.36	1.94	4.54	2	9665.32	3797.62	282.44
N25_b	5016.07	4989.15	5393.20	5543.72	7.12	9.73	0	6938.32		479.56
N28_b	5398.61	5380.06	5997.20	6649.27	10.08	17.61	0	5117.72		534.36
N31_b	5818.75	5802.24	6795.50	7875.88	13.14	21.00	0	4394.28		579.56
N34_b	6151.87	6136.01	7626.50	8439.29	12.84	18.44	0	2377.84		539.96
N37_b	6709.31	6696.03	7506.00	7506.00	3.45	3.45	0	1596.32		380.6
N13_c	2889.20	2889.20	2889.20	2889.20	0.00	0.00	5	357.32	135.67	35.12
N16_c	3269.40	3269.40	3269.40	3269.40	0.00	0.00	5	974.44	465.34	54.16
N19_c	3943.50	3929.61	4027.60	4094.64	1.92	3.57	3	4594.84	999.69	175
N22_c	4042.22	4015.90	4362.40	4779.80	7.23	14.64	1	5642.04	2073.46	399.36
N25_c	4538.23	4521.96	5633.20	6385.60	18.51	27.22	0	4998.84		580.96
N28_c	5004.68	4990.41	7641.60	8154.79	34.62	38.12	0	2928.88		567.08
N31_c	5383.12	5364.59	8676.60	8968.10	36.85	39.34	0	1618.44		401.12
N34_c	5627.57	5607.78	9690.20	10243.10	41.20	44.70	0	1016.28		383.48
N37_c	6007.35	5990.51	10541.60	10571.30	43.14	43.33	0	558.88		304.44

Table A6: The result of $B\&C3$. See table A3 for column headings description.

	LB		UB		gap (%)					
instance	best	av	best	av	best	av	opt	N nodes	time(s)	N cuts
N13_a	3198.60	3198.60	3198.60	3198.60	0.00	0.00	5	1.16	81.02	7.96
N16_a	3642.20	3642.20	3642.20	3642.20	0.00	0.00	5	1168.20	536.51	42.92
N19_a	4341.60	4321.72	4341.60	4345.24	0.00	0.49	5	4494.00	2089.27	87.44
N22_a	4531.72	4510.44	4563.80	4568.84	0.78	1.28	4	3283.24	1938.57	119.88
N25_a	5452.41	5404.77	5522.80	5551.04	1.20	2.52	3	6366.48	3856.27	194.68
N28_a	5958.27	5941.92	6196.40	6309.76	3.94	5.69	1	4523.48	1568.95	303.4
N31_a	6326.50	6315.90	6653.20	6944.23	4.94	8.61	0	3466.16		335.88
N34_a	6735.95	6723.91	7403.40	7635.81	7.57	10.69	0	2008.68		366.96
N37_a	7387.46	7379.13	7857.50	7889.83	2.48	2.64	0	1263.00		297.68
N13_b	3088.80	3088.80	3088.80	3088.80	0.00	0.00	5	41.24	306.31	14.56
N16_b	3447.80	3447.80	3447.80	3447.80	0.00	0.00	5	270.84	332.87	29.08
N19_b	4131.60	4109.89	4180.60	4200.32	1.14	1.96	3	3377.20	509.20	114.44
N22_b	4554.39	4517.70	4687.20	4728.12	3.07	4.44	2	5180.88	4642.48	229.56
N25_b	4999.59	4979.42	5329.40	5497.76	6.10	9.13	0	3948.12		392.88
N28_b	5395.55	5373.52	5939.00	6285.06	6.18	10.12	0	2360.56		411.4
N31_b	5812.48	5800.88	6441.33	6441.33	6.76	6.76	0	1337.52		323.12
N34_b	6139.08	6125.24	6504.00	6504.00	1.30	1.30	0	740.80		300.48
N37_b	6698.72	6678.43	NF	NF	NF	NF	0	410.20		210.52
N13_c	2889.20	2889.20	2889.20	2889.20	0.00	0.00	5	510.88	320.34	33.56
N16_c	3269.40	3269.40	3269.40	3269.40	0.00	0.00	5	1169.36	750.27	49.56
N19_c	3941.54	3927.10	3998.20	4043.52	1.31	2.58	3	3336.40	2962.46	136.64
N22_c	4018.28	3999.02	4398.80	4636.98	8.65	12.80	1	3209.92	2044.61	301.04
N25_c	4518.67	4503.14	5463.67	5732.39	9.52	12.05	0	2378.04		403.96
N28_c	5000.78	4980.79	7484.00	7484.00	12.84	12.84	0	1063.32		340.16
N31_c	5377.16	5360.35	NF	NF	NF	NF	0	380.52		191.32
N34_c	5659.87	5638.08	NF	NF	NF	NF	0	178.40		190.24
N37_c	6049.67	6023.15	NF	NF	NF	NF	0	61.44		161.24

‘NF’ implies that no feasible solution was found by CPLEX within the 2-h limit.

Table A7: The result of $C\&B$. See table A3 for column headings description.

	LB		UB		gap (%)					
instance	best	av	best	av	best	av	opt	N nodes	time(s)	N cuts
N13_a	3198.60	3198.60	3198.60	3198.60	0.00	0.00	5	2.60	0.33	8.48
N16_a	3642.20	3642.20	3642.20	3642.20	0.00	0.00	5	1344.96	353.98	26.96
N19_a	4341.60	4341.60	4341.60	4341.60	0.00	0.00	5	6954.00	812.56	44.52
N22_a	4563.80	4544.78	4563.80	4565.40	0.00	0.45	5	6990.96	1530.77	67.24
N25_a	5461.37	5434.42	5524.80	5554.08	1.10	2.07	2	15810.00	915.76	136.88
N28_a	5971.83	5951.38	6236.20	6416.96	4.45	7.06	1	12091.80	1452.91	145.56
N31_a	6385.58	6357.99	6726.60	7187.52	5.03	10.49	1	17043.88	5257.44	161.92
N34_a	6782.09	6748.23	6961.80	7249.88	2.57	6.46	2	11172.40	2389.74	139.04
N37_a	7405.64	7392.85	7935.00	8512.72	6.77	12.06	0	19832.56		172.92
N13_b	3088.80	3088.80	3088.80	3088.80	0.00	0.00	5	41.28	28.32	16.56
N16_b	3447.80	3447.80	3447.80	3447.80	0.00	0.00	5	110.52	242.61	24.12
N19_b	4123.45	4116.80	4176.00	4208.44	1.20	1.95	3	9613.04	178.66	79
N22_b	4571.06	4544.12	4696.20	4734.52	2.67	3.99	2	12386.12	4330.28	119.84
N25_b	5018.13	4995.20	5366.80	5541.40	6.47	9.62	0	11597.20		164.96
N28_b	5413.37	5394.57	6052.20	6611.64	10.64	17.28	0	11013.08		140.72
N31_b	5848.22	5825.61	6388.60	7275.48	8.77	18.58	0	25568.40		203
N34_b	6188.43	6165.70	7047.00	7887.92	12.03	20.40	0	28744.68		261.68
N37_b	6767.70	6743.32	7981.60	8646.38	15.40	21.45	0	61955.28		341.72
N13_c	2889.20	2889.20	2889.20	2889.20	0.00	0.00	5	362.64	130.95	27.16
N16_c	3269.40	3269.40	3269.40	3269.40	0.00	0.00	5	912.00	469.07	28.64
N19_c	3956.14	3935.87	3998.20	4052.12	1.04	2.55	3	6965.00	712.63	81.48
N22_c	4039.77	4021.67	4407.40	4765.32	8.29	13.88	1	8569.52	974.06	131.6
N25_c	4536.72	4524.61	5548.20	6111.28	18.11	24.96	0	11169.36		160.88
N28_c	5057.12	5024.91	6035.00	6962.72	16.34	26.75	0	31294.24		246.32
N31_c	5456.20	5420.76	6482.80	7214.88	15.89	24.21	0	66944.72		312.2
N34_c	5706.37	5684.35	7853.80	8862.04	25.60	33.39	0	37859.92		270.76
N37_c	6137.36	6098.39	9660.00	10483.00	20.79	23.90	0	66200.16		260.88

Table A8: The result of *B&B*. See table A3 for column headings description.

