Ambulance routing problems with rich constraints and multiple objectives [PhD thesis]

Thesis · July 2009

CITATIONS
15

READS
1,190

1 author:

Sophie N. Parragh
Johannes Kepler University Linz
60 PUBLICATIONS **2,836** CITATIONS

SEE PROFILE

universität wien

DISSERTATION

Titel der Dissertation

Ambulance Routing Problems with Rich Constraints and Multiple Objectives

Verfasserin

Mag. Sophie Parragh

angestrebter akademischer Grad

Doctor of Philosophy (PhD)

Wien im Mai, 2009

Studienkennzahl lt. Studienblatt	A 094 146
Dissertationsgebiet lt. Studienblatt	Management
Betreuer	Univ. Doz. Dr. Karl F. Dörner

Acknowledgements

First of all, I would like to thank Univ. Doz. Dr. Karl F. Doerner for introducing me to the field of dial-a-ride problems and for supervising and supporting my work on this book with numerous ideas, extensive discussions, and constructive feedback. Then, I would like to thank Prof. Dr. Richard F. Hartl for providing ideas and feedback, supporting and supervising my work, and also for granting my PhD research position (Austrian Science Fund (FWF) grant #L286-N04). Furthermore, I would like to thank a.o. Prof. Dr. Walter J. Gutjahr for directing me to the Chair for Production and Operations Management. I also wish to thank Prof. Jean-François Cordeau for his input and feedback during my work on Chapters 5 and 6 and for enabling my research visit at the CIRRELT in Montréal. Thanks are also due to my great colleagues at the Chair for Production and Operations Management.

Finally, I would like to thank my family and friends, especially my parents for all their support throughout my life; and last but not least, dear Georg, thank you for your endless patience, encouragement, and support during my work on this book!

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

Humanitarian non-profit ambulance dispatching organizations are committed to tap the full potential of possible cost reduction in order to decrease their expenses. Taking the Austrian Red Cross (ARC), Austria's major ambulance dispatcher, as an example the following figures are available. In 2006 more than two million transportation requests, including both emergency and regular patient transports, were answered by the vehicle fleet of the ARC. Regular requests, which are known well ahead of the planning period, amount to about 80% of the total number of transports (Österreichisches Rotes Kreuz, 2006). While in the context of emergency transportation cost reduction cannot be achieved by means of combined passenger routes, this can be done when dealing with regular patients. In many companies requests are still assigned to vehicles by a dispatcher by hand, leading to routing plans of varying quality, depending on the dispatcher's knowledge of the region and his/her experience in dispatching. This situation calls for a decision support system that assists ambulance dispatchers in their day-to-day work. The research work summarized in this book represents a first step towards the development of such a tool.

Ambulance routing problems belong to the large class of vehicle routing problems involving pickups as well as deliveries. In the literature, problems involving passenger or patient transportation are usually referred to as dial-a-ride problems. A literature survey in two parts (Parragh et al., 2008a,b) resulted from the research work dedicated to the definition of a classification scheme for vehicle routing problems involving pickups as well as deliveries. In this book only a short summary of this work will be presented. The focus will lie on the review of research work belonging to the dial-a-ride problem class; since the formulation of different versions of this problem and the development of according solution methods form the main content of this thesis.

While in standard pickup and delivery problems goods are transported, in ambulance routing or dial-a-ride problem situations people are subject to transportation. Therefore, it is necessary to make sure that a certain quality of service is provided. This raises the question of what is perceived as quality of service by the persons transported. Usually, the term "user inconvenience" is used in this context. Low user inconvenience is linked to high quality of service, while high user inconvenience entails low quality of service. Low user inconvenience relates to punctual service and short individual ride times. A certain trade-off between user inconvenience and total operating costs can be observed. Lower user inconvenience usually entails higher operating costs and vice versa. User inconvenience can

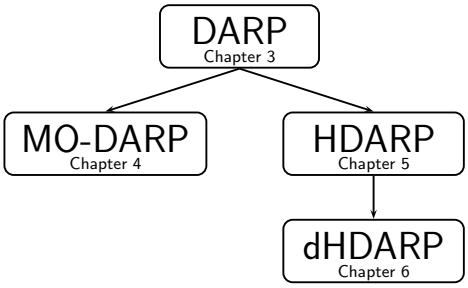


Figure 1.1: Outline

either be considered in terms of additional constraints or in terms of additional objectives. Both approaches are subject to investigation in this thesis.

In a first step, a solution method based on Variable Neighborhood Search (VNS) (Mladenovic and Hansen, 1997) for a rather standard Dial-A-Ride Problem (DARP) version will be developed in Chapter 3. A vehicle fleet of fixed size, time windows, maximum user ride times, and a route duration limit are among the constraints considered. This "standard" DARP will be extended in two ways, as shown in Figure 1.1.

In Chapter 4, besides routing costs, a user-oriented objective, minimizing user inconvenience, in terms of mean user ride time, will be introduced. This results in a problem version we will denote as Multi-Objective Dial-A-Ride-Problem (MO-DARP). An exact and a heuristic solution method will be devised. The heuristic solution method integrates VNS and Path Relinking (PR) (Glover and Laguna, 1997) in a two-phase scheme. The exact method iteratively solves single objective problems to optimality within a so-called ϵ-constraint framework (Laumanns et al., 2006). The developed procedures will provide the ambulance dispatcher with a number of transportation plans, which are incomparable across each other: neither will be better than any other transportation plan in both objectives. Information regarding their respective quality of service level as well as cost will be provided. In this case, the task of choosing a transportation plan out of the set of these efficient transportation plans is left with the person in charge.

In Chapter 5, in order to decrease the gap between theory and practice, heterogeneous patient types and a heterogeneous vehicle fleet, as employed by the ARC, will be added to the standard dial-a-ride model, resulting in a Heterogeneous Dial-A-Ride Problem (HDARP). Two mathematical problem formulations will be presented. Each of these will serve as the basis for a branch and cut algorithm. The previously developed VNS will also be adapted to HDARP.

In Chapter 6, the HDARP will be extended to the dHDARP, i.e. the heterogeneous dial-a-ride problem with driver related constraints, which corresponds to the real world ambulance routing problem faced by the ARC. Based on available information, staff related conditions will be introduced into the problem. These refer to the assignment of drivers and other

staff members to vehicles, and to the scheduling of lunch breaks and additional stops at the depot. An exact column generation procedure will be devised to compute lower bounds. These bounds will serve to assess the solution quality of the proposed VNS. In this case, further adaptations will be necessary in order to accommodate all real-world characteristics.

This book is organized as follows. In Chapter 2, first, a classification scheme for vehicle routing problems with pickups and deliveries is proposed. Then, a short introduction to the different solution paradigms applied in this field is given. This is followed by a brief review of the different solution methods proposed for each problem class. Each of the four subsequent chapters is dedicated to one of the above described ambulance routing problems. Each chapter contains a formal definition of the problem, the description of the developed solution procedure(s), and computational results for up to three (adapted) data sets from the literature. In Chapter 6 also real world instances are considered. At the end of each chapter, a short summary of the respective findings is given. The book closes with a conclusion, summarizing the obtained results and indicating future research directions.

2 Summarizing the state-of-the-art

2.1 Introduction

Over the past decades extensive research has been dedicated to modeling aspects as well as optimization methods in the field of vehicle routing. Especially transportation problems, involving both, pickups and deliveries, have received considerable attention. This is mainly due to the need for improved efficiency, as the traffic volume increases much faster than the street network grows (cf. Eurostat, 2004, 2006, for data on the European situation). Along with the increasing use of geographical information systems, companies seek to improve their transportation networks in order to tap the full potential of possible cost reduction. Ambulance routing problems, considering the transportation of people between pre-specified pickup and drop off locations, belong to the above mentioned problem class.

The rapidly growing body of research in the field of vehicle routing involving pickups as well as deliveries has led to a somewhat confusing terminology. Indeed, the same problem types are denoted by various names and different problem classes are referred to by the same denotations. The problem we will denote traveling salesman problem with mixed backhauls, e.g., has been denoted as Traveling Salesman Problem (TSP) with pickup and delivery (Mosheiov, 1994), TSP with delivery and backhauls (Anily and Mosheiov, 1994), and as TSP with deliveries and collections (Baldacci et al., 2003). Its multi vehicle version has been referred to as mixed Vehicle Routing Problem (VRP) with backhauls (Ropke and Pisinger, 2006b; Salhi and Nagy, 1999), as VRP with backhauls with mixed load (e.g. Dethloff, 2002), and as pickup and delivery problem (Mosheiov, 1998). Despite the fact that the naming pickup and delivery problem has most often been used to refer to an entirely different problem class (see below). This situation calls for a clear classification scheme and naming.

In this chapter we will introduce such a classification scheme. Thereafter, a short introduction to solution concepts (exact methods, heuristics, and metaheuristics) applied in the vehicle routing field will be given. This is followed by a condensed overview of solution methods developed in the pickup and delivery problem domain, following the developed classification scheme, while focusing on those works related to the static ambulance routing field. For further details on the different vehicle routing problem classes involving pickups as well as deliveries we refer to the two articles (Parragh et al., 2008a,b). They form the basis of this chapter. A recent work by Berbeglia et al. (2007) provides a different classification

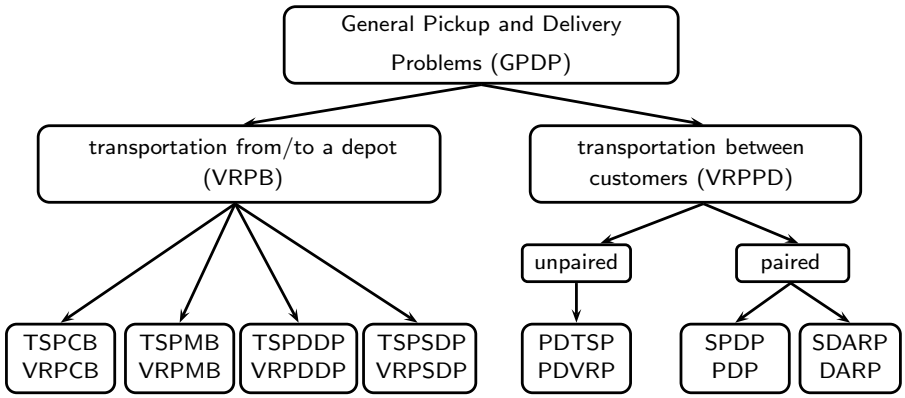


Figure 2.1: A classification scheme

scheme and survey for static vehicle routing problems involving pickups as well as deliveries. This work has been compiled in parallel to our two-part article.

2.2 Classification

We distinguish two problem classes. The first class refers to situations where all goods delivered have to be loaded at one or several depots and all goods picked up have to be transported to one or several depots. Problems of this class are usually referred to as VRP with Backhauls (VRPB), a term coined by Goetschalckx and Jacobs-Blecha (1989). The second class comprises all those problems where goods (passengers) are transported between pickup and delivery customers (points) and will be referred to as VRP with Pickups and Deliveries (VRPPD). The two pickup and delivery problem classes are depicted in Figure 2.1. Their subclasses are described in the following.

2.2.1 VRPB subclass definitions

The VRPB can be further divided into four subclasses. In the first two subclasses, customers are either delivery or pickup customers but cannot be both. In the last two subclasses, each customer requires a delivery and a pickup. The first subclass is characterized by the requirement that the group or cluster of delivery customers has to be served before the first pickup customer can be visited. Delivery customers are also denoted as linehaul customers, pickup customers as backhaul customers. We will refer to this problem class as VRP with Clustered Backhauls (VRPCB). Its single vehicle case will be denoted as TSP with Clustered Backhauls (TSPCB).

The second VRPB subclass does not consider a clustering restriction. Mixed visiting sequences are explicitly allowed. We will denote this problem class as VRP with Mixed

linehauls and Backhauls (VRPMB) in the multi vehicle case, and TSP with Mixed linehauls and Backhauls (TSPMB) in the single vehicle case.

The third VRPB subclass describes situations where customers can be associated with both a linehaul and a backhaul quantity but, in contrast to subclass four, it is not required that every customer is only visited once. Rather, two visits, one for delivery and one for pickup are possible. In this case, so called lasso solutions can occur, in which first a few customers are visited for delivery service only, in order to empty the vehicle partially. Then, in the "loop of the lasso", customers are visited for both pickup and delivery service. In the end, the pickups are performed for the customers initially visited for delivery. We will refer to the single vehicle case as TSP with Divisible Delivery and Pickup (TSPDDP) and to the multi vehicle case as VRP with Divisible Delivery and Pickup (VRPDDP) in order to emphasize that a customer can either be visited once for both pickup and delivery or twice, first for delivery and then for pickup.

The fourth VRPB subclass covers situations where every customer may be associated with a linehaul as well as a backhaul quantity. It is imposed that every customer can only be visited exactly once. We will denote this problem class as VRP with Simultaneous Delivery and Pickup (VRPSDP), its single vehicle version as TSP with Simultaneous Delivery and Pickup (TSPSDP).

2.2.2 VRPPD subclass definitions

The class we denote VRPPD refers to problems where goods are transported from pickup to delivery points. It can be further divided into two subclasses (see Figure 2.1). The first subclass refers to situations where pickup and delivery locations are unpaired. A homogeneous good is considered. Each unit picked up can be used to fulfill the demand of any delivery customer. In the literature mostly the single vehicle case is tackled. Since also a multi vehicle application has been reported (see Dror et al., 1998) we will denote this problem class as Pickup and Delivery VRP (PDVRP) and Pickup and Delivery TSP (PDTSP), in the multi and in the single vehicle case, respectively.

The second VRPPD subclass comprises the classical Pickup and Delivery Problem (PDP) and the Dial-A-Ride Problem (DARP). Both consider transportation requests, each associated with an origin and a destination, resulting in paired pickup and delivery points. The PDP deals with the transportation of goods while the DARP deals with passenger transportation. This difference is usually expressed in terms of additional constraints or objectives that take user (in)convenience into account. A majority of the work published denotes this problem class as Pickup and Delivery Problem (PDP) (see e.g. Dumas et al., 1991; van der Bruggen et al., 1993). We will follow this naming. Dial-a-ride problems are also mostly referred to as such. We denote the single vehicle case of the PDP as SPDP, the single vehicle case of the DARP as SDARP. All problems dealt with in the subsequent

chapters of this book belong to the DARP class.

2.3 Solution methods

In general, two different types of solution methods can be distinguished. These are exact algorithms yielding an optimal solution to the problem handled and heuristic algorithms, computing (hopefully) near optimal solutions within short or at least acceptable computation times.

2.3.1 Exact methods

Exact solution methods applied to vehicle routing problems considering both pickups as well as deliveries involve well known solution paradigms in combinatorial optimization; such as, among others, *branch and bound*, *branch and cut*, and *branch and price* algorithms.

In *branch and bound* algorithms, first the according Linear Programming (LP) relaxation to the respective problem formulated as a (mixed) Integer Program (IP) is solved. The solution of the LP relaxation will provide a *lower bound* for the solution of the original IP (in the context of minimization). In case the obtained solution is integer, the optimal solution to the original IP has been found. Otherwise, a branch and bound tree is built. From the root node two child nodes are generated by *branching*. At each child node, a new LP is solved with an additional constraint; either an upper or a lower bound on one of the variables which are supposed to be integer but are associated with a fractional value in the current solution is set. In the subsequent iterations each child node serves as the parent node for two new child nodes in the tree. The tree is explored in a branch and bound fashion; bounds are obtained by the optimal solution values to the LP at the nodes of the search tree. If the lower bound at some node of the search tree is greater than the upper bound obtained at another node, the former node can be excluded from the search.

An alternative method to the branch and bound method is the so-called *cutting plane* algorithm. First, as in branch and bound, the LP relaxation of the original IP is solved. In case the obtained solution is not integer, a *cut* is generated that *separates* the optimal solution from the true feasible set. A valid cut has two properties. First, any feasible point of the IP satisfies the cut; and second, the current optimal solution to the LP will violate the cut (Winston, 1994). This is illustrated in Figure 2.2. The gray space corresponds to the feasible region of the LP relaxation; the black dots to the set of feasible solutions of the original IP. The dashed line represents the objective function. The optimal solution to the LP relaxation corresponds to the lower left corner of the feasible region and yields a lower bound. A cutting plane, as given by the dotted line, separates this solution but does not cut off any of the IP feasible points. Also the optimal solution of the original IP is shown ($x_1 = 1, x_2 = 2$). Cuts are thus iteratively added to the LP relaxation until an integer

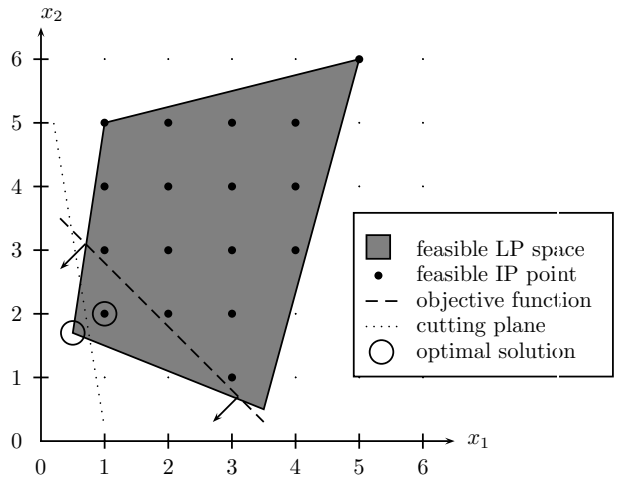


Figure 2.2: A lower bound and a cutting plane

solution is obtained.

Branch and cut methods combine the branch and bound and the cutting plane idea. In difference to branch and bound only a subset of the original constraints are considered in the LP relaxation. Typically, all constraint families of exponential size are not included. So-called *separation algorithms* will check the solution of the LP relaxation at the current node in the search tree for violations of the omitted constraints. In case none of the omitted constraints are violated, an optimal solution to the LP has been found. Otherwise, if at least one violated constraint has been detected by a separation procedure the violated constraint(s) is (are) added to the LP, and the updated LP is solved again. This is repeated until the separation procedures fail to detect additional violated constraints. A branch and cut algorithm will be employed to solve the heterogeneous dial-a-ride problem in Chapter 5. We refer to this chapter for further details.

The *column generation* method has been introduced by Gilmore and Gomory (1961) to solve large scale LP. Its basic principle is that it is not necessary to consider all columns at once. The idea is to search in an efficient way for columns that will price out favorably by taking dual information into account. The Dantzig-Wolfe decomposition method uses this idea. It has been first introduced by Dantzig and Wolfe (1960) for LP. Many problems can be formulated in two ways: in terms of a so-called *compact formulation* (such as the 2-index and 3-index formulations of the DARP presented in the following chapters) and in terms of a so-called *extensive formulation* (such as, e.g, set partitioning type formulations). Compact formulations can be transformed into extensive formulations by means of *decomposition*. The extensive formulation usually contains less rows (constraints) but a lot more columns (variables) than the compact formulation. To solve the LP relaxation of the extensive formulation, column generation is used. In vehicle routing these columns usually refer to

routes. Thus, the optimal combination out of the set of all feasible routes is searched. This set is usually too large to be considered at once. Therefore, so-called pricing procedures are used to identify favorable columns (*pricing subproblem*), by using dual information from the solution of the *restricted master* (i.e. the extensive formulation) considering the set of columns generated so far. The optimal solution to the original LP has been found if no additional favorable columns exist. For further information we refer to Desaulniers et al. (2005). This solution method is applied to the dial-a-ride problem variant considered in Chapter 6 of this book. Integrated into a branch and bound scheme, where each linear relaxation is solved by means of column generation, results in a so-called *branch and price* algorithm. In case also cuts are added (as in branch and cut) the resulting solution method is referred to as *branch and cut and price*.

In case only a subset of the tree is considered or, e.g., in column generation only heuristic pricing procedures are applied, these exact methods are turned into heuristic procedures. Further information on exact solution algorithms can be found, e.g., in Barnhart et al. (1998); Fischetti and Toth (1989); Padberg and Rinaldi (1991).

2.3.2 Heuristics

Following Semet and Laporte (2002), heuristic methods in the field of vehicle routing can be further divided into classical heuristics and metaheuristics. Classical heuristic methods comprise construction heuristics, two-phase heuristics, and improvement methods.

Construction heuristics build a feasible solution, trying to keep the objective function value as low as possible (in the context of minimization). Once a (customer) location has been inserted, it is usually not removed again. Popular concepts belonging to this class are sequential and parallel insertion heuristics. Sequential heuristics only consider one route at a time, while parallel heuristics consider multiple routes at once. A popular insertion criterion is cheapest insertion; the (customer) location resulting in the least cost increase regarding the so far constructed partial solution is inserted next.

Two-phase heuristics describe cluster-first-route-second and route-first-cluster-second implementations. In case of *cluster-first-route-second*, the (customer) locations are first assigned to routes, following some criterion and only then for each cluster (route) the order in which the (customer) locations are visited is determined. The reverse is done in case of *route-first-cluster-second* heuristics. Here, first a giant tour is constructed and only then this tour is segmented into as many clusters (routes) as necessary.

Improvement heuristics either consider each vehicle route at a time (intra-tour) or several routes (inter-tour). Popular intra-tour improvement heuristics use the λ-opt mechanism of Lin (1965) (especially 2-opt and 3-opt). λ edges are removed from a route and the remaining route segments are reconnected in all possible ways. Every time an improved solution is encountered the according reconnection is kept. This is repeated until no further improve-

ments are possible. The Or-opt (Or, 1976) method, e.g., relocates sequences consisting of 3, 2, and 1 vertices, resulting in a restricted form of 3-opt.

2.3.3 Metaheuristics

The term metaheuristic has been used for the first time by Glover (1986) to refer to a tabu search heuristic. A metaheuristic distinguishes itself from a classical heuristic by some "meta" structure that allows the search to escape from local optima. Thus, intermediate infeasible or deteriorating solutions are allowed during the search. Metaheuristics in the field of vehicle routing often apply concepts of classical construction and improvement heuristics. Usually they find better local optima than classical heuristic methods but they also tend to need longer computation times (Gendreau et al., 2002). On the one hand, there are metaheuristic approaches that are population based or related to population based methods, such as genetic algorithms or ant colony optimization, and, on the other hand, there are methods that are based on different local search neighborhoods, such as tabu search, variable neighborhood search, or simulated annealing. The term *neighborhood* refers to all solutions that can be constructed by applying a certain local search operator (e.g. a simple vertex move) to a given solution.

In *genetic algorithms*, in each iteration a population of solutions is considered. Every new population of solutions is obtained from the previous one by operators such as recombination and mutation, keeping the best (new) solutions and eliminating the worst. In *ant colony optimization*, in each iteration a construction heuristic is used to generate a number of new solutions based on information from previous iterations.

One of the most popular metaheuristics is *tabu search* (Glover and Laguna, 1997). It works as follows. A simple neighborhood operator such as, e.g. an inter-tour vertex move is used to transform the current solution into a new solution. All possible moves are considered and the best non-tabu one is used to constitute a new solution. The reverse move of the one that was used to build the new solution is then set "tabu" for a certain number of iterations. The new solution may be worse than the previous solution. This concept allows the search to escape from local optima. In *variable neighborhood search* (Mladenovic and Hansen, 1997), several neighborhoods of different size are considered. Ideally, the smallest should be contained in the next larger one and so on. If the smallest neighborhood consists of all possible inter-tour moves of one vertex, the next larger neighborhood could consist of all possible moves of one and two vertices, and so on. In every iteration a new solution is constructed at random in the current neighborhood. Then some improvement heuristic is used to optimize this solution locally. In case it improves the current incumbent solution, it becomes the new incumbent and the search continues with the smallest neighborhood. Otherwise, the next larger neighborhood is considered and another random solution is constructed. Also in *simulated annealing* (Kirkpatrick et al., 1983; Černy, 1985) in every iteration a random

solution is constructed in a given neighborhood. In case this solution is better than the current incumbent, it becomes the new incumbent solution. Otherwise, it is accepted with a certain probability. This probability depends on the solution value of the new solution and a "temperature". The "temperature" is decreased during the search according to a certain scheme. The "cooler" it gets the lower the probability that a deteriorating new solution is accepted as the new incumbent solution. The variable neighborhood search concept, integrating ideas from simulated annealing, will be used to solve the different ambulance routing problems considered in this book. For additional information we thus refer to the following chapters. Further information on local search methods can be found, e.g., in Aarts and Lenstra (1997), neighborhood based methods are discussed, e.g., in Bräysy and Gendreau (2005); Funke et al. (2005) and metaheuristics in Hoos and Stützle (2005).

2.4 Solution methods for VRPB

The development of solution methods for VRPB is most probably motivated by the fact that combined linehaul and backhaul tours may result in less empty hauls and thus in possible cost reductions. Depending on the type of good transported, company policies, or customer preferences, backhauls may have to be done last, customers have to be visited only once, etc. In the following, first, research work dealing with the clustered version (all linehauls before backhauls) will be reviewed, this is followed, by the mixed case, the combined and divisible case (all customers may demand both services and can be visited twice), and the combined case with simultaneous service (all customers may demand both services but can only be visited once).

2.4.1 All linehauls before backhauls (TSPCB, VRPCB)

The TSPCB can be viewed as a special case of the Clustered Traveling Salesman Problem (CTSP), where only two clusters are considered. The CTSP was first formulated in Chisman (1975). Already in Lokin (1978) an application of the CTSP to a backhaul problem is suggested. Thus, all solution algorithms for CTSP, e.g., those presented in Gendreau et al. (1996b); Jongens and Volgenant (1985); Laporte et al. (1996); Potvin and Guertin (1996), are also valid solution techniques for the TSPCB with the additional constraint that the set of linehaul customers is visited first. A survey covering the different VRPCB solution methods can be found in Toth and Vigo (2002).

The first exact approach for the VRPCB is due to Yano et al. (1987). The proposed branch and bound method generates optimal routing plans with up to 4 linehaul and 4 backhaul customers per route. Further, more sophisticated branch and bound derived algorithms followed (Gélinas et al., 1995; Mingozzi et al., 1999; Toth and Vigo, 1997a). The latest are able to solve benchmark instances (Goetschalckx and Jacobs-Blecha, 1989; Toth and Vigo,

1996b, 1999) with up to 100 customers to optimality .

Quite a lot of research has been conducted in the field of heuristic methods for VRPCB. The solution paradigms applied range from simple as well as more sophisticated construction heuristics (Deif and Bodin, 1984; Derigs and Metz, 1992), to construction-improvement type implementations (Gendreau et al., 1996a; Goetschalckx and Jacobs-Blecha, 1989; Thangiah et al., 1996), and cluster-first-route-second methods (Anily, 1996; Min et al., 1992; Toth and Vigo, 1996b, 1999) applying well-known operators such as 2-exchange, 3-exchange (Lin, 1965), λ-interchange (Osman, 1993), 2-opt*-exchange (Potvin and Rousseau, 1995), or the GENI idea (Gendreau et al., 1992).

Those research works that are among the most recent in the field of VRPCB, belong to the metaheuristic domain. The tabu search concept has been employed by Brandão (2006); Crispim and Brandão (2001); Duhamel et al. (1997); Osman and Wassan (2002). However, also neural networks (Ghaziri and Osman, 2003, 2006), genetic algorithms (Ganesh and Narendran, 2007; Potvin et al., 1996), an insertion based ant system (Reimann et al., 2002), guided local search (Zhong and Cole, 2005), simulated annealing (Hasama et al., 1998), variable neighborhood search (Crispim and Brandão, 2001; Mladenovic and Hansen, 1997), and adaptive large neighborhood search (Ropke and Pisinger, 2006b) have been used to solve different versions of VRPCB.

Over the years with increasing computational power a shift from simple heuristic methods towards more sophisticated metaheuristic solution procedures can be observed. Thus, recent state-of-the-art methods in the field of VRPCB predominantly belong to the metaheuristic domain. Comparison can be done by looking at the different results achieved for the same set of benchmark instances. In case of the VRPCB without time windows, the benchmark instances most often used are the ones of Goetschalckx and Jacobs-Blecha (1989) (GJ89) and Toth and Vigo (1996b) (TV96). The largest instance solved to optimality of the GJ89 data set comprises 90 customers and in case of the TV96 data set 100 customers (see Mingozzi et al., 1999). The latest new best results for these data sets are reported in Ropke and Pisinger (2006b) and Brandão (2006). In case of the VRPCB with time windows the data set proposed in Gélinas et al. (1995) (GDDS95) is the prevalent one. The largest instance solved to optimality within this data set consists of 100 customers (cf. Gélinas et al., 1995). Most recent new best results have also been reported by Ropke and Pisinger (2006b).

2.4.2 Mixed linehauls and backhauls (TSPMB, VRPMB)

We now turn to the VRPMB where linehaul and backhaul customers can occur in any order along the route. Exact solution methods for the VRPMB have only been developed for the single vehicle case. Tzoreff et al. (2002) present a linear time algorithm for tree graphs, and polynomial time algorithms for a cycle and a warehouse graph. Süral and Bookbinder (2003) work on bounds of the linear relaxation and Baldacci et al. (2003) discuss valid inequalities

and embed them into a branch and cut algorithm.