	LB		UB		gap (%)				
instance	best	av	best	av	best	av	opt	N nodes	time(s)
N13_a	3198.60	3198.60	3198.60	3198.60	0.00	0.00	5	675.00	111.24
N16_a	3625.70	3625.67	3642.20	3642.20	0.43	0.43	4	6697.48	889.33
N19_a	4230.87	4230.82	4352.20	4352.20	2.49	2.49	2	6518.20	1337.89
N22_a	4426.87	4426.87	4599.60	4599.60	3.73	3.73	1	5555.88	1546.06
N25_a	5255.14	5255.14	5533.00	5533.64	4.97	4.98	0	8968.44	
N28_a	5839.27	5839.27	6322.40	6329.00	7.59	7.68	0	7315.40	
N31_a	6227.93	6227.93	6981.60	6981.60	10.60	10.60	0	11003.04	
N34_a	6594.48	6594.44	7262.00	7313.12	8.96	9.46	0	19341.48	
N37_a	7293.61	7293.54	6908.00	6908.00	0.97	0.97	0	20749.56	
N13_b	3088.80	3088.80	3088.80	3088.80	0.00	0.00	5	1325.40	433.80
N16_b	3447.80	3447.80	3447.80	3447.80	0.00	0.00	5	3516.80	1677.75
N19_b	4056.39	4056.31	4200.60	4200.60	3.19	3.20	2	6783.68	2342.23
N22_b	4384.73	4384.73	4776.40	4776.40	8.15	8.15	0	8814.12	
N25_b	4887.29	4887.09	5578.40	5583.52	12.02	12.10	0	10598.00	
N28_b	5282.61	5282.57	6449.80	6454.12	17.83	17.89	0	12944.88	
N31_b	5767.12	5767.01	6850.50	6854.50	11.37	11.42	0	15549.04	
N34_b	6033.39	6031.53	NF	NF	NF	NF	0	8119.88	
N37_b	6609.72	6608.22	NF	NF	NF	NF	0	5614.56	
N13_c	2889.20	2889.20	2889.20	2889.20	0.00	0.00	5	5091.20	2409.23
N16_c	3236.05	3235.66	3269.80	3269.80	1.00	1.01	4	937.44	2408.22
N19_c	3865.03	3864.92	4055.20	4055.20	4.39	4.39	1	8618.12	5456.04
N22_c	3875.89	3875.89	4547.20	4547.20	14.49	14.49	0	5503.36	
N25_c	4431.48	4431.00	5359.40	5363.12	17.32	17.38	0	9264.88	
N28_c	4928.77	4928.77	6556.67	6556.67	14.80	14.80	0	11433.00	
N31_c	5303.67	5303.45	NF	NF	NF	NF	0	7315.68	
N34_c	5555.24	5555.21	NF	NF	NF	NF	0	3560.88	
N37_c	5957.35	5957.35	NF	NF	NF	NF	0	2284.96	

'NF' implies that no feasible solution was found by CPLEX within the 2-h limit.

Table A9: Time needed to solve optimality-solved instances exactly.

		time(s)				
instance	optimal	$B\&C1$	$B\&C2$	$B\&C3$	$C\&B$	$B\&B$
Leuven1_N13_a1	2984	**0.20**	0.20	0.33	0.22	2.17
Leuven1_N13_a2	3721	0.48	0.45	125.63	**0.44**	201.72
Leuven1_N13_a3	3133	0.34	0.34	177.45	**0.33**	162.47
Leuven1_N13_a4	3269	0.82	0.66	101.58	**0.61**	188.21
Leuven1_N13_a5	2886	0.04	0.03	0.11	**0.03**	1.66
Leuven1_N16_a1	3248	0.18	0.18	33.85	**0.17**	13.31
Leuven1_N16_a2	3388	0.20	**0.20**	66.85	0.20	715.03
Leuven1_N16_a3	3812	**387.94**	453.71	880.53	723.08	
Leuven1_N16_a4	3682	**155.76**	221.88	359.90	402.99	1156.28
Leuven1_N16_a5	4081	**238.58**	716.77	1341.42	643.46	1672.70
Leuven1_N19_a1	4786	**1833.81**	3226.36	5656.44	1854.38	
Leuven1_N19_a2	3865	0.44	0.44	2.80	**0.43**	573.92
Leuven1_N19_a3	4394	**775.67**	960.44	1380.85	834.92	
Leuven1_N19_a4	4988	952.69	727.50	2461.23	**604.30**	
Leuven1_N19_a5	3675	**572.44**	1432.01	945.03	768.74	2101.86
Leuven1_N22_a1	4533				**5304.93**	
Leuven1_N22_a2	4363	1018.15	1430.19	**520.46**	652.73	
Leuven1_N22_a3	4777	**168.46**	226.63	1330.29	184.28	1546.06
Leuven1_N22_a4	4591	787.28	**361.21**	1112.87	811.57	
Leuven1_N22_a5	4555	1716.64	1628.11	4790.68	**700.32**	
Leuven1_N25_a1	5324	**515.80**	1458.90	2683.23	1380.80	
Leuven1_N25_a2	5475	**4199.31**	4606.62	5572.52		
Leuven1_N25_a3	5027	657.92	**399.05**	3313.07	450.72	
Leuven1_N25_a4	5950		**5636.13**			
Leuven1_N28_a2	5989	1899.33	**846.12**	1568.95	1452.91	
Leuven1_N31_a2	6347	3181.65	**2867.80**		5257.44	
Leuven1_N34_a4	7104	4557.89	**2100.55**		2234.10	
Leuven1_N34_a5	7031	3766.73	**512.31**		2545.37	
Leuven1_N13_b1	3099	0.36	0.35	0.58	**0.35**	293.61
Leuven1_N13_b2	3418	1.16	1.30	245.28	**1.07**	814.98
Leuven1_N13_b3	3138	22.38	**22.15**	154.98	23.03	187.57
Leuven1_N13_b4	3188	129.82	78.85	450.17	89.54	**14.43**
Leuven1_N13_b5	2601	20.59	**1.39**	680.53	27.63	858.40
Leuven1_N16_b1	3318	**4.83**	126.00	201.05	74.65	17.71
Leuven1_N16_b2	3548	83.28	**6.18**	141.96	136.93	1005.92
Leuven1_N16_b3	3568	711.72	**476.76**	644.12	610.30	3724.87
Leuven1_N16_b4	3421	207.17	207.13	268.32	**28.07**	235.30
Leuven1_N16_b5	3384	429.88	512.10	408.92	**363.12**	3404.94

(continued on next page)

Table A9: (continued)

		time(s)				
instance	optimal	*B&C*1	*B&C*2	*B&C*3	*C&B*	*B&B*
Leuven1_N19_b2	3832	563.94	554.58	910.63	**302.51**	
Leuven1_N19_b3	4118	69.03	**35.39**	472.90	111.82	2411.46
Leuven1_N19_b4	4702		**5807.65**			
Leuven1_N19_b5	3774	132.83	**104.13**	144.07	121.65	2272.99
Leuven1_N22_b2	4815	**3641.11**	5203.12	3728.68	5703.66	
Leuven1_N22_b3	4777	5056.78	**2392.12**	5556.28	2956.90	
Leuven1_N22_b4	4348	**6910.20**				
Leuven1_N13_c1	2644	0.56	0.56	1.58	**0.52**	973.93
Leuven1_N13_c2	3314	262.21	226.91	**131.11**	253.49	2472.07
Leuven1_N13_c3	2944	**103.90**	193.34	751.53	113.95	3146.40
Leuven1_N13_c4	2947	203.20	**159.24**	159.77	195.91	2374.15
Leuven1_N13_c5	2597	98.43	98.27	557.73	**90.89**	3079.63
Leuven1_N16_c1	2971	212.43	194.83	**79.28**	130.79	2035.46
Leuven1_N16_c2	3186	839.74	669.77	**224.32**	762.63	2099.17
Leuven1_N16_c3	3481	**151.04**	728.50	649.22	582.51	2484.51
Leuven1_N16_c4	3338	292.52	**127.90**	555.33	139.92	3013.72
Leuven1_N16_c5	3371	872.16	**605.73**	2243.19	729.52	
Leuven1_N19_c2	3726	1618.08	1539.82	4294.73	**895.50**	
Leuven1_N19_c3	3945	**702.17**	1112.19	1749.01	935.56	5456.04
Leuven1_N19_c5	3605	589.15	347.06	2843.66	**306.82**	
Leuven1_N22_c2	4075	1813.01	2073.46	2044.61	**974.06**	

Table A10: Best lower and upper bounds value for unsolved instances.