The first solution methods proposed for the VRPMB belong to the field of heuristics. These are two heuristics using the Clarke-Wright algorithm (Clarke and Wright, 1964) to schedule the linehauls and different insertion procedures to insert the backhauls (Casco et al., 1988; Golden et al., 1985). A similar idea is also used by Wade and Salhi (2002). Salhi and Nagy (1999) extend the idea of Golden et al. (1985), introducing the notion of clusters and borderline customers in a multi-depot setting. Giant tour partitioning heuristics have been developed by Mosheiov (1998). Further solution methods involve a minimum spanning tree based procedure (Anily and Mosheiov, 1994), a TSP tour based algorithm, a cheapest feasible insertion algorithm (Mosheiov, 1994) for the single vehicle case, and an extension of the cheapest insertion heuristic (Dethloff, 2002) for the multi vehicle case. The construction-improvement principle is applied by Nagy and Salhi (2005).

Metaheuristics have not been applied as extensively to the VRPMB as to other problem types. The solution paradigms applied range from greedy randomized adaptive search (Kontoravdis and Bard, 1995), simulated annealing (Hasama et al., 1998), and ant systems (Reimann and Ulrich, 2006; Wade and Salhi, 2004) to a combination of tabu search and variable neighborhood descent (Crispim and Brandão, 2005) and adaptive large neighborhood search (Ropke and Pisinger, 2006b).

The largest TSPMB instance solved to optimality is reported in Baldacci et al. (2003). It is a single vehicle instance of the data set provided by Gendreau et al. (1999), containing 200 customers. The VRPMB instances most widely used are those proposed by Salhi and Nagy (1999). Most recent improved results for these instances are given by Ropke and Pisinger (2006b), outperforming earlier results by Nagy and Salhi (2005).

2.4.3 Divisible delivery and pickup (TSPDDP, VRPDDP)

This problem class is a mixture of the previously described VRPMB and the VRPSDP, subject to review in the next section. In contrast to the VRPMB, every customer can be associated with a pickup and a delivery quantity. However, these customers do not have to be visited exactly once. They can be visited twice, once for pickup and once for delivery service. Only little research has been explicitly dedicated to this problem class. However, all the solution methods designed for the VRPMB can be applied to VRPDDP instances if every customer demanding pickup and delivery service is modeled as two separate customers. Salhi and Nagy (1999) apply their cluster insertion algorithm to VRPSDP instances, resulting in solutions that may violate the one-single-visit-per-customer restriction. Thus, they actually solve a VRPDDP. Halskau et al. (2001), on the other hand, explicitly relax the VRPSDP to the VRPDDP. The aim is to create so-called lasso solutions, i.e. customers along the spoke are visited twice (first for delivery and second for pickup service). Customers along the loop are only visited once. Hoff and Løkketangen (2006) also study lasso solutions but restricted

to the single vehicle case. An in-depth study of different solution shapes for TSPDDP is conducted by Gribkovskaia et al. (2007); they consider lasso, Hamiltonian, and double-path solutions. The concept of "general solutions" is introduced. Their work is motivated by the fact that additional cost reductions can be realized when relaxing the VRPSDP to the VRPDDP. The proposed methods are classical construction and improvement heuristics and a tabu search algorithm. They are tested on instances containing up to 100 customers. The results show that the best solutions obtained are often non-Hamiltonian and may contain up to two customers that are visited twice.

2.4.4 Simultaneous delivery and pickup (TSPSDP, VRPSDP)

The difference between VRPDDP and VRPSDP refers to customers demanding pickup and delivery service. In case of the VRPSDP these customers have to be visited exactly once for both services. The VRPMB is a special case of the VRPSDP where every customer only demands a pickup or a delivery but not both. This problem class was first defined by Min (1989).

The only exact algorithm for the VRPSDP with time windows is presented in Angelelli and Mansini (2002). Based on a set covering formulation of the master problem a branch and price approach is designed. Dell'Amico et al. (2006) also propose a branch and price algorithm to solve the VRPSDP but without time windows. They use a hierarchy based on five pricing procedures: four heuristics and one exact method.

Several heuristic methods for different versions of the VRPSDP can be found in the literature. Construction-improvement algorithms for the single vehicle case have been developed by Gendreau et al. (1999) and Alshamrani et al. (2007). Both are based on TSP cycles, improved using arc-exchanges, or the Or-opt operator. The latter considers a periodic version and stochastic demand figures. Gendreau et al. (1999) compare the construction algorithms of Mosheiov (1994) and Anily and Mosheiov (1994) to a cheapest feasible insertion heuristic. A cheapest insertion based algorithm for the multi vehicle case is also used by Dethloff (2001). The cluster-first-route-second idea is applied by Halse (1992) and Min (1989).

Also metaheuristic solution methods have been applied to the VRPSDP. The first metaheuristic for the TSPSDP is a tabu search algorithm using a 2-exchange neighborhood (Gendreau et al., 1999). Tabu search has also been applied to the multi vehicle version (see Bianchessi and Righini, 2007; Tang Montané and Galvão, 2006). Chen and Wu (2006) integrate a tabu list into a record to record algorithm, a hybrid between a tabu search and a variable neighborhood descent algorithm is described in Crispim and Brandão (2005)

Again the same trend as for VRPMB can be observed. Early research favored simple heuristic algorithms whereas recent algorithms mostly belong to the field of metaheuristic solution procedures. The largest VRPSDP instance solved to optimality comprises 40 requests (Dell'Amico et al., 2006); however, no standard benchmark instance is considered.

Two data sets have been most often referred to. These are those of Salhi and Nagy (1999) (SN99b) and Dethloff (2001) (Det01). The best pooled results for the SN99b instances hold Ropke and Pisinger (2006b) and Nagy and Salhi (2005). Tang Montané and Galvão (2006) also report improved solutions, however, not the whole set is considered. Consequently, a direct comparison to the other two is impossible. For the Det01 data set Ropke and Pisinger (2006b), Tang Montané and Galvão (2006), and Bianchessi and Righini (2007) obtain new best results of similar quality, but present them in different pooled form, only comparing themselves to the results of Dethloff (2001). Whatever method produces the best results, all of them are metaheuristics, clearly indicating that these more sophisticated methods outperform straightforward heuristic procedures.

2.5 Solution methods for VRPPD

In the following section an overview of the different solution methods for the PDVRP, the PDP, and the DARP, forming the second class of vehicle routing problems involving pickups and deliveries, are presented. Solution methods again classify into exact, heuristic, and metaheuristic approaches. In all previously described problem classes, goods were loaded at the depot and delivered to different locations, and picked up at different locations and returned to the depot. All problem classes described in the following deal with situations where goods are transported between pickup and delivery customers. In the first problem class, an identical good is considered. Thus, every customer's demand may be fulfilled by every other customer's supply. In the last two problem classes pickup and delivery points are paired and the goods transported may or may not be identical. In the last class instead of goods, people are subject to transportation. The focus will lie on the last class (DARP). All ambulance routing problems discusses in the following chapters belong to this problem class.

2.5.1 Unpaired pickups and deliveries (PDVRP, PDTSP)

The PDVRP, i.e. the problem class where every good can be picked up and transported anywhere, did not receive as much attention in the literature as the other problem classes. Moreover, most of the literature is restricted to the PDTSP. Therefore, with the exception of Dror et al. (1998), all solution methods presented are only applicable to the one vehicle case. To the authors' knowledge no metaheuristic approach for the PDTSP has been proposed until today.

The only exact method proposed for the problem at hand has been introduced by Hernández-Pérez and Salazar-González (2003, 2004a), using a branch and cut framework. The test instances solved are adaptations of the ones used in Mosheiov (1994) and Gendreau et al. (1999), containing up to 75 customers. This branch and cut algorithm is also used as a

heuristic in an incomplete optimization scheme (Hernández-Pérez and Salazar-González, 2004b). A construction-improvement type algorithm, applying a greedy construction procedure, improved by 2-opt and 3-opt exchanges is proposed in the same paper.

A special case of the PDTSP is considered in Chalasani and Motwani (1999); the number of goods to be picked up is equal to the number of goods to be delivered; the demand (supply) at every delivery (pickup) location is equal to one. This problem is an extension of the swapping problem where the vehicle's capacity is also set to one. Chalasani and Motwani propose an approximation algorithm with a worst case bound of 9.5. Anily and Bramel (1999) devise a polynomial time iterated tour matching algorithm for the same problem. An approximation algorithm on a tree graph with a worst case bound of 2 is developed in Lim et al. (2005). The PDTSP on a tree and on a line is also subject to investigation in Wang et al. (2006). They propose an $O(|V|^2/\min\{C,|V|\})$ algorithm for the line case. The unit capacity as well as the uncapacitated version can be solved in linear time. On a tree an $O(|V|)$ algorithm is devised for the case of unit capacity and an $O(|V|^2)$ algorithm for the uncapacitated case. $|V|$ gives the number of vertices and C the vehicle capacity.

Finally, Dror et al. (1998) propose a heuristic algorithm for the application of the PDVRP to the redistribution of self-service cars. It is related to Dijkstra's algorithm (Dijkstra, 1959). Also other solution approaches are briefly discussed.

2.5.2 Pickup and delivery problems (SPDP, PDP)

Solution methods for the classical PDP, where every transportation request is associated with a pickup and a delivery point, are presented in this section. Lokin (1978) was the first to discuss the incorporation of precedence constraints into the traditional TSP, needed to formulate the PDP. The first attempt to generalize the PDP in unified notation was proposed by Savelsbergh and Sol (1995), covering all possible versions of the PDP, including the DARP. They also provide a brief overview of existing solution methods until 1995. Mitrović-Minić (1998) present a survey on the PDP with Time Windows (PDPTW). An early survey on vehicle routing problems, already including the PDP is given in Desrochers et al. (1988). Cordeau et al. (2004) review demand responsive transport, covering PDP and DARP. Further surveys on solution methods can be found in Assad (1988); Desaulniers et al. (2002); Desrochers et al. (1988). In contrast to all problem classes discussed so far, in case of PDP and also DARP, in addition to the various static problem versions, also dynamic variants exist. The term "dynamic" refers to situations were only some requests are known ahead of the planning period. A major part, however, comes in during the planning period and has to be scheduled "on-line". In the following paragraphs the various solution techniques for the static PDP are summarized according to exact, heuristic, and metaheuristic approaches. Thereafter, the different dynamic approaches are briefly discussed.

2.5.2.1 Static variants

A number of exact solution procedures have been presented for the static PDP. Ruland and Rodin (1997) present a branch and cut algorithm to solve the SPDP. A branch and bound algorithm and various valid inequalities for the single vehicle PDPTW with Last-In-First-Out (LIFO) loading, i.e. goods have to be delivered in the reverse order they were picked up, is reported in Cordeau et al. (2006). The first exact solution method applicable to both the single as well as the multi vehicle case dates back to Kalantari et al. (1985). The branch and bound algorithm described is an extension of the one developed by Little et al. (1963). Another branch and bound algorithm and valid inequalities are discussed in Lu and Dessouky (2004). Column generation approaches have also been applied to the PDPTW (Desrosiers and Dumas, 1988; Dumas et al., 1991; Sigurd et al., 2004). A branch and cut algorithm departing from two different 2-index PDPTW formulations is studied in Ropke et al. (2007). A branch and cut and price approach, i.e. valid inequalities are added in a branch and cut fashion, and their impact on the structure of the pricing problem is discussed in Ropke and Cordeau (2008). For further details we refer to a survey by Cordeau et al. (2007), covering recent work on PDP while focusing on exact approaches.

Heuristics for the static PDP have first been proposed in the 1980s. Sexton and Choi (1986) use Bender's decomposition procedure to solve the static SPDP approximately. Construction-improvement heuristics for the SPDP are discussed in Renaud et al. (2000); van der Bruggen et al. (1993), using exchange, deletion, and re-insertion operators (e.g. 4-opt* (Renaud et al., 1996)) in the improvement phase. Perturbation heuristics have been employed by Renaud et al. (2002); a sophisticated construction procedure for the multi vehicle case by Lu and Dessouky (2006). The idea of mini-clusters has also been used (Shang and Cuff, 1996; Thangiah and Awan, 2006). A so-called "squeaky wheel" iterative insertion procedure has been developed by Lim et al. (2002); Xu et al. (2003) propose a column generation based heuristic algorithm, considering several real world motivated constraints. A construction-improvement heuristic for the multi vehicle case with transshipment is presented in Mitrović-Minić and Laporte (2006).

In the metaheuristic domain mostly the PDP with time windows is considered. In the single vehicle case, a (probabilistic) tabu search (Landrieu et al., 2001) and a variable neighborhood search heuristic (Carrabs et al., 2007) have been used. The latter considers an additional constraint regarding rear loading (items can only be delivered in the reverse order they were picked up). A multi vehicle version including this constraint is solved by means of a greedy randomized adaptive search procedure (Ambrosini et al., 2004). Otherwise, mostly heuristics using the tabu search paradigm (Caricato et al., 2003; Lau and Liang, 2001, 2002; Nanry and Barnes, 2000) and evolutionary techniques, such as genetic algorithms or indirect search, (Creput et al., 2004; Derigs and Döhmer, 2008; Jung and Haghani, 2000; Pankratz, 2005; Schönberger et al., 2003) have been used to solve different versions of the PDP. Most

of them consider time windows. Also two hybrid algorithms have been reported in the literature, combining simulated annealing with tabu search (Li and Lim, 2001), and with large neighborhood search (Bent and van Hentenryck, 2006). Large neighborhood search has also been successfully applied in combination with an adaptive removal and insertion procedure selection scheme (Ropke and Pisinger, 2006a).

Summarizing, the largest static PDP problem instance solved to optimality with a state-of-the-art method comprises 205 requests (Sigurd et al., 2004). However, the size of the largest instance solved is not always a good indicator; tightly constrained problems are easier to solve than less tightly constrained ones. The benchmark data set most often used to assess the performance of (meta)heuristic methods for the static PDPTW is the one described in Li and Lim (2001) (LL01, LL01+). Recent new best results have been presented in Ropke and Pisinger (2006a) and Bent and van Hentenryck (2006), two metaheuristic solution procedures. The metaheuristic of Li and Lim (2001) still holds the best results for a part of the smaller instances. However, also exact methods advanced; the more tightly constrained part of the LL01 data set has recently been solved by a state-of-the-art branch and cut and price algorithm (Ropke and Cordeau, 2008).

2.5.2.2 Dynamic and stochastic variants

Although many real world PDP are inherently dynamic, the dynamic version of the PDP has not received as much attention as its static counterpart. The term *dynamic* usually indicates that the routing and scheduling of requests has to be done in real time; new requests come in dynamically during the planning horizon and have to be inserted into existing partial routes. In general, the same objectives as for the static PDP are applied, e.g. the minimization of total operating costs. Surveys on dynamic routing can be found in Ghiani et al. (2003); Psaraftis (1988). In *stochastic* variants of the PDP, information regarding a certain part of the data (e.g. vehicle travel times) are only available in terms of probability distributions. So far, exact procedures have not been used to solve dynamic or stochastic PDP.

Swihart and Papstavrou (1999) solve a stochastic SPDP. They test three routing policies, a sectoring, a nearest neighbor and a stacker crane policy. Lower bounds on the expected time a request remains in the system under light and heavy traffic conditions are computed.

Three *online* algorithms (REPLAN, IGNORE, and SMARTSTART) for a single server PDP are investigated in Ascheuer et al. (2000b). The objective considered is the completion time. In the literature the problem at hand is called online dial-a-ride problem. However, the denotation *dial-a-ride problem* is not used in the same way as defined in this book. The term *online* is used whenever no request is known in advance. By means of *competitive analysis*, i.e. the online (no knowledge about the future) algorithm is compared to its offline (complete knowledge about the future) counterpart (Jaillet and Stafford, 2001; Van Hentenryck and Bent, 2006), it can be shown that REPLAN and IGNORE are 5/2 competitive, while SMARTSTART has a competitive ratio of 2. Hauptmeier et al. (2000) discuss the

performance of REPLAN and IGNORE under reasonable load. Feuerstein and Stougie (2001) devise another 2-competitive algorithm. A probabilistic version (Coja-Oghlan et al., 2005) and the online problem minimizing maximum flow time (Krumke et al., 2005) are also investigated. Lipmann et al. (2004) study the influence of restricted information on the multi server online PDP, i.e. the destination of a request is only revealed after the object has been picked up. Competitive ratios of two deterministic strategies (REPLAN and SMARTCHOICE) for the time window case are computed in Yi and Tian (2005).

A heuristic using incomplete optimization for the dynamic multi vehicle PDP is proposed in Savelsbergh and Sol (1998). The solution methodology called DRIVE (Dynamic Routing of Independent VEhicles) incorporates a branch and price algorithm based on a set partitioning problem formulation. An insertion based heuristic using different types of order circuity control is discussed in Popken (2006). Fabri and Recht (2006) present an adaptation of a heuristic, initially designed for the dynamic DARP by Caramia et al. (2002).

Early research on dynamic multi vehicle PDP is conducted in Shen et al. (1995) and Potvin et al. (1995). Both articles focus on neural networks with learning capabilities to support vehicle dispatchers in real-time. A hybrid population based method, combining a genetic with a dynamic programming algorithm, for the dynamic SPDP with time windows is proposed in Jih and Hsu (1999). The first neighborhood based metaheuristic solution method for the dynamic multi vehicle PDPTW is presented in Malca and Semet (2004). A two-phase solution procedure is described in Mitrović-Minić and Laporte (2004). In the first phase an initial solution is constructed via cheapest insertion and improved by means of a tabu search algorithm. In the second phase different waiting strategies are used to schedule the requests. The tested waiting strategies (drive first, wait first, dynamic waiting, advanced dynamic waiting) differ regarding the vehicle's location when waiting occurs. Advanced dynamic waiting is also used by Mitrović-Minić et al. (2004) in a double horizon based heuristic. The routing part is again solved by means of a construction heuristic improved by tabu search. Another tabu search algorithm for the dynamic multi vehicle PDPTW has been proposed by Gendreau et al. (2006), using an ejection chain neighborhood (Glover, 1996). Gutenschwager et al. (2004) compare a steepest descent, a reactive tabu search, and a simulated annealing algorithm.

To summarize, over the past decades, in the field of dynamic PDP, a number of solution procedures have been developed. However, the proposed algorithms cannot be directly compared since, so far, no standardized simulation environment has been used by more than one group of authors. Benchmark instances are available; e.g. those used by Mitrović-Minić et al. (2004); Mitrović-Minić and Laporte (2004).

2.5.3 Dial-a-ride problems (SDARP, DARP)

The last problem class dealt with in this chapter is the dial-a-ride problem class. It has received considerable attention in the literature. The first publications in the field of passenger transportation date back to the late 1960s and early 1970s (cf. Rebibo, 1974; Wilson and Colvin, 1977; Wilson et al., 1971; Wilson and Weissberg, 1967). Additional information in the form of literature surveys on the different solution methods can be found in Cordeau and Laporte (2003a, 2007); Cordeau et al. (2004); Gendreau and Potvin (1998). As in the PDP class we distinguish between static and dynamic variants. The focus is put on the static part. All subsequent chapters deal with different problem variants belonging to the field of static DARP.

2.5.3.1 Static variants

As in the previous sections we distinguish between exact, heuristic, and metaheuristic approaches. Given the vast amount of research published in the field of DARP, an overview of the different heuristic and metaheuristic approaches is given in tabular form.

Exact methods Early research in the field of exact methods for the static SDARP predominantly belongs to the dynamic programming domain. An early exact dynamic programming algorithm for the SDARP is introduced in Psaraftis (1980). Service quality is taken care of by means of maximal position shift constraints compared to a first-come-first-serve visiting policy. A modified version of the above algorithm, considering time windows, is discussed in Psaraftis (1983b). Instead of backward recursion forward recursion is applied. A forward dynamic programming algorithm for SDARP is also discussed in Desrosiers et al. (1986). Possible states are reduced by eliminating those that are incompatible with respect to vehicle capacity, precedence, and time window restrictions. User inconvenience in terms of ride times is incorporated into time window construction, resulting in tight time windows at both, origin and destination of the transportation request.

Kikuchi (1984) develop a balanced linear programming transportation problem for the multi vehicle DARP, minimizing empty vehicle travel as well as idle times and thus fleet size. In a pre-processing step the service area is divided into zones, the time horizon into time periods. Every request is classified according to an origin and a destination zone as well as a departure and an arrival time period.

A branch and cut algorithm based on a 3-index formulation has been proposed by Cordeau (2006). New valid inequalities as well as previously developed ones for the PDP and the VRP are employed. The largest instance solved to optimality comprises 36 requests. Two branch and cut algorithms are described in Ropke et al. (2007). Instead of a 3-index formulation, two more efficient 2-index problem formulations and additional valid inequalities are employed. These two articles, presenting the most recent exact algorithms in the static DARP field,

form the basis of the branch and cut algorithm for the DARP with heterogeneous fleet and passengers discussed in Chapter 5 of this book. Further details regarding the different families of valid inequalities employed can be found in the same chapter.

Heuristics Over the past decades, a large number of heuristic algorithms have been proposed for the static DARP. Table 2.1 gives an overview of the various solution methods reported in chronological order, divided into single and multi vehicle approaches. For each entry the according literature reference, objective(s), and additional constraints considered are provided. Furthermore, either the benchmark instances used, or the size of the largest instance solved to optimality, in terms of number of requests, are given.

A heuristic routing and scheduling algorithm for the SDARP using Bender's decomposition is described in Sexton and Bodin (1985a,b). The scheduling problem can be solved optimally; the routing problem is solved with a heuristic algorithm. Other heuristic methods dealing with the single vehicle case involve a minimum spanning tree heuristic (Psaraftis, 1983a), adaptations of 2-opt and 3-opt heuristics (Psaraftis, 1983c), and a 2-opt improvement scheme switching between optimizing and sacrificing phases (Healy and Moll, 1995).

One of the first heuristic solution procedures for a static multi vehicle DARP is discussed in Cullen et al. (1981). They develop an interactive algorithm that follows the cluster-first-route-second approach. It is based on a set partitioning formulation solved by means of column generation. The location-allocation subproblem is only solved approximately. Other cluster-first-route-second approaches for different versions of the DARP are due to Bodin and Sexton (1986); Psaraftis (1986); Stein (1978a,b); Wolfler Calvo and Colorni (2007). A classical cluster-first-route-second algorithm has also been proposed by Borndörfer et al. (1997). Here, the clustering as well as the routing problem are modeled as set partitioning problems. The clustering problem can be solved optimally while the routing subproblems are solved approximately by a branch and bound algorithm (only a subset of all possible tours is used). Customer satisfaction is taken into account in terms of punctual service; customer ride times are implicitly considered by means of time windows.

An optimization based mini-clustering algorithm is presented in Ioachim et al. (1995). It uses column generation to obtain mini-clusters and an enhanced initialization procedure to decrease processing times. As in Desrosiers et al. (1988) also the case of multiple depots is considered. The idea of mini-clusters has also been employed by Desrosiers et al. (1988, 1991); Dumas et al. (1989).

Jaw et al. (1986) propose a sequential insertion procedure. First, customers are ordered by increasing earliest time for pickup. Then, they are inserted according to the cheapest feasible insertion criterion. The notion of active vehicle periods is used. Other construction heuristics have been proposed by Alfa (1986); Kikuchi and Rhee (1989); Roy et al. (1985a,b). A multi-objective approach is followed in Madsen et al. (1995). They discuss an insertion based algorithm called REBUS. The objectives considered are the total driving time, the

Table 2.1: Heuristics for the static DARP

Reference	Type	Obj.	Constr.	Algorithm	Bench./ Size
The single vehicle case					
Psaraftis (1983a)	-	min. RC	-	MST heur., local interchanges	up to 50 req.
Psaraftis (1983c)	-	min. RC	-	adapted 2-opt and 3-opt	up to 30 req.
Sexton and Bodin (1985a,b)	-	min. CI	DDT	routing and scheduling algorithm based on Bender's decomposition	up to 20 req.
Healy and Moll (1995)	-	min. RC	-	2-opt improvement, optimizing/sacrificing phases	up to 100 req.
The multi vehicle case					
Stein (1978a,b)	transfers	min. RC	TW	cluster first route second	-
Cullen et al. (1981)	-	min. RC	-	cluster first route second, column generation	up to 50 req.
Roy et al. (1985a,b)	HF	min. RC, min. CI	TW	parallel insertion	up to 578 req.
Bodin and Sexton (1986)	-	min. CI	DDT	cluster first route second	up to 85 req.
Jaw et al. (1986)	-	min. RC, min. CI	TW, RT	sequential feasible insertion algorithm	up to 2617 req.
Alfa (1986)	HF	min. RC	TW, RT	adapted heur. of (Jaw et al., 1986)	up to 49 req.
Psaraftis (1986)	-	min. RC, min. CI	TW, RT	comparison of Jaw's heur. and grouping-clustering- routing heur.	-
Desrosiers et al. (1988); Dumas et al. (1989)	MD	min. RC	TW	mini-clustering algorithm, column generation	up to 200 req.
Kikuchi and Rhee (1989)	-	max. NCS	TW	sequential insertion	up to 200 req.
Desrosiers et al. (1991)	HF	min. RC	TW, RD	improved mini- clustering algorithm of (Desrosiers et al., 1988)	up to 2411 req.
Potvin and Rousseau (1992)	-	min. RC	TW, RT	constraint directed search (beam search)	up to 90 req.
Ioachim et al. (1995)	HF, MD, S	min. NV, min. RC	TW	mini-clustering, column generation	up to 2545 req.
Madsen et al. (1995)	HF, S	min. RC, NV, TWT, DPS	TW, RD, RT	REBUS. insertion based algorithm	up to 300 req.
Toth and Vigo (1996a)	HF	min. RC	TW, RT	parallel insertion, improved by trip insertion, exchange, double insertion, moves	TV96a
Borndörfer et al. (1997)	HF, MD, S	min. RC	TW, RD, (RT)	cluster first route second, set partitioning, branch and bound	up to 1771 req.

continued on next page

Reference	Type	Obj.	Constr.	Algorithm	Bench./ Size
Diana and Dessouky (2004)	-	min. RC, min CI, idle times	TW, RT	parallel regret insertion heur.	up to 1000 req.
Xiang et al. (2006)	HF, S	min. RC	TW, RT, RD, BR	construction-improvement; clustering by TW, ideas of sweep heur.; local search improvement	up to 2000 req.
Wong and Bell (2006)	HF, S	min. RC, min. CI	TW, RT, RD	parallel insertion, improved by trip insertion	up to 150 req.
Wolfler Calvo and Colorni (2007)	-	max. NCS, max. SL	TW	cluster first route second, assignment heur., vertex reinsertions	up to 180 req.

Bench. = Benchmark, BR = BReak time between two trips, CI = Customer Inconvenience, Constr. = Constraints, DDT = Desired Delivery Time, DPS = Deviation from Promised Service, HF = Heterogeneous Fleet, heur. = heuristic(s), MD = Multi Depot, NCS = Number of Customers Served, NV = Number of Vehicles, Obj. = Objective(s), RC = Routing Cost, RD = Route Duration, req. = requests, TW = Time Windows, TWT = Total Waiting Time, RT = Ride Time, S = Service time, SL = Service Level; The respective benchmark instances are described in Section 2.6.

number of vehicles, the total waiting time, the deviation from promised service times as well as cost. The regret insertion based process is also subject to analysis in a study by Diana (2004); in order to determine why the performance of this heuristic is superior to that of other insertion rules.

The construction-improvement concept has also been used to solve the DARP (see Toth and Vigo, 1996a; Wong and Bell, 2006; Xiang et al., 2006). In the improvement phase operators such as trip insertion, exchange, and move are employed. Also beam search has been used to solve the multi-vehicle DARP (Potvin and Rousseau, 1992).

Metaheuristics A much smaller number of metaheuristic solution methods have been developed for the static DARP. Table 2.2 provides an overview of the different algorithms proposed. For each reference the same information as in the previous table is given.