	best LB		best UB		
instance	value	method	value	method	gap(%)
Leuven1_N25_a5	5698.50	BC1	5818	BC1,BC2,BC3	2.10
Leuven1_N28_a1	6119.50	BC3	6351	BC3	3.78
Leuven1_N28_a3	5674.00	CB	5914	BC1	4.23
Leuven1_N28_a4	6254.50	BC2	6502	BC2	3.96
Leuven1_N28_a5	5848.00	BC1	6115	BC3	4.57
Leuven1_N31_a1	6351.00	CB	6695	BC3	5.42
Leuven1_N31_a3	6678.68	CB	7082	BC2	6.04
Leuven1_N31_a4	6308.94	BC2	6437	BC3	2.03
Leuven1_N31_a5	6256.45	CB	6603	BC3	5.54
Leuven1_N34_a1	6888.58	CB	7300	CB	5.97
Leuven1_N34_a2	6471.82	CB	6569	CB	1.50
Leuven1_N34_a3	6415.06	CB	6637	BC3	3.46
Leuven1_N37_a1	6708.13	CB	6908	BB	2.98
Leuven1_N37_a2	7354.49	CB	7502	BC1	2.01
Leuven1_N37_a3	8093.65	BC1	8318	BC1	2.77
Leuven1_N37_a4	7026.00	CB	7402	BC1	5.35
Leuven1_N37_a5	7852.34	CB	8296	BC1	5.65
Leuven1_N19_b1	4317.49	BC2	4454	BC1,BC2,CB	3.16
Leuven1_N22_b1	4451.04	BC2	4555	BC1,BC2,CB	2.34
Leuven1_N22_b5	4650.17	CB	4921	BC2	5.82
Leuven1_N25_b1	5212.18	CB	5596	BC3	7.36
Leuven1_N25_b2	4990.50	BC3	5509	CB	10.39
Leuven1_N25_b3	4665.91	CB	4868	BC3	4.33
Leuven1_N25_b4	5113.45	BC2	5168	BC1,BC2,BC3,CB	1.07
Leuven1_N25_b5	5176.37	CB	5436	CB	5.02
Leuven1_N28_b1	5582.50	CB	5891	BC3	5.53
Leuven1_N28_b2	5466.77	CB	5815	BC3	6.37
Leuven1_N28_b3	5560.94	CB	6049	BC1	8.78
Leuven1_N28_b4	5431.31	CB	5749	CB	5.85
Leuven1_N28_b5	5032.17	BC1	5770	BC2	14.66
Leuven1_N31_b1	5972.50	CB	6417	BC3	7.44
Leuven1_N31_b2	5950.61	CB	6398	BC1	7.52
Leuven1_N31_b3	6492.83	CB	7168	CB	10.40
Leuven1_N31_b4	5347.41	CB	5699	BC1	6.57
Leuven1_N31_b5	5477.75	CB	6016	CB	9.83
Leuven1_N34_b1	6339.95	CB	7848	CB	23.79
Leuven1_N34_b2	6075.12	CB	6518	BC2	7.29
Leuven1_N34_b3	6048.92	CB	6739	BC2	11.41

(continued on next page)

Table A10: (continued)

instance	best LB value	best LB method	best UB value	best UB method	gap(%)
Leuven1_N34_b4	6359.81	CB	7663	CB	20.49
Leuven1_N34_b5	6118.35	CB	6278	BC2	2.61
Leuven1_N37_b1	6256.70	CB	7506	BC2	19.97
Leuven1_N37_b2	6886.38	CB	7738	CB	12.37
Leuven1_N37_b3	7145.17	CB	7783	CB	8.93
Leuven1_N37_b4	6434.92	CB	7562	CB	17.52
Leuven1_N37_b5	7115.35	CB	8905	CB	25.15
Leuven1_N19_c1	4117.90	BC1	4236	BC1,BC3,CB	2.87
Leuven1_N19_c4	4413.11	CB	4479	BC1,BC3,CB	1.49
Leuven1_N22_c1	3830.43	CB	4313	BC2	12.60
Leuven1_N22_c3	4208.75	CB	4611	CB	9.56
Leuven1_N22_c4	3983.89	BC2	4124	BC1	3.52
Leuven1_N22_c5	4146.48	CB	4448	BC2	7.27
Leuven1_N25_c1	4698.83	BC1	5404	CB	15.01
Leuven1_N25_c2	4583.51	BC2	5407	BB	17.97
Leuven1_N25_c3	4202.04	BC2	4668	BC1	11.09
Leuven1_N25_c4	4748.35	BC1	4943	BC2,BC3	4.10
Leuven1_N25_c5	4536.94	CB	5469	BC1	20.54
Leuven1_N28_c1	5337.42	CB	6392	BC1	19.76
Leuven1_N28_c2	5031.48	CB	5465	CB	8.62
Leuven1_N28_c3	5002.97	CB	6286	CB	25.65
Leuven1_N28_c4	5133.50	CB	6035	CB	17.56
Leuven1_N28_c5	4780.21	CB	5777	CB	20.85
Leuven1_N31_c1	5606.95	CB	6418	CB	14.47
Leuven1_N31_c2	5570.33	CB	6143	CB	10.28
Leuven1_N31_c3	5666.61	CB	6906	CB	21.87
Leuven1_N31_c4	5155.39	CB	6213	CB	20.51
Leuven1_N31_c5	5281.71	CB	6734	CB	27.50
Leuven1_N34_c1	5769.37	CB	9876	BC1	71.18
Leuven1_N34_c2	5351.83	CB	7426	CB	38.76
Leuven1_N34_c3	5496.00	CB	7134	CB	29.80
Leuven1_N34_c4	6008.81	CB	7505	CB	24.90
Leuven1_N34_c5	5905.82	CB	6516	CB	10.33
Leuven1_N37_c1	5639.75	CB	9412	BC2	66.89
Leuven1_N37_c2	6094.25	CB	10453	BC2	71.52
Leuven1_N37_c3	6643.50	CB	10429	CB	56.98
Leuven1_N37_c4	5864.68	CB	10223	BC2	74.31
Leuven1_N37_c5	6444.60	CB	8038	CB	24.72

Table A11: The performance of the VNS algorithm.

instance	optimal	best	gap (%) worst	av	fs
Leuven1_N13_a1	2984	0.00	0.00	0.00	5
Leuven1_N13_a2	3721	0.00	0.00	0.00	5
Leuven1_N13_a3	3133	0.00	0.00	0.00	5
Leuven1_N13_a4	3269	0.00	0.00	0.00	5
Leuven1_N13_a5	2886	0.00	0.00	0.00	5
Leuven1_N16_a1	3248	0.00	0.00	0.00	5
Leuven1_N16_a2	3388	0.00	0.00	0.00	5
Leuven1_N16_a3	3812	0.00	0.00	0.00	5
Leuven1_N16_a4	3682	0.00	0.00	0.00	5
Leuven1_N16_a5	4081	0.00	0.00	0.00	5
Leuven1_N19_a1	4786	0.00	0.00	0.00	5
Leuven1_N19_a2	3865	0.00	0.00	0.00	5
Leuven1_N19_a3	4394	0.00	0.00	0.00	5
Leuven1_N19_a4	4988	0.00	0.00	0.00	5
Leuven1_N19_a5	3675	0.00	0.00	0.00	5
Leuven1_N22_a1	4533	0.00	0.00	0.00	5
Leuven1_N22_a2	4363	0.00	0.00	0.00	5
Leuven1_N22_a3	4777	0.00	0.00	0.00	5
Leuven1_N22_a4	4591	0.00	0.00	0.00	5
Leuven1_N22_a5	4555	0.00	0.00	0.00	5
Leuven1_N25_a1	5324	0.00	0.00	0.00	5
Leuven1_N25_a2	5475	0.00	0.00	0.00	5
Leuven1_N25_a3	5027	0.00	0.00	0.00	5
Leuven1_N25_a4	5950	0.00	0.00	0.00	5
Leuven1_N28_a2	5989	0.00	0.00	0.00	5
Leuven1_N31_a2	6347	0.00	0.00	0.00	5
Leuven1_N34_a4	7104	0.00	0.00	0.00	5
Leuven1_N34_a5	7031	0.00	0.00	0.00	5
Leuven1_N13_b1	3099	0.00	0.00	0.00	5
Leuven1_N13_b2	3418	0.00	0.00	0.00	5
Leuven1_N13_b3	3138	0.00	0.00	0.00	5
Leuven1_N13_b4	3188	0.00	0.00	0.00	5
Leuven1_N13_b5	2601	0.00	0.00	0.00	5
Leuven1_N16_b1	3318	0.00	0.00	0.00	5
Leuven1_N16_b2	3548	0.00	0.00	0.00	5
Leuven1_N16_b3	3568	0.00	0.00	0.00	5
Leuven1_N16_b4	3421	0.00	0.00	0.00	5
Leuven1_N16_b5	3384	0.00	0.00	0.00	5

(continued on next page)

Table A11: (continued)

instance	optimal	gap (%) best	worst	av	fs
Leuven1_N19_b2	3832	0.00	0.00	0.00	5
Leuven1_N19_b3	4118	0.00	0.00	0.00	5
Leuven1_N19_b4	4702	0.00	0.00	0.00	5
Leuven1_N19_b5	3774	0.00	0.00	0.00	5
Leuven1_N22_b2	4815	0.00	0.00	0.00	5
Leuven1_N22_b3	4777	0.00	0.00	0.00	5
Leuven1_N22_b4	4348	0.00	0.00	0.00	5
Leuven1_N13_c1	2644	0.00	0.00	0.00	5
Leuven1_N13_c2	3314	0.00	0.00	0.00	5
Leuven1_N13_c3	2944	0.00	0.00	0.00	5
Leuven1_N13_c4	2947	0.00	0.00	0.00	5
Leuven1_N13_c5	2597	0.00	0.00	0.00	5
Leuven1_N16_c1	2971	0.00	0.00	0.00	5
Leuven1_N16_c2	3186	0.00	0.00	0.00	5
Leuven1_N16_c3	3481	0.00	0.00	0.00	5
Leuven1_N16_c4	3338	0.00	0.00	0.00	5
Leuven1_N16_c5	3371	0.00	0.00	0.00	5
Leuven1_N19_c2	3726	0.00	0.00	0.00	5
Leuven1_N19_c3	3945	0.00	0.00	0.00	5
Leuven1_N19_c5	3605	0.00	0.00	0.00	5
Leuven1_N22_c2	4075	0.00	0.00	0.00	5
average		**0.00**	**0.00**	**0.00**	**295**

Table A12: The performance of the VNS algorithm.

instance	best LB	gap (%) best	gap (%) worst	gap (%) av	best UB	gap (%) best	gap (%) worst	gap (%) av
Leuven1_N25_a5	5698.50	2.10	2.10	2.10	5818	0.00	0.00	0.00
Leuven1_N28_a1	6119.50	3.46	3.46	3.46	6351	-0.31	-0.31	-0.31
Leuven1_N28_a3	5674.00	3.45	3.45	3.45	5914	-0.74	-0.74	-0.74
Leuven1_N28_a4	6254.50	3.61	3.61	3.61	6502	-0.34	-0.34	-0.34
Leuven1_N28_a5	5848.00	3.73	3.73	3.73	6115	-0.80	-0.80	-0.80
Leuven1_N31_a1	6351.00	3.67	3.67	3.67	6695	-1.66	-1.66	-1.66
Leuven1_N31_a3	6678.68	4.54	4.54	4.54	7082	-1.41	-1.41	-1.41
Leuven1_N31_a4	6308.94	2.00	2.00	2.00	6437	-0.03	-0.03	-0.03
Leuven1_N31_a5	6256.45	4.52	4.52	4.52	6603	-0.97	-0.97	-0.97
Leuven1_N34_a1	6888.58	4.11	4.11	4.11	7300	-1.75	-1.75	-1.75
Leuven1_N34_a2	6471.82	1.41	1.41	1.41	6569	-0.09	-0.09	-0.09
Leuven1_N34_a3	6415.06	2.90	2.90	2.90	6637	-0.54	-0.54	-0.54
Leuven1_N37_a1	6708.13	2.44	2.44	2.44	6908	-0.52	-0.52	-0.52
Leuven1_N37_a2	7354.49	1.42	1.42	1.42	7502	-0.57	-0.57	-0.57
Leuven1_N37_a3	8093.65	2.67	2.67	2.67	8318	-0.10	-0.10	-0.10
Leuven1_N37_a4	7026.00	2.72	2.72	2.72	7402	-2.50	-2.50	-2.50
Leuven1_N37_a5	7852.34	3.06	3.06	3.06	8296	-2.45	-2.45	-2.45
Leuven1_N19_b1	4317.49	3.16	3.16	3.16	4454	0.00	0.00	0.00
Leuven1_N22_b1	4451.04	2.34	2.34	2.34	4555	0.00	0.00	0.00
Leuven1_N22_b5	4650.17	4.49	5.82	5.29	4921	-1.26	0.00	-0.50
Leuven1_N25_b1	5212.18	5.10	5.10	5.10	5596	-2.11	-2.11	-2.11
Leuven1_N25_b2	4990.50	6.14	6.14	6.14	5509	-3.85	-3.85	-3.85
Leuven1_N25_b3	4665.91	3.65	3.65	3.65	4868	-0.66	-0.66	-0.66
Leuven1_N25_b4	5113.45	1.07	1.07	1.07	5168	0.00	0.00	0.00
Leuven1_N25_b5	5176.37	4.69	4.69	4.69	5436	-0.31	-0.31	-0.31
Leuven1_N28_b1	5582.50	4.43	4.43	4.43	5891	-1.04	-1.04	-1.04
Leuven1_N28_b2	5466.77	3.92	3.92	3.92	5815	-2.30	-2.30	-2.30
Leuven1_N28_b3	5560.94	4.28	4.28	4.28	6049	-4.13	-4.13	-4.13
Leuven1_N28_b4	5431.31	4.89	4.89	4.89	5749	-0.90	-0.90	-0.90
Leuven1_N28_b5	5032.17	6.85	6.85	6.85	5770	-6.81	-6.81	-6.81
Leuven1_N31_b1	5972.50	5.35	5.35	5.35	6417	-1.95	-1.95	-1.95
Leuven1_N31_b2	5950.61	4.51	4.51	4.51	6398	-2.80	-2.80	-2.80
Leuven1_N31_b3	6492.83	5.12	5.12	5.12	7168	-4.79	-4.79	-4.79
Leuven1_N31_b4	5347.41	4.65	4.65	4.65	5699	-1.81	-1.81	-1.81
Leuven1_N31_b5	5477.75	7.33	7.33	7.33	6016	-2.28	-2.28	-2.28
Leuven1_N34_b1	6339.95	7.29	7.29	7.29	7848	-13.33	-13.33	-13.33
Leuven1_N34_b2	6075.12	3.77	3.77	3.77	6518	-3.28	-3.28	-3.28
Leuven1_N34_b3	6048.92	5.03	5.03	5.03	6739	-5.73	-5.73	-5.73

(continued on next page)

Table A12: (continued)

instance	best LB	gap (%) best	gap (%) worst	gap (%) av	best UB	gap (%) best	gap (%) worst	gap (%) av
Leuven1_N34_b4	6359.81	5.21	5.21	5.21	7663	-12.68	-12.68	-12.68
Leuven1_N34_b5	6118.35	1.99	1.99	1.99	6278	-0.61	-0.61	-0.61
Leuven1_N37_b1	6256.70	5.90	6.13	5.95	7506	-11.72	-11.54	-11.69
Leuven1_N37_b2	6886.38	7.62	7.62	7.62	7738	-4.23	-4.23	-4.23
Leuven1_N37_b3	7145.17	4.10	4.10	4.10	7783	-4.43	-4.43	-4.43
Leuven1_N37_b4	6434.92	7.79	7.79	7.79	7562	-8.28	-8.28	-8.28
Leuven1_N37_b5	7115.35	6.19	6.19	6.19	8905	-15.15	-15.15	-15.15
Leuven1_N19_c1	4117.90	2.87	2.87	2.87	4236	0.00	0.00	0.00
Leuven1_N19_c4	4413.11	1.49	1.49	1.49	4479	0.00	0.00	0.00
Leuven1_N22_c1	3830.43	9.10	9.10	9.10	4313	-3.11	-3.11	-3.11
Leuven1_N22_c3	4208.75	5.76	5.76	5.76	4611	-3.47	-3.47	-3.47
Leuven1_N22_c4	3983.89	3.52	3.52	3.52	4124	0.00	0.00	0.00
Leuven1_N22_c5	4146.48	4.98	4.98	4.98	4448	-2.14	-2.14	-2.14
Leuven1_N25_c1	4698.83	7.69	7.69	7.69	5404	-6.37	-6.37	-6.37
Leuven1_N25_c2	4583.51	8.02	8.02	8.02	5407	-8.43	-8.43	-8.43
Leuven1_N25_c3	4202.04	8.02	8.02	8.02	4668	-2.76	-2.76	-2.76
Leuven1_N25_c4	4748.35	4.10	4.10	4.10	4943	0.00	0.00	0.00
Leuven1_N25_c5	4536.94	7.83	7.83	7.83	5469	-10.55	-10.55	-10.55
Leuven1_N28_c1	5337.42	5.29	5.29	5.29	6392	-12.08	-12.08	-12.08
Leuven1_N28_c2	5031.48	4.66	4.66	4.66	5465	-3.64	-3.64	-3.64
Leuven1_N28_c3	5002.97	6.06	6.06	6.06	6286	-15.59	-15.59	-15.59
Leuven1_N28_c4	5133.50	6.98	6.98	6.98	6035	-9.00	-9.00	-9.00
Leuven1_N28_c5	4780.21	5.89	5.89	5.89	5777	-12.38	-12.38	-12.38
Leuven1_N31_c1	5606.95	6.08	6.46	6.16	6418	-7.32	-7.00	-7.26
Leuven1_N31_c2	5570.33	6.19	6.19	6.19	6143	-3.71	-3.71	-3.71
Leuven1_N31_c3	5666.61	10.82	10.82	10.82	6906	-9.06	-9.06	-9.06
Leuven1_N31_c4	5155.39	5.71	5.71	5.71	6213	-12.28	-12.28	-12.28
Leuven1_N31_c5	5281.71	8.09	8.09	8.09	6734	-15.22	-15.22	-15.22
Leuven1_N34_c1	5769.37	11.68	11.68	11.68	9876	-34.76	-34.76	-34.76
Leuven1_N34_c2	5351.83	7.22	7.22	7.22	7426	-22.73	-22.73	-22.73
Leuven1_N34_c3	5496.00	5.48	5.48	5.48	7134	-18.74	-18.74	-18.74
Leuven1_N34_c4	6008.81	8.22	8.87	8.61	7505	-13.35	-12.83	-13.04
Leuven1_N34_c5	5905.82	4.46	4.46	4.46	6516	-5.33	-5.33	-5.33
Leuven1_N37_c1	5639.75	8.04	8.04	8.04	9412	-35.26	-35.26	-35.26
Leuven1_N37_c2	6094.25	14.06	14.06	14.06	10453	-33.50	-33.50	-33.50
Leuven1_N37_c3	6643.50	10.56	10.56	10.56	10429	-29.57	-29.57	-29.57
Leuven1_N37_c4	5864.68	10.34	11.17	10.67	10223	-36.70	-36.22	-36.51
Leuven1_N37_c5	6444.60	11.29	11.71	11.45	8038	-10.77	-10.44	-10.64
average		**5.38**	**5.43**	**5.41**		**-6.70**	**-6.66**	**-6.68**