Among the most popular metaheuristic concepts applied to the DARP are simulated annealing (Baugh et al., 1998; Colorni et al., 1996), tabu search (Aldaihani and Dessouky, 2003; Cordeau and Laporte, 2003a; Melachrinoudis et al., 2007), and genetic algorithms (Jørgensen et al., 2007; Rekiek et al., 2006; Uchimura et al., 1999). Toth and Vigo (1997b) describe a further local search based metaheuristic, a tabu thresholding algorithm, employing the neighborhoods defined in Toth and Vigo (1996a). In this book a variable neighborhood search algorithm will be developed and applied to different versions of the DARP. In Chapter 3 it is compared to the tabu search algorithm proposed by Cordeau and Laporte (2003b). They consider time windows at either origin or destination depending on the type of request (inbound or outbound). The neighborhood used is defined by moving one request to another route. The best possible move serves to generate a new incumbent solution. Reverse moves are declared tabu. However, an aspiration criterion is defined, such that tabu moves that provide a better solution, with respect to all other solutions already constructed

Table 2.2: Metaheuristics for the static DARP

Reference	Type	Obj.	Constr.	Algorithm	Bench./ Size
The multi vehicle case					
Colorni et al. (1996)	-	max. NCS, min CI	RD	simulated annealing	up to 100 req.
Toth and Vigo (1997b)	HF, MD	min. RC	TW, RT	parallel insertion algorithm, tabu thresholding	TV96a
Baugh et al. (1998)	-	min. NV, min. RC, min. CI	TW	simulated annealing	up to 300 req.
Uchimura et al. (1999)	-	min. RC	RT, RD	genetic algorithm	10 req.
Cordeau and Laporte (2003b)	S	min. RC	TW, RT, RD	tabu search	CL03, up to 295 req.
Aldaihani and Dessouky (2003)	mix with FRT	min. RC, min. CI	TW	tabu search	up to 155 req.
Melachrinoudis et al. (2007)	HF, MD	min. RC, min CI	TW	tabu search	up to 8 req.
Rekiek et al. (2006)	S, MD, HF	min. NV, min. CI	TW, RT, VA	grouping genetic algorithm	up to 164 req.
Jørgensen et al. (2007)	S	min. RC, min. RT, min. TWV, ...	TW, RT, RD	genetic algorithm, space-time nearest neighbor heur.	CL03

Bench. = Benchmark, CI = Customer Inconvenience, Con. = Constraints, FRT = Fixed Route Transit, HF = Heterogeneous Fleet, MD = Multi Depot, mRT = mean RT, NV = Number of Vehicles, Obj. = Objective(s), RD = Route Duration, RT = Ride Time, RC = Routing Cost, S = Service time, TW = Time Windows, TWV = TW Violation, VA = Vehicle Availability; The respective benchmark instances are described in Section 2.6.

by the same move, can constitute a new incumbent solution.

Summary State-of-the-art exact methods for the static DARP solve some instances with up to 96 requests to optimality (Ropke et al., 2007). However, the same limitation applies as mentioned above. The size of the test instance is not a very meaningful indicator; tightly constrained instances are easier to solve than those with less tight constraints and no standardized data set has been solved by the proposed approaches. In case of heuristic and metaheuristic methods, comparison becomes even harder since a large part of the solution procedures developed are motivated by real world problem situations. They differ regarding problem type (single and multi depot, homogeneous and heterogeneous fleet), constraints, and objective(s). Moreover, even when the same data sets are used different objectives are considered (see e.g. Cordeau and Laporte, 2003a; Jørgensen et al., 2007). Consequently, we can only state that in general heuristic methods run faster whereas metaheuristics usually outperform basic heuristic procedures with respect to solution quality.

Related work Dealing with the transportation of people, especially handicapped or elderly, research has also been dedicated to the comparison of dial-a-ride systems with other modes of

transportation. Early studies of dial-a-ride transportation systems are reported in Carlson (1976); Teixeira and Karash (1975). Elmberg (1978) test a robot dispatcher dial-a-ride system in Sweden. Daganzo (1984) compare fixed route transit systems with checkpoint dial-a-ride and door-to-door dial-a-ride systems. He concludes that most of the time either a fixed route system or door-to-door transportation is the appropriate choice. Belisle et al. (1986) investigate the impact of different operating scenarios on the quality of transportation systems for the handicapped. More recent studies comparing dial-a-ride and traditional bus systems by means of simulation are reported in Noda (2005); Noda et al. (2003). Shinoda et al. (2003) study the usability of dial-a-ride systems in urban areas. Mageean and Nelson (2003) evaluate telematics based demand responsive transport services in Europe. Palmer et al. (2004) analyze the impact of management practices and advanced technologies in the context of demand responsive transport systems. The impact of information flows, e.g. the percentage of real time requests or the length of the interval between the arrival of a new request and its requested pickup time window, is investigated in Diana (2006). Diana et al. (2006) study optimal fleet sizes with respect to predetermined service quality.

Research has also been dedicated to possible ways of computation time reduction. Hunsaker and Savelsbergh (2002), e.g., propose a fast feasibility check for the DARP. The proposed procedure can deal with time window, waiting time as well as ride time restrictions. Castelli et al. (2002) discuss three algorithms granting 2-opt-improvement feasibility.

2.5.3.2 Dynamic and stochastic variants

Less research has been dedicated to the domain of dynamic and/or stochastic DARP. Again, the term *dynamic* indicates that routing is done in real time; new requests pop up dynamically during the day and have to be scheduled into existing routes. Predominantly heuristic methods have been used to solve dynamic versions of the DARP.

Exact methods have not been explicitly developed for the dynamic DARP. A possible reason is that, in the context of dynamic routing, the concept of "optimal solutions" becomes debatable. However, in Psaraftis (1980) the static version of their algorithm is adapted to the dynamic case. In a cluster-first-route-second framework Colorni and Righini (2001), considering only the most urgent requests, solve the routing subproblems to optimality with a branch and bound algorithm. Also Caramia et al. (2002) iteratively solve the single vehicle subproblems to optimality, using a dynamic programming algorithm. Another cluster-first-route-second approach, applying dynamic programming in the routing phase, is reported in Dial (1995). New transportation requests are assigned to clusters according to least cost insertion. Teodorovic and Radivojevic (2000) propose a two-stage fuzzy logic based heuristic algorithm also following the cluster-first-route-second idea. The first approximate reasoning heuristic decides which vehicle a new request is assigned to. The second heuristic handles the adjustment of the vehicle's route.

Daganzo (1978) analyze three different insertion heuristics. In the first the closest stop

is visited next. The second consists in visiting the closest origin or the closest destination in alternating order. The third only allows the insertion of delivery locations after a fixed number of passengers have been picked up.

Horn (2002a) provide a software environment for fleet scheduling and dispatching of demand responsive services. The system can handle advance as well as immediate requests. New incoming requests are inserted into existing routes according to least cost insertion. A steepest descent improvement phase is run periodically. Also automated vehicle dispatching procedures, to achieve a good combination of efficient vehicle deployment and customer service, are included. The system was tested in the modeling framework LITRES-2 (Horn, 2002b). Another simulation environment to test solution methods for the dynamic DARP can be found in Fu (2002b).

Coslovich et al. (2005) propose a two-phase insertion heuristic. A simple insertion procedure allows for quick answers with respect to inclusion or rejection of a new customer. The initial solution is improved by means of local search using 2-opt arc swaps.

A dynamic DARP in a stochastic environment is considered in Xiang et al. (2007), taking into account travel time fluctuations, absent customers, vehicle breakdowns, cancellation of requests, traffic jams etc. To solve this complex problem situation the heuristic proposed in Xiang et al. (2006) is adapted to the dynamic case. Fu (2002a) develop a parallel insertion algorithm for a stochastic version of the DARP, considering time-dependent travel times.

Metaheuristic solution methods have not been explicitly developed for the dynamic DARP, since short response times are necessary in dynamic settings. Only one algorithm developed for the static version has been used to solve the dynamic case. The tabu search of Cordeau and Laporte (2003b) is adapted to the dynamic DARP by means of parallelization (Attanasio et al., 2004); and in Beaudry et al. (2008) it is used to solve a dynamic DARP with additional real world constraints. A tabu search and a hybrid method consisting of a greedy randomized adapted search procedure and a tabu search for probabilistic DARP is discussed in Ho and Haugland (2004).

Summarizing, only a limited number of solution algorithms have been proposed for the dynamic and the stochastic DARP. In case of the dynamic DARP, most of them are based on repeated calls to static solution routines. Comparison across the different methods proposed becomes a difficult task since a majority of the work presented is motivated by real life applications. Consequently, each solution methodology was tested on different data sets with varying problem specific characteristics. Whenever a solution method originally developed for the static case was used we refer to Section 2.5.3 of this chapter for further details on its performance in a static setting.

Table 2.3: Benchmark instances for VRPB

Literature Ref.	Type	Cust.	#	Characteristics	Abbr.
Golden et al. (1985)	VRPMB	55	1	based on instance 8 of Christofides and Eilon (1969), 10% bh.	Gal.85
Min (1989)	VRPSDP	22	1	real life instance	Min89
Goetschalckx and Jacobs-Blecha (1989)	VRPCB	25-150	62	25, 50 and 100% of the linehaul customers are bh.	GJ89
Gélinas et al. (1995)	VRPCB	25-100	45	TW, based on the first five problems proposed by Solomon (1987) for the VRPTW, 10, 30 50% bh.	GDDS95
Kontoravdis and Bard (1995)	VRPMB	100	27	based on the sets R2, C2 and RC2 (Solomon, 1987), $C^k = 250$, 50% bh.	KB95
Gendreau et al. (1996a)	TSPCB	100-1000	750	randomly generated points in the square $[0, 100]$, uniformly distributed, 10-50% bh.	GHL96
Toth and Vigo (1996b)	VRPCB	21-100	33	based on VRP instances available at the TSPLIB library, 50, 66 and 80% bh.	TV96
Thangiah et al. (1996)	VRPCB	250-500	24	TW, based on the sets R1 and RC1 (Solomon, 1987), 10, 30 and 50% converted into bh.	TPS96
Toth and Vigo (1999)	VRPCB	33-70	24	asymmetric, adapted from the real world instances used by Fischetti et al. (1994)	TV99
Salhi and Nagy (1999)	VRPMB	20-249	SD:42 MD:33	based on SD instances (Christofides et al., 1979) and MD instances (Gillett and Johnson, 1976), adapted by defining 10, 25 and 50% of the customers as bh.	SN99a
	VRPSDP	20-249	SD:28 MD:22	same instances, adapted by splitting every customer's demand into a demand and a supply part	SN99b
Gendreau et al. (1999)	VRPSDP	6-261	1308	partly based on VRP instances from the literature, partly randomly generated	GLV99
Dethloff (2001)	VRPSDP	50	40	randomly generated, 2 geographical scenarios: (1) uniformly distributed customer locations over the interval $[0, 100]$, (2) more urban configuration; the pickup amount has at least half the size of the delivery amount	Det01

= number of instances, Abbr. = Abbreviation, bh. = backhaul customers, Cust. = number of Customers per instance, MD = Multi Depot, SD = Single Depot

2.6 Benchmark instances

In order to provide the interested researcher with some information on available benchmark instances, we decided to dedicate this section to a brief description of the data sets used in the literature. Table 2.3 and 2.4 provide the following information in chronological order. In the first column the according literature reference is given. Column two states the VRPB or VRPPD type the respective instances were designed for. Columns three and four give the size of the smallest and the size of the largest instance, in terms of number of customers, and the number of instances provided, respectively. In column five a brief description of the instances can be found. Column six contains the abbreviations used in this chapter or in

Table 2.4: Benchmark Instances for VRPPD

Literature Ref.	**Type**	**Req.**	**#**	**Characteristics**	**Abbr.**
Shang and Cuff (1996)	PDPTW, transfers	159	1	real world data	SC96
Nanry and Barnes (2000)	PDPTW	13-50	43	based on VRPTW instances (Solomon, 1987), optimal solution schedules by procedure of Carlton (1995), customers randomly paired	NB00
Renaud et al. (2000)	SPDP	25-249	108	based on 36 TSPLIB instances (Reinelt, 1991), for each pickup a delivery chosen among the 5 (10) closest or all unselected neighbors.	RBO00
Li and Lim (2001)	PDPTW	50	56	based on those of (Solomon, 1987), customers appearing on the same route in a solution of the VRPTW, using the solution procedure of Li et al. (2001), were randomly paired;	LL01
		100-500	298	extended data set	LL01+
Toth and Vigo (1996a)	DARP	276-312	5	real life data, Municipality of Bologna	TV96a
Cordeau and Laporte (2003b)	DARP	24-144	20	randomly generated around seed points, half of the requests have a tight TW at the origin, half a tight TW at the destination, 10 instances with narrow, 10 with wider TW.	CL03
Mitrović-Minić and Laporte (2004)	dyn. PDPTW	100-1000	40	ten hours service period, 60 x 60 km^2 area, vehicle move at 60 km/h, requests occur according to a continuous uniform distribution, no requests are known in advance	ML04
Cordeau (2006)	DARP	16-48	24	randomly generated; 12 instances with $C = 3$, unit user demand and $\bar{L} = 30$; 12 instances with $C = 6$, varying user demand and $\bar{L} = 60$.	Cor06
Ropke et al. (2007)		16-96	42	extended data set	Cor06+
Ropke et al. (2007)	PDPTW	30-75	40	randomly generated as described in (Savelsbergh and Sol, 1998).	RCL07

= number of instances, Abbr. = Abbreviation, bh. = backhaul customers, C = vehicle capacity, $\bar{L}$ = maximum ride time, MD = Multi Depot, Req. = approximate number of Requests of each instance, SD = Single Depot, TW = Time Window

the two-part survey (Parragh et al., 2008a,b).

In case of the VRPCB subclass the benchmark data sets most often used in the literature are GJ89 and TV96. The most recent new best results have been presented by Brandão (2006) and Ropke and Pisinger (2006b), two metaheuristic approaches, outperforming earlier results by Osman and Wassan (2002).

Regarding the VRPMB and the VRPSDP, the single depot instances, SN99a and SN99b, have been most often solved in the literature. The most recent new best results for these two data sets are presented in Ropke and Pisinger (2006b), and Nagy and Salhi (2005) for the second half of the SN99b data set. Also Tang Montané and Galvão (2006) report good results for parts of the SN99b instances.

The data set predominantly used to asses the performance of PDPTW is the one proposed by Li and Lim (2001). The latest new best results for both the primary (LL01) and the extended data set (LL01+) can be found in Ropke and Pisinger (2006a) and Bent and van Hentenryck (2006).

In contrast to the field of PDP, solution methods developed for the DARP have not been tested on standardized benchmark instances. This might be due to the fact that most methods vary considerably with respect to the constraints considered as well as the objectives minimized. However, since data sets for rather standard problems settings do exist now, this might change in the near future. Data sets CL03 and Cor06+ will be used to test the performance of the heuristic methods developed in Chapters 3 and 4, respectively. The test instances applied in Chapters 5 and 6 are based on data set Cor06.

3 Solving a simplified problem version

3.1 Introduction

The overall aim of this book is to solve a real world static ambulance routing problem, as, e.g., encountered by the ARC. In Chapter 2, solution methods for different DARP versions and related problems have been discussed. As already pointed out previously, ambulance routing problems are usually modeled as DARP. The first step towards the aim of solving the real world problem is to deal with a simplified version. This simplified version will include a number of identical transportation requests, served by a given fleet of identical vehicles. Each transportation request is associated with a pickup and a delivery location. Since people are transported, service oriented criteria are considered in terms of time windows and maximum user ride times. Another approach to deal with user inconvenience is investigated in the subsequent chapter: a second objective function is introduced. In this chapter, user inconvenience will only be considered in terms of constraints. The objective will be the minimization of the total routing costs.