Table A13: The result of solving $G1$ instances of MTVRP.

name	m	T_H	optimal	best	gap(%) worst	av	opt	fs
CMT1	1	551	524.61	0.00	0.00	0.00	5	5
	2	275	533.00	0.00	1.89	1.37	1	5
	1	577	524.61	0.00	0.00	0.00	5	5
	2	289	529.85	0.00	0.18	0.07	3	5
	4	144	546.29	NF	NF	NF	0	0
CMT2	1	877	835.26	0.00	0.62	0.24	2	5
	2	439	835.26	0.00	1.10	0.23	3	5
	3	292	835.26	0.00	0.67	0.26	3	5
	4	219	835.26	0.00	0.94	0.30	3	5
	5	175	835.80	0.12	1.85	0.72	0	5
	1	919	835.26	0.00	0.54	0.19	3	5
	2	459	835.26	0.00	0.75	0.37	1	5
	3	306	835.26	0.00	0.67	0.35	2	5
	4	230	835.26	0.00	1.19	0.26	3	5
	5	184	835.26	0.40	1.02	0.69	0	5
	6	153	839.22	0.46	1.54	1.05	0	5
CMT3	1	867	826.14	0.15	0.59	0.24	0	5
	2	434	826.14	0.15	0.71	0.31	0	5
	3	289	826.14	0.00	0.43	0.23	1	5
	1	909	826.14	0.15	0.40	0.25	0	5
	2	454	826.14	0.15	0.67	0.36	0	5
	3	303	826.14	0.15	0.59	0.40	0	5
	4	227	826.14	0.15	0.41	0.20	0	5
CMT11	1	1094	1042.11	0.00	0.00	0.00	5	5
	2	547	1042.11	0.00	0.00	0.00	5	5
	3	365	1042.11	0.00	0.00	0.00	5	5
	5	219	1042.11	0.00	0.00	0.00	5	5
	1	1146	1042.11	0.00	0.00	0.00	5	5
	2	573	1042.11	0.00	0.00	0.00	5	5
	3	382	1042.11	0.00	0.00	0.00	5	5
	4	287	1042.11	0.00	0.00	0.00	5	5
	5	229	1042.11	0.00	0.00	0.00	5	5
CMT12	1	861	819.56	0.00	0.00	0.00	5	5
	2	430	819.56	0.00	0.00	0.00	5	5
	3	287	819.56	0.00	0.00	0.00	5	5
	4	215	819.56	0.00	0.00	0.00	5	5
	1	902	819.56	0.00	0.00	0.00	5	5
	2	451	819.56	0.00	0.00	0.00	5	5

(continued on next page)

Table A13: (continued)

name	m	T_H	optimal	best	gap(%) worst	av	opt	fs
CMT12	3	301	819.56	0.00	0.00	0.00	5	5
	4	225	819.56	0.00	0.00	0.00	5	5
	5	180	824.78	0.00	0.00	0.00	5	5
	6	150	823.14	0.00	0.00	0.00	5	5
average				**0.05**	**0.41**	**0.20**		
total							**130**	**205**

'NF' implies that no feasible solution was found within the time limit.

Table A14: The result of solving $G2$ instances of MTVRP.

name	m	T_H	best UB	gap(%) best	worst	av	fs
CMT1	3	192	552.68	0.00	0.00	0.00	5
CMT2	6	146	855.34	0.26	0.26	0.26	1
	7	131	844.55	1.42	2.13	1.71	4
CMT3	4	217	829.45	0.00	0.29	0.11	5
	5	173	832.89	0.00	0.00	0.00	1
	6	145	836.22	NF	NF	NF	0
	5	182	831.20	0.15	0.80	0.46	5
	6	151	834.35	0.07	0.62	0.20	5
CMT4	1	1080	1031.00	**-0.25**	1.20	0.63	5
	2	540	1031.07	0.23	1.50	0.69	5
	3	360	1028.42	0.35	0.77	0.63	5
	4	270	1031.10	1.17	1.93	1.45	5
	5	216	1029.65	**-0.05**	1.51	0.81	5
	6	180	1034.61	0.64	1.61	1.11	5
	7	154	1067.10	NF	NF	NF	0
	8	135	1056.54	NF	NF	NF	0
	1	1131	1031.07	**-0.14**	1.10	0.42	5
	2	566	1030.45	0.10	1.37	0.62	5
	3	377	1031.59	0.44	1.10	0.83	5
	4	283	1031.07	**-0.22**	1.85	0.53	5
	5	226	1030.86	**-0.12**	1.26	0.82	5
	6	189	1030.45	0.22	1.36	0.78	5
	7	162	1032.07	0.64	1.54	1.19	5
	8	141	1044.32	0.18	0.18	0.18	1
CMT5	1	1356	1298.35	1.74	3.14	2.41	5
	2	678	1302.15	0.83	1.73	1.33	5
	3	452	1301.29	1.19	2.63	1.85	5
	4	339	1299.70	1.20	3.14	2.01	5
	5	271	1300.02	1.53	2.64	2.11	5
	6	226	1303.37	0.19	2.14	1.28	5
	7	194	1304.02	1.07	1.85	1.57	4
	8	170	1303.11	1.44	2.34	1.78	3
	9	151	1307.93	0.58	1.66	1.12	2
	10	136	1315.47	NF	NF	NF	0
	1	1421	1299.86	1.07	2.48	1.70	5
	2	710	1305.35	0.43	1.74	1.26	5
	3	474	1301.03	1.45	2.40	1.78	5
	4	355	1303.65	1.22	2.19	1.69	5

(continued on next page)

Table A14: (continued)

name	m	T_H	best UB	gap(%) best	worst	av	fs
CMT5	5	284	1300.62	1.19	2.20	1.55	5
	6	237	1306.17	0.99	2.31	1.47	5
	7	203	1301.54	0.63	2.21	1.36	5
	8	178	1308.78	0.56	1.03	0.82	5
	9	158	1304.28	1.19	1.89	1.52	5
	10	142	1305.01	0.80	5.75	2.73	4
CMT11	4	274	1078.64	0.01	0.01	0.01	1
CMT12	5	172	845.37	NF	NF	NF	0
F11	1	254	241.97	0.00	0.00	0.00	5
	2	127	250.85	0.00	0.00	0.00	5
	1	266	241.97	0.00	0.00	0.00	5
	2	133	241.97	0.00	0.00	0.00	5
	3	89	254.07	0.00	0.00	0.00	5
F12	1	1221	1162.96	0.00	0.30	0.23	5
	2	611	1162.96	0.00	0.38	0.08	5
	3	407	1162.96	0.00	0.00	0.00	5
	1	1279	1162.96	0.00	0.11	0.02	5
	2	640	1162.96	0.00	0.31	0.07	5
	3	426	1162.96	0.00	0.26	0.05	5
average				**0.47**	**1.33**	**0.87**	
total							**236**

'NF' implies that no feasible solution was found within the time limit.

Table A15: New feasible solutions for some MTVRP instances.

name	v	t	τ_t	l_t	
CMT4	1	1	128.71	200	0-46-57-23-69-7-61-114-99-43-86-97-24-96-14-68-0
1080		2	85.46	196	0-77-81-60-8-31-82-140-113-26-112-48-138-27-0
		3	113.82	197	0-108-44-107-65-93-88-40-136-13-67-134-55-47-0
		4	80.69	197	0-56-146-4-149-111-66-41-94-19-64-42-92-137-147-17-0
		5	57.32	199	0-63-145-142-148-87-150-141-135-143-109-144-0
		6	97.05	200	0-38-9-104-30-105-75-117-89-39-54-10-49-76-0
		7	89.82	197	0-90-71-123-122-124-125-106-73-33-72-91-45-15-52-37-0
		8	77.47	199	0-32-51-22-101-3-59-20-131-83-2-100-11-0
		9	77.33	199	0-139-18-110-133-25-95-58-98-132-6-102-0
		10	21.32	64	0-5-103-12-0
		11	123.68	199	0-129-29-128-84-35-85-36-115-121-116-28-70-80-120-1-119-0
		12	75.76	188	0-78-126-16-127-53-21-79-74-34-130-50-118-62-0
CMT4	1	1	128.71	200	0-46-57-23-69-7-61-114-99-43-86-97-24-96-14-68-0
216		2	85.46	196	0-27-138-48-112-26-113-140-82-31-8-60-81-77-0
	2	1	123.68	199	0-129-29-128-84-35-85-36-115-121-116-28-70-80-120-1-119-0
		2	89.82	197	0-37-52-15-45-91-72-33-73-106-125-124-122-123-71-90-0
	3	1	113.82	197	0-108-44-107-65-93-88-40-136-13-67-134-55-47-0
		2	97.05	200	0-76-49-10-54-39-89-117-75-105-30-104-9-38-0
	4	1	82.94	200	0-63-137-92-42-64-19-94-41-66-111-148-142-147-17-0
		2	77.47	199	0-11-100-2-83-131-20-59-3-101-22-51-32-0
		3	21.32	64	0-5-103-12-0
	5	1	77.33	199	0-139-18-110-133-25-95-58-98-132-6-102-0
		2	75.76	188	0-62-118-50-130-34-74-79-21-53-127-16-126-78-0
		3	55.80	196	0-144-145-109-87-150-141-135-143-4-149-146-56-0
CMT4	1	1	128.71	200	0-46-57-23-69-7-61-114-99-43-86-97-24-96-14-68-0
1131		2	123.68	199	0-129-29-128-84-35-85-36-115-121-116-28-70-80-120-1-119-0
		3	112.86	188	0-37-44-107-65-93-88-40-136-13-67-134-55-47-0
		4	111.07	198	0-90-10-54-106-73-117-89-39-75-105-30-104-38-0
		5	85.46	196	0-27-138-48-112-26-113-140-82-31-8-60-81-77-0
		6	80.69	197	0-17-147-137-92-42-64-19-94-41-66-111-4-149-146-56-0
		7	78.44	199	0-78-126-16-127-53-21-79-74-34-118-130-50-9-62-0
		8	77.47	199	0-11-100-2-83-131-20-59-3-101-22-51-32-0
		9	77.33	199	0-139-18-110-133-25-95-58-98-132-6-102-0
		10	75.30	197	0-108-52-15-45-91-72-33-125-124-122-123-71-49-76-0
		11	57.32	199	0-63-145-142-87-148-150-141-135-143-109-144-0
		12	21.32	64	0-12-103-5-0

(continued on next page)

Table A15: (continued)

name	v	t	τ_t	l_t	
CMT4	1	1	97.05	200	0-76-49-10-54-39-89-117-75-105-30-104-9-38-0
283		2	59.35	199	0-146-109-143-135-141-150-148-87-142-147-145-144-0
		3	89.82	197	0-37-52-15-45-91-72-33-73-106-125-124-122-123-71-90-0
		4	21.32	64	0-12-103-5-0
	2	1	113.82	197	0-108-44-107-65-93-88-40-136-13-67-134-55-47-0
		2	85.46	196	0-27-138-48-112-26-113-140-82-31-8-60-81-77-0
		3	75.76	188	0-78-126-16-127-53-21-79-74-34-130-50-118-62-0
	3	1	123.68	199	0-129-29-128-84-35-85-36-115-121-116-28-70-80-120-1-119-0
		2	77.33	199	0-102-6-132-98-58-95-25-133-110-18-139-0
		3	79.01	197	0-56-149-4-111-66-41-94-19-64-42-92-137-17-63-0
	4	1	128.71	200	0-46-57-23-69-7-61-114-99-43-86-97-24-96-14-68-0
		2	77.47	199	0-32-51-22-101-3-59-20-131-83-2-100-11-0
CMT4	1	1	128.71	200	0-46-57-23-69-7-61-114-99-43-86-97-24-96-14-68-0
226		2	85.46	196	0-27-138-48-112-26-113-140-82-31-8-60-81-77-0
	2	1	123.68	199	0-129-29-128-84-35-85-36-115-121-116-28-70-80-120-1-119-0
		2	80.69	197	0-56-146-4-149-111-66-41-94-19-64-42-92-137-147-17-0
		3	21.32	64	0-5-103-12-0
	3	1	112.86	188	0-37-44-107-65-93-88-40-136-13-67-134-55-47-0
		2	111.07	198	0-38-104-30-105-75-39-89-117-73-106-54-10-90-0
	4	1	78.44	199	0-62-9-130-50-118-34-74-79-21-53-127-16-126-78-0
		2	77.47	199	0-11-100-2-83-131-20-59-3-101-22-51-32-0
		3	57.32	199	0-63-145-142-148-87-150-141-135-143-109-144-0
	5	1	77.33	199	0-102-6-132-98-58-95-25-133-110-18-139-0
		2	75.30	197	0-108-52-15-45-91-72-33-125-124-122-123-71-49-76-0

Table A16: The result of solving large-size instances with $Q_P = 20$ and $T_P = 1800$.

name	$W_P = 100$ $TPRP_1$	$TPRP_2$	$TPRP_3$	$W_P = 1000$ $TPRP_1$	$TPRP_2$	$TPRP_3$
Leuven1_R1_a1_a5	17955	20399	18076	22219	23638	22793
Leuven1_R1_b1_b5	16929	21311	17259	21333	23834	22849
Leuven1_R1_c1_c5	15669	22185	15989	20429	23373	22456
Leuven1_C1_a1_a5	17316	19668	17462	21747	23087	22648
Leuven1_C1_b1_b5	16360	20486	16591	20889	23186	22709
Leuven1_C1_c1_c5	15249	21010	15620	20117	22631	22638
Leuven1_C2_a1_a5	16599	19470	16704	20662	22364	21053
Leuven1_C2_b1_b5	15477	20385	15730	19834	22545	21228
Leuven1_C2_c1_c5	14001	20904	14161	19064	21804	20479
Leuven1_C3_a1_a5	15373	17983	15442	19077	20697	19673
Leuven1_C3_b1_b5	14297	19679	14529	18653	21479	19457
Leuven1_C3_c1_c5	12990	20438	13301	17586	21338	18975
Leuven1_R1_a6_a10	33774	38056	33928	41812	43497	43015
Leuven1_R1_b6_b10	31931	40360	32114	40560	44274	43012
Leuven1_R1_c6_c10	29808	42168	30113	39471	43972	42715
Leuven1_C1_a6_a10	33795	38298	33925	41996	44081	42973
Leuven1_C1_b6_b10	32069	40245	32353	40802	44203	43547
Leuven1_C1_c6_c10	29651	41756	30078	39093	43630	42920
Leuven1_C2_a6_a10	30948	36215	31001	38367	41013	38885
Leuven1_C2_b6_b10	28387	38295	28497	36756	41715	38133
Leuven1_C2_c6_c10	26122	40130	26348	35429	41930	37448
Leuven1_C3_a6_a10	30622	36005	30751	37817	40734	38322
Leuven1_C3_b6_b10	28764	39046	28842	36992	42652	38251
Leuven1_C3_c6_c10	26227	40869	26330	35575	42675	37628
Leuven1_R1_a11_a15	49582	56258	49556	61509	63994	62616
Leuven1_R1_b11_b15	46600	59761	46732	59944	65256	62320
Leuven1_R1_c11_c15	43340	62624	43462	58299	65326	61991
Leuven1_C1_a11_a15	49399	56086	49443	62082	64294	63128
Leuven1_C1_b11_b15	46404	59322	46525	59753	65000	62961
Leuven1_C1_c11_c15	42963	62007	43204	57429	64714	62410
Leuven1_C2_a11_a15	46885	54370	46868	58331	61861	59040
Leuven1_C2_b11_b15	43324	57896	43332	55984	63113	58315
Leuven1_C2_c11_c15	39744	60728	39781	54210	63429	57447
Leuven1_C3_a11_a15	45771	53085	45789	57264	60391	57699
Leuven1_C3_b11_b15	42321	57000	42415	55590	61780	57350
Leuven1_C3_c11_c15	38864	60399	38945	53523	63121	56542