3.2 Related Work

When looking at the literature, due to the application oriented character of this problem, no "standardized" problem definition can be found. The objectives applied range from the maximization of the number of patients served to the minimization of user waiting time or routing costs. The constraints considered are usually tailored to the specific problem situation. Let us give some examples. Baugh et al. (1998) design a simulated annealing algorithm minimizing the weighted sum over total routing costs, time window violations (user inconvenience), and the number of vehicles used. Toth and Vigo (1997b), on the other hand, consider a heterogeneous fleet, service times, multiple depots, time windows and maximum user ride times, while minimizing the total distance traveled by the vehicles using a tabu thresholding algorithm. Aldaihani and Dessouky (2003) solve a mix between a DARP, i.e. transportation on demand, and a fixed route transit system by means of a tabu search heuristic. Customer inconvenience in terms of the total ride time of all the passengers and the total distance traveled by the on demand vehicles are minimized. Time windows are considered as a hard constraint. Soft time windows are considered by Jørgensen et al. (2007). They propose a Genetic Algorithm (GA) to solve the DARP, minimizing the

weighted sum of customer transportation time, excess user ride time with respect to direct ride time, customer waiting time, time window violations, excess user ride time with respect to maximum ride time, and excess work time. This clearly shows that the DARP lacks a generic problem formulation. For further references we refer to Chapter 2 of this book. Furthermore, we refer to Paquette et al. (2007) for an overview of the different quality of service criteria that have been applied in dial-a-ride systems.

A rather, in our opinion, general problem definition, suitable to conduct the first step into the DARP research domain, together with a benchmark data set has been introduced by Cordeau and Laporte (2003b). They consider time windows, a maximum user ride time limit, and a maximum route duration limit, while minimizing total routing costs. Results obtained with a Tabu Search (TS) heuristic are reported. In this chapter we will develop a competitive VNS heuristic (Mladenovic and Hansen, 1997) for the problem at hand, as defined in further detail in Section 3.3. Also a scientific article resulted from the research work dedicated to the development of a VNS for the DARP (Parragh et al., 2009c). Many passages of this chapter are taken from this article.

A popular question in the vehicle routing community, working on metaheuristic methods, is to demand a reason for choosing a certain solution method among the different solution paradigms available. In case of this book several reasons can be given. First, VNS belongs to the more recent metaheuristic concepts. It has only been proposed about a decade ago. Therefore, when, e.g., compared to the plethora of articles dealing with TS algorithms, only a much smaller number of articles dealing with VNS can be found in the literature. Second, many recent successful implementations exist. VNS has shown to be competitive when used to solve vehicle routing problems such as, among others, the vehicle routing problem with time windows (Bräysy, 2003), the multi-depot vehicle routing problem with time windows (Polacek et al., 2004), the periodic vehicle routing problem (Hemmelmayr et al., 2009), or the pickup and delivery traveling salesman problem with last-in-first-out loading restrictions (Carrabs et al., 2007). Third, many recent applications of VNS in the vehicle routing domain stem from our research group. The expertise thus acquired and easily accessible during the development of this book has certainly been another reason for choosing VNS. And last but not least, to the best of our knowledge this will be the first time VNS will be applied to the static DARP.

The reminder of this chapter is organized as follows. First, a more detailed problem definition is given. This is followed by the description of the proposed metaheuristic algorithm. Finally, computational results are reported and compared to those of the TS of Cordeau and Laporte (2003b), and, using an adapted objective function, to the GA of Jørgensen et al. (2007).

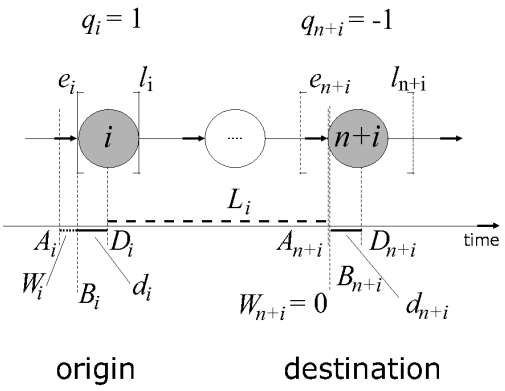


Figure 3.1: An inbound request

3.3 Problem definition

The DARP is modeled on a complete directed graph $G = (V, A)$ where V is the set of all vertices and A the set of all arcs. For each arc (i, j) a non-negative travel cost c_{ij} and a non-negative travel time t_{ij} is considered. A total amount of n customer requests, each consisting of a pickup and delivery vertex pair $\{i, n+i\}$, are to be served by a set K of m identical vehicles with a capacity of C. All vehicles are stationed at a central depot denoted by 0 (origin depot) and $2n+1$ (end depot). Furthermore, every route has to be completed within a maximum route duration limit T. At every pickup vertex a certain number of passengers ($q_i > 0$) waits to be transported. The demand at every delivery vertex is equal to $q_{n+i} = -q_i$. Either the pickup vertex (origin) or the delivery vertex (destination) is associated with a time window $[e_i, l_i]$, depending on the type of request. Outbound requests, e.g. from home to the hospital, have a tight time window on the destination. Inbound requests, e.g. from the hospital back home, have a tight time window on the origin. An inbound request is illustrated in Figure 3.1. The vertex that is associated with the tight time window is also referred to as the *critical vertex* of a request (Cordeau and Laporte, 2003b). The service time for loading or unloading operations at each vertex is denoted by d_i. In order to limit user inconvenience caused by long detours, a maximum user ride time $\bar{L}$ has to be respected. The arrival time at a vertex is denoted as A_i. Beginning of service at each vertex i (on vehicle k), denoted as $B_i^{(k)}$, and the load when leaving vertex i (with vehicle k), denoted as $Q_i^{(k)}$, are to be determined. The waiting time W_i at a vertex i is $W_i = B_i - A_i$. The ride time of a client is calculated by $L_i^{(k)} = B_{n+i}^{(k)} - (B_i^{(k)} + d_i)$, i.e. the time he/she spends on board the vehicle. The departure time is given by $D_i = B_i + d_i$. The duration of a route k is calculated as $B_{2n+1}^k - B_0^k$, where B_{2n+1}^k (B_0^k) denotes the beginning of service at the depot $2n+1$ (0) by vehicle k. The objective is to minimize total routing

costs. It can be formulated as follows (Cordeau, 2006),

$$\sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} x_{ij}^k, \tag{3.1}$$

$$\tag{3.2}$$

subject to,

$$\sum_{i \in V} \sum_{k \in K} x_{ij}^k = 1 \qquad \forall j \in P, \tag{3.3}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{n+i,j}^k = 0 \qquad \forall i \in P, k \in K, \tag{3.4}$$

$$\sum_{j \in V} x_{0j}^k = 1 \qquad \forall k \in K, \tag{3.5}$$

$$\sum_{j \in V} x_{ji}^k - \sum_{j \in V} x_{ij}^k = 0 \qquad \forall i \in P \cup D, k \in K, \tag{3.6}$$

$$\sum_{i \in V} x_{i,2n+1}^k = 1 \qquad \forall k \in K, \tag{3.7}$$

$$x_{ij}^k = 1 \Rightarrow \quad B_j^k \geq B_i^k + d_i + t_{ij} \qquad \forall i \in V, j \in V, k \in K \tag{3.8}$$

$$x_{ij}^k = 1 \Rightarrow \quad Q_j^k \geq Q_i^k + q_j \qquad \forall i \in V, j \in V, k \in K \tag{3.9}$$

$$L_i^k = B_{n+i}^k - (B_i^k + d_i) \qquad \forall i \in P, k \in K, \tag{3.10}$$

$$B_{2n+1}^k - B_0^k \leq T \qquad \forall k \in K, \tag{3.11}$$

$$e_i \leq B_i^k \leq l_i \qquad \forall i \in V, k \in K \tag{3.12}$$

$$t_{i,n+i} \leq L_i^k \leq \bar{L} \qquad \forall i \in P, k \in K \tag{3.13}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{C, C + q_i\} \qquad \forall i \in V, \tag{3.14}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i \in V, j \in V, k \in K. \tag{3.15}$$

The objective function (3.1) minimizes total routing costs. Constraints (3.3)–(3.7) ensure that each request is served exactly once; that each origin is visited by the same vehicle as its destination; and that each vehicle's route starts at the origin depot and ends at the destination depot. Load propagation and consistency with respect to time variables is guaranteed by inequalities (3.8) and (3.9). User ride times are set by equalities (3.10). Maximum route duration restrictions, time window compliance, and maximum ride time limits are enforced by constraints (3.11)–(3.13). Finally, capacity limits are imposed by (3.14).

Algorithm 3.1 *heurVNS*

1: // *pre-processing*
2: do graph pruning and time window tightening;
3: // *initial solution*
4: generate $\int_{init}$;
5: set $\int := \int_{init}$; set $\hat{k} := 1$;
6: **repeat**
7: // *shaking*
8: randomly compute $\int'$ in $\hat{N}_{\hat{k}}(\int)$;
9: // *local search*
10: **if** $\hat{c}(\int') < 1.02\,\hat{c}(\int)$ or $p_{rand} < 0.01$ **then**
11: apply local search to $\int'$ yielding $\int''$;
12: **else**
13: set $\int'' := \int'$;
14: **end if**
15: // *move or not*
16: **if** $\hat{f}(\int'') < \hat{f}(\int)$ or meets other acceptance criteria **then**
17: **if** $\hat{c}(\int'') \geq 1.05\,\hat{c}(\int)$ **then**
18: apply local search to $\int''$;
19: **end if**
20: set $\int := \int''$; set $\hat{k} := 0$;
21: **end if**
22: **if** $\int''$ is feasible and better than $\int_{best}$ **then**
23: set $\int_{best} := \int''$;
24: **end if**
25: set $\hat{k} := (\hat{k} \mod \hat{k}_{max}) + 1$;
26: **until** some *stopping criterion* is met
27: **return** $\int_{best}$;

3.4 Solution framework

As indicated, we will solve the DARP by means of a VNS based heuristic. In general, VNS departs from an *initial solution* $\int_{init}$, which is equal to the first incumbent solution $\int$. Then, in every iteration a random solution $\int'$ is generated in the current neighborhood $\hat{N}_{\hat{k}}(\int)$ of $\int$ (*shaking*). A subsequent *local search* step applied to $\int'$ yields $\int''$. If $\int''$ is better than $\int$, it replaces $\int$ and the search continues with the first neighborhood $\hat{k} = 1$. If $\int''$ is worse, $\int$ is not replaced and the next (larger) neighborhood is used in the subsequent iteration $\hat{k} = \hat{k} + 1$ (*move or not*). A maximum number of neighborhoods $\hat{k}_{\max}$ has to be defined. Whenever $\hat{k}_{\max}$ is attained, the search continues with the first neighborhood ($\hat{k} = 1$). This is repeated until some *stopping criterion* is met.

In our case, also ascending moves, fulfilling pre-specified acceptance criteria, are permitted. This means that also deteriorating solutions may become incumbent solutions with a certain probability. Furthermore, like Cordeau and Laporte (2003b), we allow intermediate infeasible solutions (infeasibilities are penalized). Therefore, in addition to $\int$, we also keep

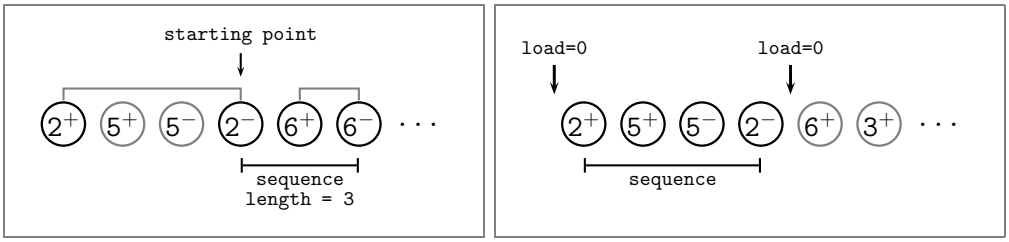
track of the best feasible solution $\int_{best}$ encountered so far. The local search step has also been subject to modifications; and, in order to reduce the search space, the graph pruning and time window tightening techniques, described by Cordeau (2006), are applied prior to starting the optimization procedure. In Algorithm 3.1 the general framework of our implementation is given. It will be referred to as *heurVNS*. The different design elements are described in further detail in the following paragraphs.

3.4.1 Pre-processing

Before the optimization procedure is started, we apply graph pruning and time window tightening techniques, as described by Cordeau (2006), see Algorithm 3.1 line 2. First, time windows are strengthened. At all vertices that are not associated with a tight time window an artificial time window is generated. In case of an outbound request the time window at the origin i is set to $e_i = \max\left\{0, e_{n+i} - \bar{L} - d_i\right\}$ and $l_i = \min\left\{l_{n+i} - t_{i,n+i} - d_i, \hat{H}\right\}$, where $\hat{H}$ is the end of the planning horizon. In case of an inbound request the destination time window is set to $e_{n+i} = e_i + d_i + t_{i,n+i}$ and $l_{n+i} = \min\left\{l_i + d_i + \bar{L}, \hat{H}\right\}$. Then, all those arcs are removed that cannot be part of a feasible solution. These are all arcs connecting the origin depot and a destination, the destination with its origin, and an origin with the destination depot. Furthermore, request pairs are checked for compatibility regarding time windows and ride time limits. All arcs connecting incompatible parts of two requests are eliminated. For the complete procedure we refer to Cordeau (2006).

3.4.2 Initial solution

To generate the first incumbent solution $\int$ the following procedure is used (Algorithm 3.1 line 4). First, requests are sorted according to some artificial beginning of service. These are set randomly within the time window of the critical vertex (Cordeau and Laporte, 2003b). Then, all routes are initialized with one request, using the first m requests on the list. After that, requests are inserted at the end of one of these partial routes in the order they appear on the list. Which route is selected is chosen according to one of four minimum distance criteria, comparing the distance between either origin or destination of the last request on each route and the next request on the list. Criterion one considers the two origins, criterion two the origin of the last request and the destination of the next; criterion three inspects the destination of the last and the origin of the next request and criterion four uses the two destinations to determine the minimum distance. Which criterion is chosen is selected randomly for each request individually. This is repeated until all requests have been inserted into some route.



(a) Swap sequence

(b) Zero split sequence

Figure 3.2: Different sequences

3.4.3 Shaking

We use four different types of neighborhoods (Algorithm 3.1 line 8). The first swaps two sequences of requests, the second is based on the move operator, the third applies the idea of ejection chains (Glover, 1996), and the fourth redistributes requests forming a "natural" sequence to different routes.