Table A17: The result of solving large-size instances with $Q_P = 20$ and $T_P = 3600$.

name	$W_P = 100$ $TPRP_1$	$TPRP_2$	$TPRP_3$	$W_P = 1000$ $TPRP_1$	$TPRP_2$	$TPRP_3$
Leuven1_R1_a1_a5	17670	20138	17704	19950	21939	20406
Leuven1_R1_b1_b5	16535	21115	16798	19140	22733	19505
Leuven1_R1_c1_c5	15203	22102	15422	17865	22930	19020
Leuven1_C1_a1_a5	17055	19475	17174	19414	21275	19874
Leuven1_C1_b1_b5	15990	20362	16249	18536	21982	19169
Leuven1_C1_c1_c5	14910	20930	15232	17688	21830	18724
Leuven1_C2_a1_a5	16359	19352	16434	18398	21152	18954
Leuven1_C2_b1_b5	15247	20198	15382	17532	21458	18083
Leuven1_C2_c1_c5	13650	20904	13771	16367	21804	16838
Leuven1_C3_a1_a5	15074	17887	15137	16876	19687	16933
Leuven1_C3_b1_b5	14017	19571	14217	16126	20471	16919
Leuven1_C3_c1_c5	12647	20438	12900	15221	21338	15599
Leuven1_R1_a6_a10	33180	37722	33313	37216	40408	37816
Leuven1_R1_b6_b10	31279	40088	31437	35816	41889	36849
Leuven1_R1_c6_c10	28993	42049	29280	34282	42948	35566
Leuven1_C1_a6_a10	33230	37904	33373	37381	40613	37896
Leuven1_C1_b6_b10	31448	39997	31684	36028	41974	37111
Leuven1_C1_c6_c10	28974	41635	29257	34211	42535	35567
Leuven1_C2_a6_a10	30424	35924	30478	34062	38625	34269
Leuven1_C2_b6_b10	27763	38115	27863	31900	39908	32362
Leuven1_C2_c6_c10	25374	40030	25617	30150	40930	31048
Leuven1_C3_a6_a10	30134	35641	30202	33702	38343	33985
Leuven1_C3_b6_b10	28102	38805	28173	32353	40605	32676
Leuven1_C3_c6_c10	25487	40768	25583	30110	41668	30791
Leuven1_R1_a11_a15	48606	55688	48677	54489	59298	54970
Leuven1_R1_b11_b15	45497	59388	45613	52371	62087	52812
Leuven1_R1_c11_c15	42110	62516	42196	49941	64316	51055
Leuven1_C1_a11_a15	48553	55541	48595	54716	59681	55067
Leuven1_C1_b11_b15	45436	58971	45552	52501	61682	53416
Leuven1_C1_c11_c15	41838	61888	42035	49952	63685	51075
Leuven1_C2_a11_a15	46109	53873	46063	51687	57476	51659
Leuven1_C2_b11_b15	42353	57561	42352	48741	60269	49190
Leuven1_C2_c11_c15	38611	60608	38602	46076	62228	46923
Leuven1_C3_a11_a15	44902	52569	44962	50295	56177	50393
Leuven1_C3_b11_b15	41337	56679	41424	47623	59375	47939
Leuven1_C3_c11_c15	37742	60278	37753	44823	62084	45644

Table A18: The result of solving large-size instances with $Q_P = 40$ and $T_P = 1800$.

name	$W_P = 100$ $TPRP_1$	$TPRP_2$	$TPRP_3$	$W_P = 1000$ $TPRP_1$	$TPRP_2$	$TPRP_3$
Leuven1_R1_a1_a5	17777	20363	17895	21481	23600	22573
Leuven1_R1_b1_b5	16617	21222	16983	20637	23564	22386
Leuven1_R1_c1_c5	15353	22114	15698	19751	23374	21946
Leuven1_C1_a1_a5	16617	19264	16737	20553	21965	21236
Leuven1_C1_b1_b5	15637	20339	15768	19928	22859	20968
Leuven1_C1_c1_c5	14375	20894	14574	19037	22334	20372
Leuven1_C2_a1_a5	16014	19152	16054	19679	21852	20014
Leuven1_C2_b1_b5	14638	20206	14785	18436	22186	19465
Leuven1_C2_c1_c5	12934	20721	12999	17644	21621	18203
Leuven1_C3_a1_a5	14773	17657	14818	17690	19997	18419
Leuven1_C3_b1_b5	13510	19563	13521	17063	21363	17841
Leuven1_C3_c1_c5	11757	20375	11832	15925	21275	16513
Leuven1_R1_a6_a10	32544	37502	32898	39583	42585	41009
Leuven1_R1_b6_b10	30443	40003	30613	38070	43614	40038
Leuven1_R1_c6_c10	28125	41924	28425	36778	43725	39403
Leuven1_C1_a6_a10	32550	37701	32588	39766	42739	40386
Leuven1_C1_b6_b10	30433	39985	30530	38291	43780	39896
Leuven1_C1_c6_c10	27656	41589	27902	36630	43389	38556
Leuven1_C2_a6_a10	29561	35616	29597	35750	40104	36252
Leuven1_C2_b6_b10	26619	37836	26569	33754	40896	34305
Leuven1_C2_c6_c10	23871	39842	23975	32015	41642	33341
Leuven1_C3_a6_a10	29287	35301	29418	35529	39614	36077
Leuven1_C3_b6_b10	27103	38665	27130	34130	41910	35238
Leuven1_C3_c6_c10	23883	40715	24070	32173	42515	33194
Leuven1_R1_a11_a15	47656	55434	47771	57770	62624	58883
Leuven1_R1_b11_b15	44406	59268	44521	56023	64259	58064
Leuven1_R1_c11_c15	40533	62333	40857	53621	65033	56820
Leuven1_C1_a11_a15	46999	54849	47015	57185	62049	57936
Leuven1_C1_b11_b15	43438	58746	43420	55185	63829	56383
Leuven1_C1_c11_c15	39440	61744	39623	52638	64446	55015
Leuven1_C2_a11_a15	44578	53132	44485	54009	59641	54585
Leuven1_C2_b11_b15	40300	57018	40267	51270	61514	52156
Leuven1_C2_c11_c15	35941	60148	36050	48696	62128	49913
Leuven1_C3_a11_a15	43273	51788	43304	52154	57908	52919
Leuven1_C3_b11_b15	39128	56130	39134	49763	60456	50465
Leuven1_C3_c11_c15	35030	60000	35128	47078	62519	48451

Table A19: The result of solving large-size instances with $Q_P = 40$ and $T_P = 3600$.