Swap neighborhood (S) In the first neighborhood two sequences of vertices are exchanged. First, two routes are chosen randomly. Then, on each route a sequence to be swapped is selected. This is done by, first, randomly selecting the starting vertices of the two sequences and then, by randomly selecting the length of each sequence — the maximum sequence length is the size of this neighborhood. Finally, all requests forming the respective sequences are deleted from their original routes and inserted, one by one in the best possible way, again using the notion of critical vertices, into the other route. A swap of size three, e.g., consists in exchanging two sequences of at most length three.

Note that for each origin (destination) within the selected sequence the corresponding destination (origin) has to be moved as well, even if it is not part the sequence. This is depicted in Figure 3.2(a). Let i^+ denote the origin and i^- the destination of request i. Here, both vertices forming request 6 are part of the selected sequence. In case of request 2 only the destination is part of the sequence. However, also its origin denoted by 2^+ will be moved.

Move neighborhood (M) The second neighborhood consists in moving requests from their original routes to other routes. The requests to be moved are selected randomly across all requests. The insertion route is either selected randomly or the "closest" route, excluding the request's original route is chosen. Closeness is measured by the total spatial distance between the request to be inserted and the vertices constituting the insertion route r $\bar{d} = \sum_{i \in r}(t_{ij} + t_{i,n+j} + t_{ji} + t_{n+j,i})$, j refers to the request to be inserted. If some t_{ij} corresponds to a forbidden arc due to the applied pre-processing steps it is not counted. However, a correction term χ is used. It is set to four times the average number of vertices per route

$\chi = 4\frac{2n}{m}$. Let $\tilde{n}$ give the number of not counted forbidden arcs, $\bar{d}$ is set to $\bar{d} := \bar{d}/(\chi - \tilde{n})$ such that the more forbidden arcs the larger the total distance calculated. Random selection and "closeness" selection are chosen with a probability of 0.5 each. Requests are inserted in the best possible way into their new routes, using the notion of critical vertices as depicted by Cordeau and Laporte (2003b). Thus, first, the critical vertex is inserted at the best possible position, and then the other vertex is inserted at the best possible position with respect to the critical one. The size of this neighborhood is defined as the maximum number of requests that are moved. A move of size two, e.g., consists in moving at most two requests.

Chain neighborhood (C) The third neighborhood applies the ejection chain idea. In contrast to its original version, the number of routes is a parameter. In a first step, two routes are chosen. From the first route a sequence of requests (selected as in the swap neighborhood) is moved to the second route. In a second step, from the sequence, that decreases the evaluation function value of the second route the most, is moved to a third route (it may also be the first route). The second step is repeated until the maximum number of sequences moved has been reached. All insertions of sequences into their new routes are done one by one in the best possible way using the notion of critical vertices. All routes are selected randomly. The neighborhood size represents at the same time the number of sequences moved and the maximum sequence length. E.g., a chain neighborhood of size three consists of moving a random sequence of at most three requests from its original route to a second route. From the second route the sequence of at most three requests that decreases the route's evaluation function value the most is moved to a third route. Finally, from this third route again the maximum savings sequence with a length of at most three is moved to a fourth route. Thus, three sequences are moved and three plus one routes are affected.

Zero split neighborhood (Z) The fourth neighborhood is a parameterless neighborhood. As in the two previous neighborhoods, a sequence of requests is moved. However, this sequence is chosen in a different way. Within every route "natural" sequences of requests can be defined. These sequences lie between two arcs where the vehicle load is equal to zero, i.e. the vehicle is empty (see Figure 3.2(b)). Every route contains at least two such arcs. These are the arcs from the depot to the first origin along the route and from the last destination back to the depot. In addition, we discovered that routes quite often contain more than only one sequence of this type. This led to the idea of the zero split neighborhood. In this neighborhood a random sequence of these natural sequences is removed from a random route. All requests part of this sequence are then reinserted one by one into different routes chosen at random. Insertion is again done one by one in the best possible way using the notion of critical vertices.

Neighborhood sequel These four neighborhoods are applied in the following order (the number given in addition to the neighborhood abbreviation indicates the neighborhood size): S1 – M1 – C1 – S2 – M2 – C2 – S3 – M3 – C3 – S4 – M4 – C4 – S5 – M5 – C5 – S6 – M6 – C6 – Z. This means that $\hat{N}_1$ = M1, $\hat{N}_2$ = C1, etc. (see Algorithm 3.1 line 8). Thus, a total of $\hat{k}_{\max} = 19$ different neighborhoods are used. The idea of the VNS is to increase the neighborhood searched from one neighborhood to the other. To mimic this behavior the above sequence was chosen: S is ordered before M because in S at most two routes may be affected, while in M as many routes as requests may be affected. M is ordered before C because in C the same number of routes but even more requests may be moved.

3.4.4 Local search

While in the shaking step mainly inter-tour neighborhood operators are applied, the local search step of *heurVNS* (Algorithm 3.1 line 11) is of the intra-tour first improvement type. It moves along each route as follows. In a first step, the first origin and its corresponding destination are removed from the route. Then, the critical vertex of the two is re-inserted at the first possible position with respect to time window feasibility. Thereafter, the non-critical vertex is inserted at the first possible position with respect to the critical vertex. If this positioning improves the route's evaluation function value, the two vertices are kept at their new positions and the search continues with the new first origin on the route. Otherwise, the non-critical vertex is tried at the next possible position. In case of an improvement the search proceeds as before. Otherwise, the non-critical vertex is moved to the next possible position. This is repeated until no other insertion position for the non-critical vertex exists. In this case, the critical vertex is moved to the next possible position, and the non-critical vertex is again inserted in accordance with the critical one. This is repeated until either an improved positioning is encountered or the last possible positioning has been reached. In this case, the vertex pair is kept at its original position, and the next origin on the route is searched. The procedure ends as soon as the last origin on the route and its destination have been tried at all possible positions, and no improvement has been found.

3.4.5 Local search frequency

In standard VNS implementations, the local search step is conducted after every shaking step. However, in our case we always insert the requests to be moved or swapped one by one in the best possible way (best insertion), using the notion of critical vertices, as defined by Cordeau and Laporte (2003b). Thus, request insertion already includes some kind of local search for the inserted request. Consequently, the local search step is not absolutely necessary.

In order to avoid unnecessary calls to the time consuming local search heuristic, we have chosen to use local search only if $\int'$ is a *promising* solution. A *promising* solution

is a solution that has some potential to become a new incumbent solution. After several experiments, we defined as *promising* a solution $\int'$ that is at most 2% worse than the incumbent $\int$. Since the objective of the DARP is to minimize total routing costs $\hat{c}(\int)$, a promising solution is characterized by $\hat{c}(\int') < 1.02\,\hat{c}(\int)$. In order to introduce another element of diversification, every solution has a 1% chance to be subject to local search based improvement, see Algorithm 3.1 line 9.

Furthermore, every solution $\int''$ that is accepted to become the new incumbent solution is also subject to local search based improvement; given that its routing costs are at least 5% worse than those of the current incumbent, see Algorithm 3.1 line 17.

Surprisingly, preliminary tests showed that a 5% limit is better than a limit of 2%. A limit of 2% would mean that every solution constituting a new incumbent solution undergoes local search; either due to its *promising* solution property or because it will be the new incumbent solution. A possible reason why the 5% limit worked better than the 2% one is increased diversification: a solution that has not been locally optimized may, in some cases, provide better options regarding the removal and reinsertion of a request in the subsequent iteration than a locally optimized one.

3.4.6 Move or not

In order to decide whether the search should move to the new solution $\int''$ or not (see Algorithm 3.1 line 15), i.e. if $\int''$ replaces $\int$, as in Hemmelmayr et al. (2009) and Ropke and Pisinger (2006a) a simulated annealing (Kirkpatrick et al., 1983) type criterion is used. Let $\hat{f}(\int)$ denote the evaluation function value of solution $\int$. In the beginning $\int''$ can only replace $\int$ if $\hat{f}(\int'') < \hat{f}(\int)$. As soon as the first feasible solution has been found, also deteriorating solutions may become incumbent solutions with probability $\exp(-[\hat{f}(\int'') - \hat{f}(\int_{best})]/\bar{t})$. The temperature $\bar{t}$ is then set such that if $[\hat{f}(\int'')/\hat{f}(\int_{best})] - 1 = 0.005$, $\int''$ is accepted with a probability of 0.2. Thereafter, $\bar{t}$ is linearly decreased until 0, i.e. when the maximum number of iterations has been attained $\bar{t} = 0$.

3.4.7 Solution evaluation

Following Cordeau and Laporte (2003b) we penalize load violation $\hat{q}(\int)$, duration violation $\hat{d}(\int)$, time window violation $\hat{w}(\int)$, and ride time violation $\hat{t}(\int)$ in the evaluation function. Load violation is computed as $\hat{q}(\int) = \sum_{i=1}^{2n}(Q_i - C)^+$, where $x^+ = \max\{0, x\}$, duration violation as $\hat{d}(\int) = \sum_{k=1}^{m}(B^k_{2n+1} - B^k_0 - T)^+$, time window violation as $\hat{w}(\int) = \sum_{i=1}^{2n}(B_i - l_i)^+$, and ride time violation as $\hat{t}(\int) = \sum_{i=1}^{n}(L_i - \bar{L})^+$. For the according notation see Section 3.3. Let now $\hat{c}(\int)$ denote the total routing costs of all vehicles, which is the sum of the costs c_{ij} associated with the arcs (i, j) traversed by the vehicles, the following evaluation

function is applied,

$$\hat{f}(\int) = \hat{c}(\int) + \hat{\alpha}\hat{q}(\int) + \hat{\beta}\hat{d}(\int) + \hat{\gamma}\hat{w}(\int) + \hat{\tau}\hat{t}(\int). \tag{3.16}$$

The penalty terms for load, duration, time window, and ride time violation are given by $\hat{\alpha}$, $\hat{\beta}$, $\hat{\gamma}$, and $\hat{\tau}$, respectively. These are set to $\hat{\alpha} = \hat{\beta} = \hat{\gamma} = \hat{\tau} = 1$ at the beginning of the search. Similar to Cordeau and Laporte (2003b), their values are then adjusted in a dynamic way. Every time a new incumbent solution $\int$ is identified, the penalty parameters are either increased or decreased. In case $\int$ violates a penalized constraint, the corresponding penalty term is multiplied by $\hat{\delta}$. If, e.g., $\int$ violates a capacity constraint $\hat{q}(\int) > 0$, then $\hat{\alpha} := \hat{\alpha}(1+\hat{\delta})$. Whereas if $\hat{q}(\int) = 0$, then $\hat{\alpha} := \hat{\alpha}/(1+\hat{\delta})$. The value of $\hat{\delta}$ is randomly chosen between $\underline{\delta} = 0.05$ and $\overline{\delta} = 0.1$, every time a new incumbent is identified. Using different values for $\hat{\delta}$ decreases the chances of cycling, and works as a diversification mechanism. Note that a solution $\int''$ can only become $\int_{best}$ if and only if $\hat{q}(\int'') = \hat{d}(\int'') = \hat{w}(\int'') = \hat{t}(\int'') = 0$, see Algorithm 3.1 line 22ff.

In order to set the beginning of service in the best possible way, such that route duration is minimum and ride time limits are respected where possible, the route evaluation scheme introduced by Cordeau and Laporte (2003b) is applied. It is based on the forward time slack F_i, defined by Savelsbergh (1992), adjusted to the DARP,

$$F_i = \min_{i \le j \le q} \left\{ \sum_{i < p \le j} W_p + \left(\min\left\{l_j - B_j, \bar{L} - P_j\right\}\right)^+ \right\}, \tag{3.17}$$

where W_p denotes the waiting time at vertex p, q the last vertex on the route, and P_j the ride time of the user whose destination is $j \in \{n+1, \ldots, 2n\}$ given that $j - n$ is visited before i on the route; $P_j = 0$ for all other j. The slack at vertex j is the cumulative waiting time until j, plus the minimum of the difference between the end of the time window and the beginning of service at j, and the difference between the maximum user ride time and P_j. The forward time slack at vertex i is the minimum of all slacks between i and q. F_0 thus gives the maximum amount of time by which the departure from the origin depot can be delayed (with equal time window violations and smaller or equal ride time violations) to yield a modified route of minimal duration. Furthermore, Cordeau and Laporte (2003b) observed that delaying the departure by $\sum_{0<p<q} W_p$ does not affect the arrival time at q, whereas delaying the departure by more will only increase the arrival time at q by as much. Consequently, the departure from the depot should only be delayed by at most $\min\left\{F_0, \sum_{0<p<q} W_p\right\}$.

The forward time slack can also be applied to delay B_i at each origin vertex such that the ride time of the corresponding request is reduced. This reasoning led Cordeau and Laporte (2003b) to an eight step evaluation scheme. We use the same eight step scheme. However,

Algorithm 3.2 Eight step evaluation scheme

1. Set $D_0 := e_0$.
2. Compute A_i, W_i, B_i, D_i and Q_i for each vertex i on the route.
   If some $B_i > l_i$ or $Q_i > C$ GO TO STEP 8.
3. Compute F_0.
4. Set $D_0 := e_0 + \min\left\{F_0, \sum_{0<p<q} W_p\right\}$.
5. Update A_i, W_i, B_i and D_i for each vertex i on the route.
6. Compute L_i for each request on the route.
   If all $L_i \leq \bar{L}$ GO TO STEP 8.
7. For every vertex j that is an origin
   a) Compute F_j.
   b) Set $W_j := W_j + \min\left\{F_j, \sum_{j<p<q} W_p\right\}$; $B_j := A_j + W_j$; $D_j := B_j + d_j$
   c) Update A_i, W_i, B_i and D_i for each vertex i that comes after j in the route.
   d) Update L_i for each request i whose destination is after j.
      If all $L_i \leq \bar{L}$ of requests whose destinations lie after j GO TO STEP 8.
8. Compute changes in violations of vehicle load, duration, time window and ride time constraints.

in case of no more reparable violations or irreparable violations, we stop and use the current approximated violations in the evaluation function. Let D_i denote the departure time from vertex i and A_i the arrival time at vertex i, this yields the evaluation scheme given in Algorithm 3.2.

The eight step procedure first minimizes time window constraint violations in steps (1) and (2). In addition to the original version, also the vehicle load at each vertex on the route is calculated. In case either of the two (time window or loading restriction) is violated, an irreparable violation is encountered and we proceed with step (8). Time window violations cannot be repaired hereafter since all B_i are set to the earliest feasible point in time, which is, coming from vertex j, $D_j := (B_j + d_j)$, $A_i := (D_j + t_{ij})$, $W_i := (e_i - A_i)^+$, and $B_i := (A_i + W_i)$. In steps (3) – (6) route duration is minimized, without increasing time window violations. Here, a ride time feasibility step is inserted. In case all ride times are already feasible, we can skip step (7) and go directly to step (8). In step (7) ride times are sequentially minimized. Here again the feasibility of the ride times of those requests whose destinations lie behind j is checked. In case all of them are feasible, go to step (8).