name	$W_P = 100$			$W_P = 1000$		
	$TPRP_1$	$TPRP_2$	$TPRP_3$	$TPRP_1$	$TPRP_2$	$TPRP_3$
Leuven1_R1_a1_a5	16821	19780	16816	18870	21580	19516
Leuven1_R1_b1_b5	15478	20834	15616	18089	21914	18313
Leuven1_R1_c1_c5	13882	22031	13953	16600	22931	17071
Leuven1_C1_a1_a5	15531	18668	15557	17668	20468	18108
Leuven1_C1_b1_b5	14283	19994	14346	16937	21434	17046
Leuven1_C1_c1_c5	12814	20748	12827	15680	21648	15745
Leuven1_C2_a1_a5	15232	18659	15259	17120	20463	17274
Leuven1_C2_b1_b5	13842	19817	13926	16046	20717	16446
Leuven1_C2_c1_c5	11958	20721	12004	14709	21621	14739
Leuven1_C3_a1_a5	14145	17370	14144	15947	18810	15968
Leuven1_C3_b1_b5	12871	19250	12942	14738	20150	14938
Leuven1_C3_c1_c5	11239	20383	11241	13767	21283	13923
Leuven1_R1_a6_a10	31033	36605	31118	34841	39315	35157
Leuven1_R1_b6_b10	28564	39438	28594	33181	41239	33284
Leuven1_R1_c6_c10	25638	41787	25754	31084	42687	31305
Leuven1_C1_a6_a10	30547	36484	30560	34606	39185	34651
Leuven1_C1_b6_b10	27985	39225	28069	32856	41026	32911
Leuven1_C1_c6_c10	24726	41401	24790	30442	42301	30580
Leuven1_C2_a6_a10	28480	34855	28512	31805	37189	32125
Leuven1_C2_b6_b10	25235	37438	25192	29098	39238	29183
Leuven1_C2_c6_c10	22151	39678	22127	26899	40578	26851
Leuven1_C3_a6_a10	28240	34656	28218	31676	36996	31640
Leuven1_C3_b6_b10	25570	38191	25508	29349	39991	29484
Leuven1_C3_c6_c10	22253	40494	22232	26814	41394	26903
Leuven1_R1_a11_a15	45221	54005	45257	50873	57620	50859
Leuven1_R1_b11_b15	41266	58298	41215	48117	60998	48196
Leuven1_R1_c11_c15	36771	62020	36720	44834	63820	44946
Leuven1_C1_a11_a15	44293	53241	44334	50178	56999	50167
Leuven1_C1_b11_b15	40274	57697	40242	47357	60388	47378
Leuven1_C1_c11_c15	35651	61403	35686	44000	63203	44060
Leuven1_C2_a11_a15	42724	52004	42650	47585	55444	47553
Leuven1_C2_b11_b15	38075	56489	37994	44013	59188	43844
Leuven1_C2_c11_c15	33300	60003	33218	40367	60904	40229
Leuven1_C3_a11_a15	41777	50861	41785	46565	53716	46465
Leuven1_C3_b11_b15	37393	55619	37334	43041	58156	43070
Leuven1_C3_c11_c15	32746	59701	32679	39559	60959	39583

Mathematical formulation for the TPRPS

The model uses six types of decision variables. The truck binary variable $\bar{x}_{ij}$ takes value 1 if and only if the truck traverses arc $(i,j) \in A$, otherwise its equal to zero. The load's size and weight after leaving node i to node j, where $i,j \in V$, by the truck is denoted by $\bar{f}^v_{ij}$ and $\bar{f}^w_{ij}$ respectively. The porter binary variable x^k_{ij} takes value 1 if and only if porter $k \in M$ traverses arc $(i,j) \in A$, otherwise its equal to zero. The load's size and weight carried by porter $k \in M$ when moving from node i to node j, where $i,j \in S \cup C$, is represented by f^{vk}_{ij} and f^{wk}_{ij} respectively. Thus, the TPRPS can be formulated as follows:

$$\min \quad \sum_{i \in V} \sum_{j \in V} \bar{c}_{ij} \bar{x}_{ij} + \sum_{k \in M} \sum_{i \in S \cup C} \sum_{j \in S \cup C} c_{ij} x^k_{ij} \tag{43}$$

$$s.t. \quad \sum_{i \in V} \bar{x}_{ij} + \sum_{k \in M} \sum_{i \in S \cup C} x^k_{ij} = 1, \quad \forall j \in C \tag{44}$$

$$\sum_{i \in V} \bar{x}_{iz} - \sum_{j \in V} \bar{x}_{zj} = 0, \quad \forall z \in V \tag{45}$$

$$\sum_{i \in S \cup C} x^k_{iz} - \sum_{j \in S \cup C} x^k_{zj} = 0, \quad \forall z \in C, \ \forall k \in M \tag{46}$$

$$\sum_{s \in S} \sum_{i \in C} x^k_{is} - \sum_{s \in S} \sum_{j \in C} x^k_{sj} = 0, \quad \forall k \in M \tag{47}$$

$$\sum_{j \in V} \bar{x}_{sj} \geq \min \left(1; \sum_{k \in M} \sum_{j \in C} x^k_{sj} \right), \quad \forall s \in S \tag{48}$$

$$\sum_{i \in C} x^k_{is} - \sum_{j \in C} x^k_{sj} \leq 1, \quad \forall s \in S, \ \forall k \in M \tag{49}$$

$$\sum_{i \in S \cup C} \sum_{j \in S \cup C} t'_{ij} x^k_{ij} \leq T_P, \quad \forall k \in M \tag{50}$$

$$\sum_{i \in S \cup C} \sum_{j \in S \cup C} c_{ij} x^k_{ij} \leq \sum_{i \in S \cup C} \sum_{j \in S \cup C} c_{ij} x^{k-1}_{ij}, \quad \forall k \in M \backslash \{1\} \tag{51}$$

$$\sum_{i \in V} \left(\bar{f}^v_{ij} - \bar{f}^v_{ji} \right) + \sum_{k \in M} \left(\sum_{i \in S \cup C} \left(f^{vk}_{ij} - f^{vk}_{ji} \right) \right) = q^v_j, \quad \forall j \in C \tag{52}$$

$$\sum_{i \in V} \left(\bar{f}^w_{ij} - \bar{f}^w_{ji} \right) + \sum_{k \in M} \left(\sum_{i \in S \cup C} \left(f^{wk}_{ij} - f^{wk}_{ji} \right) \right) = q^w_j, \quad \forall j \in C \tag{53}$$

$$\sum_{i \in V} \bar{f}^v_{is} - \sum_{j \in V} \bar{f}^v_{sj} = \sum_{k \in M} \sum_{j \in C} f^{vk}_{sj}, \quad \forall s \in S \tag{54}$$

$$\sum_{i \in V} \bar{f}_{is}^{w} - \sum_{j \in V} \bar{f}_{sj}^{w} = \sum_{k \in M} \sum_{j \in C} f_{sj}^{wk}, \quad \forall s \in S \tag{55}$$

$$\sum_{i \in V} \bar{f}_{is}^{v} - \sum_{j \in V} \bar{f}_{sj}^{v} \leq S^{v}, \quad \forall s \in S \tag{56}$$

$$q_j^v \bar{x}_{ij} \leq \bar{f}_{ij}^{v} \leq (Q_T^v - q_i^v)\bar{x}_{ij}, \quad \forall i, j \in V \tag{57}$$

$$q_j^w \bar{x}_{ij} \leq \bar{f}_{ij}^{w} \leq (Q_T^w - q_i^w)\bar{x}_{ij}, \quad \forall i, j \in V \tag{58}$$

$$q_j^v x_{ij}^k \leq f_{ij}^{vk} \leq (Q_P^v - q_i^v) x_{ij}^k, \quad \forall i, j \in S \cup C, \ \forall k \in M \tag{59}$$

$$q_j^w x_{ij}^k \leq f_{ij}^{wk} \leq (Q_P^w - q_i^w) x_{ij}^k, \quad \forall i, j \in S \cup C, \ \forall k \in M \tag{60}$$

$$\bar{x}_{ij} \in \{0, 1\}, \quad \forall i, j \in V, \ i \neq j \tag{61}$$

$$x_{ij}^k \in \{0, 1\}, \quad \forall k \in M, \ \forall i, j \in S \cup C, \ i \neq j \tag{62}$$

$$\bar{f}_{ij}^{v}, \bar{f}_{ij}^{w} \geq 0, \quad \forall i, j \in V, \ i \neq j \tag{63}$$

$$f_{ij}^{vk}, f_{ij}^{wk} \geq 0, \quad \forall s \in S, \ \forall k \in M, \ \forall i, j \in S \cup C, \ i \neq j \tag{64}$$

The objective function minimises the total distribution cost. Constraints (44) ensure that every customer is visited exactly once. Constraints (45) guarantee that entering and leaving truck's arcs to each node are equal. Whereas constraints (46) guarantee that entering and leaving arcs to each customer by each porter are equal. Constraints (47) ensure that the number of ingoing and outgoing arcs to the set of satellites are equal. Constraints (48) are logical constraints. They impose the truck to visit each satellite that has been used by any porter. Constraints (49) guarantee that the difference between the number of entering and leaving arcs from each satellite by each porter is less than or equal to one. Each porter cannot travel more than the pre-set time limit with constraints (50). Constraints (51) are the symmetry breaking constraints, they ensure that the first porter travel at least as much as porter the second porter. Constraints (52) and (53) ensure that the demand (size and weight) of each customer is met. With constraints (54) and (55), the amount of demands (size and weight) delivered to each satellite by the truck is equal to the amount of demands collected from that satellite by porters. The storage capacity of satellites cannot be violated with constraints (56). Constraints (57) – (60) ensure that the load after visiting a node is equal to the load before minus the demand of the respective node. Constraints (61)–(64) define the variable domains.