3.4.8 Stopping criterion

Initially the stopping criterion in *heurVNS* was set to a limit on the maximum number of iterations. One iteration corresponds to constructing one new solution (shaking) and to optimize it locally in the local search step. However, especially in case of small problem instances, the best (possibly optimal) solution is usually found very quickly. Therefore, a second stopping criterion has been defined. This criterion is the maximum number of

iterations between the identification of two new global best solutions $\int_{best}$. If during 2x10^6 iterations no new global best solution can be generated, the search terminates and returns $\int_{best}$. Summarizing, the search either stops as soon as the maximum total number of iterations (10^7) has been attained, or if during 2x10^6 iterations no new global best solution can be identified.

3.5 Computational experiments

The VNS was implemented in C++. All experiments were conducted on a Pentium D computer with 3.2 GHz. Our algorithm was tested on the benchmark data set of Cordeau and Laporte (2003b). Based on this data set, two different versions of the DARP have been considered in the literature.

In the "standard version" of the DARP, as introduced at the beginning of this chapter, all constraints are hard constraints and the objective is the minimization of total routing costs, which corresponds to minimizing the total travel distance; since $c_{ij} = t_{ij}$ for all i and j. This version was also considered by Cordeau and Laporte (2003b). We compare the results of *heurVNS* to the best results obtained by the TS developed by the same authors.

In the "modified version" introduced by Jørgensen et al. (2007) a different objective function was selected, namely a weighted combination of routing costs, total route duration, user ride time, user waiting time, and penalties for time limit violations is minimized. All time related constraints are thus considered to be soft.

In the following, first the test data set is described. Then, results obtained by *heurVNS* are compared to those of the TS by Cordeau and Laporte (2003b). Finally, results for the modified version and comparisons to the GA by Jørgensen et al. (2007) are presented.

3.5.1 Test instances

Cordeau and Laporte (2003b) proposed a data set of 20 randomly generated instances, containing between 24 and 144 requests (see also Chapter 2, Section 2.6 CL03). In each instance the first $\frac{n}{2}$ requests were defined as inbound while the remaining $\frac{n}{2}$ were defined as outbound requests. Origin and destination locations were generated using the procedure introduced in Cordeau et al. (1997). For each vertex a service time $d_i = 10$ was set. The load was set to $q_i = 1$ if $i \in \{1, \ldots, n\}$ and to $q_i = -1$ if $i \in \{n+1, \ldots, 2n\}$. Routing costs c_{ij} and travel times t_{ij} from a vertex i to a vertex j are equal to the Euclidean distance between these two vertices. Each vertex is associated with a time window $[e_i, l_i]$. Origins of outbound requests and destinations of inbound requests have "no" time window. Thus, their time windows were set to the length of the planning horizon $[0, 1440]$. The data set was split into two groups. In group (a) narrow time windows were set while in group (b) wider time windows were created. For all instances route duration was set to $T = 480$, vehicle capacity

Table 3.1: *heurVNS* vs. TS

				heurVNS (5 runs)					all tests	
	m	n	TS[a]	avg.	(%)	best	(%)	CPU[b]	best	(%)
R1a	3	24	*190.02*	**190.02**	0.00	**190.02**	0.00	8.28	**190.02**	0.00
R2a	5	48	302.08	302.10	0.01	**301.34**	-0.25	19.59	**301.34**	-0.24
R3a	7	72	532.08	537.18	0.96	533.86	0.33	32.11	**532.00**	-0.02
R4a	9	96	572.78	576.40	0.63	573.98	0.21	74.98	**570.25**	-0.44
R5a	11	120	636.97	639.50	0.40	637.80	0.13	158.22	**628.11**	-1.39
R6a	13	144	801.40	814.47	1.63	**798.17**	-0.40	184.61	**794.06**	-0.92
R7a	4	36	*291.71*	294.34	0.90	292.80	0.37	10.54	**291.71**	0.00
R8a	6	72	494.89	**494.64**	-0.05	**487.84**	-1.42	46.08	**487.84**	-1.42
R9a	8	108	672.44	**666.99**	-0.81	**661.33**	-1.65	203.57	**658.31**	-2.10
R10a	10	144	878.76	**872.74**	-0.69	**857.11**	-2.46	383.37	**857.11**	-2.46
Avg.			537.31	538.84	0.30	533.42	-0.51	112.14	531.07	-0.90
R1b	3	24	*164.46*	**164.46**	0.00	**164.46**	0.00	10.21	**164.46**	0.00
R2b	5	48	296.06	297.55	0.50	**295.76**	-0.10	35.07	**295.66**	-0.14
R3b	7	72	493.30	**489.75**	-0.72	**486.57**	-1.36	63.70	**486.57**	-1.36
R4b	9	96	535.90	538.23	0.44	**534.78**	-0.21	94.25	**530.70**	-0.97
R5b	11	120	589.74	591.44	0.29	**583.83**	-1.00	275.24	**579.76**	-1.69
R6b	13	144	*743.60*	754.87	1.52	747.05	0.46	394.20	743.69	0.01
R7b	4	36	*248.21*	248.75	0.22	**248.21**	0.00	14.23	**248.21**	0.00
R8b	6	72	462.69	467.87	1.12	463.39	0.15	69.36	**461.39**	-0.28
R9b	8	108	601.96	607.57	0.93	**601.72**	-0.04	208.18	**597.75**	-0.70
R10b	10	144	798.63	820.26	2.71	804.70	0.76	531.80	**795.16**	-0.43
Avg.			493.46	498.07	0.70	493.05	-0.13	169.62	490.33	-0.56
Total avg.			515.38	518.46	0.50	513.24	-0.32	140.88	510.70	-0.73

[a] best known solutions computed by Cordeau and Laporte (2003b)
[b] average run times in minutes on a Pentium D computer with 3.2 GHz

to $C = 6$, and maximum user ride time to $\bar{L} = 90$. Furthermore, for instances R1a–R6a and R1b–R6b the number of vehicles was set such that routes are only moderately full while instances R7a–R10a and R7b–R10b might be infeasible with fewer vehicles.

3.5.2 Comparison to tabu search on standard DARP

To our best knowledge no other solution method, besides the TS, has been used to solve this set of instances, using the minimum distance objective function. Table 3.1 gives the results obtained with *heurVNS* compared to the best known solutions by the TS. Column one contains the name of the instance; columns two and three the size in terms of the number of available vehicles (m) and the number of requests (n). Column "TS" gives the best solutions of the TS described in Cordeau and Laporte (2003b). These solutions are the best solutions across all experiments conducted by the previous mentioned authors; either the best out of 10 runs of the TS with 10^4 or one run with 10^5 iterations. Several runs of their TS yield different solutions since the initial solution is constructed randomly; and also the tabu tenure as well as the amount of adjustment, regarding the penalty terms, includes some

randomness. The subsequent columns give the results obtained by means of *heurVNS*. First, average values over 5 independent runs and the deviations from the best known solutions are given. Negative percentage deviations indicate an improved solution with respect to the best known value. Those solution values are given in bold. The subsequent columns give the best solution values out of these 5 random runs and the respective deviations from the best known values. Column "CPU" gives the average run time in minutes per instance. The last two columns provide the best solution values obtained during all experiments in the parameter tuning phase of *heurVNS*. Cordeau and Laporte (2003b) obtain their best results with an average run time of approximately 338.82 minutes on a Pentium 4, 2 GHz computer, for either 10 random runs with 10^4 iterations or one run with 10^5 iterations. *heurVNS* needs on average 140.88 minutes. Taking average values over 5 runs, it is on average 0.5% worse than the TS. Taking the best values out of these 5 runs, it improves the solutions of the TS by, on average, 0.32%. Taking the best values across all versions tested in the parameter tuning phase, *heurVNS* finds 15 new best solutions and 4 ties. Those solution values that could not be improved are marked in italic letters in column "TS". Summarizing the obtained results, *heurVNS* is able to generate high quality solutions within reasonable computation times.

Further experiments, following the intuition that a varying correction term χ in the move neighborhood may perform better than a fixed one, are summarized in Appendix B. Comparing average results for fixed and varying correction terms for the first 10 instances (narrow time windows) and for the second 10 instances (wider time windows), in all cases better results for the first 10 instances can be achieved. This indicates that *heurVNS* performs slightly better on instances with moderately narrow time windows. Choosing χ randomly in $[4\frac{1.5n}{m}, 4\frac{2.5n}{m})$ leads to slightly better results on average for the first half of the data set (0.25% worse than TS compared to 0.3% in case of a fixed χ) but to slightly worse results for the second half of the data set (0.78% compared to 0.7%), see Table B.1. The opposite is true if χ is randomly set to the number of vertices on a currently existing route excluding the depots and multiplied by 4. In this case worse results for the first part of the data set are obtained (0.47% worse than TS compared to 0.3% in case of a fixed χ) but better results for the second part of the data set (0.65% compared to 0.7%), see Table B.1. A mix of these two setting would thus yield a slightly lower deviation from the tabu search (0.45% on average) than with a fixed χ. However, so far an appropriate setting suitable to achieve better results for both parts of the data set could not be identified. Therefore, in all subsequent experiments a fixed correction term will be used.

3.5.3 Comparison to genetic algorithm with modified objective function

In order to test the flexibility of *heurVNS*, we adapted the objective function to the one used by Jørgensen et al. (2007). As stated above, they minimize a weighted combination

of total routing costs, total excess ride time with respect to direct ride time, total waiting time with passengers aboard the vehicle, and route duration. Furthermore, the solution framework of Jørgensen et al. (2007) allows time window, route duration, and ride time violations, but penalizes them. These constraints are therefore treated as soft constraints. Only vehicle capacity is treated as a hard constraint. The objective function thus applied is the following,

$$\hat{f}'(\int) = w_1\hat{c}(\int) + w_2\hat{r}(\int) + w_3\hat{l}(\int) + w_4\hat{g}(\int) + w_5\hat{d}(\int) + w_6[\hat{w}(\int) + \hat{e}(\int)] + w_7\hat{t}(\int). \quad (3.18)$$

The term $\hat{r}(\int)$ denotes excess ride times with respect to direct ride times and is computed as $\hat{r}(\int) = \sum_{i=1}^{n} (B_{n+i} - D_i - t_{i,n+i})$; $\hat{l}(\int)$ denotes the sum over all waiting times weighted by the number of passengers aboard the vehicle when waiting, $\hat{l}(\int) = \sum_{i=1}^{2n} [W_i(Q_i - q_i)]$; $\hat{g}(\int)$ gives the sum over all individual route durations, $\hat{g}(\int) = \sum_{k=1}^{m} (B^k_{2n+1} - B^k_0)$; $\hat{e}(\int)$, finally, denotes the sum over early arrivals, $\hat{e}(\int) = \sum_{i=1}^{2n} (e_i - A_i)^+$. Early arrivals are penalized in the same way as late arrivals $\hat{w}(\int)$. Jørgensen et al. (2007) set the weights to $w_1 = 8$, $w_2 = 3$, $w_3 = 1$ and $w_4 = 1$, and $w_5 = w_6 = w_7 = n$.

In order to accommodate the above objectives, we adapted our evaluation function in the following way:

$$\hat{f}''(\int) = w_1\hat{c}(\int) + w_2\hat{r}(\int) + w_3\hat{l}(\int) + w_4\hat{g}(\int) + \hat{\alpha}\hat{q}(\int) + \hat{\beta}\hat{d}(\int) + \hat{\gamma}\hat{w}(\int) + \hat{\tau}\hat{t}(\int). \quad (3.19)$$

We thus minimize routing costs $\hat{c}(\int)$, excess ride time $\hat{r}(\int)$, waiting with passengers aboard $\hat{l}(\int)$, and route duration $\hat{g}(\int)$, using the same weights as Jørgensen et al. (2007); and we penalize load violation $\hat{q}(\int)$, duration violation $\hat{d}(\int)$, time window violation $\hat{w}(\int)$, and ride time violation $\hat{t}(\int)$. Note that, as before, in our framework a solution $\int$ can only become a new global best solution $\int_{best}$, if and only if $\hat{q}(\int) = \hat{d}(\int) = \hat{w}(\int) = \hat{t}(\int) = 0$. Our framework thus still treats vehicle capacity, maximum route duration, maximum ride time, and time windows as hard constraints. Early arrivals are neither penalized nor prohibited in our case. We still assume that a vehicle can arrive early at a pickup or delivery location but it will have to wait until service is possible. The waiting time for passengers, which incurs due to this assumption, will be minimized by the term $\hat{l}(\int)$ in the objective function.

Jørgensen et al. (2007) run their algorithm 5 times and provide average values over these 5 runs for total route duration $\hat{g}(\int)$, total and average vehicle waiting time, and total and average ride time. The according objective function values $\hat{f}'(\int)$ as well as the values for the total distance traveled $\hat{c}(\int)$ and passenger waiting time $\hat{l}(\int)$, see Table 3.2, are taken from Bergvinsdottir (2004). Total vehicle waiting time is computed as $\sum_{i=1}^{2n} W_i$ and average ride time as $\sum_{i=1}^{n} L_i/n$. Furthermore, Jørgensen et al. (2007) only provide solution values for 13 instances: R1a–R3a, R5a, R9a and R10b and R1b, R2b, R5b–R7b, R9b and R10b. Therefore, we also restrict our computations to these instances and we also provide average values over 5 runs. In order to yield comparable computation times, the maximum iteration

Table 3.2: GA by Jørgensen et al. (2007) (average values over 5 runs)

	total cost $f'(\int)$	travel distance $c(\int)$	total duration $g(\int)$	passenger waiting $l(\int)$	average ride time	vehicle waiting time	CPU[a]
R1a	4696	309	1041	29	19.86	252	5.57
R2a	19426	539	1969	81	28.47	470	11.43
R3a	65306	1047	2779	144	42.79	292	21.58
R5a	213420	1350	4250	286	42.49	500	58.23
R9a	333283	1343	3597	132	57.88	94	40.78
R10a	740890	1811	5006	401	58.42	315	65.98
R1b	4762	284	907	5	26.24	143	5.46
R2b	13580	561	1719	53	25.3	198	11.72
R5b	98111	1344	4296	221	38.46	552	58.93
R6b	185169	1799	5309	361	42.59	630	81.23
R7b	9169	478	1299	27	27.5	102	8.29
R9b	167709	1372	3679	166	49.65	147	44.66
R10b	474758	1740	4733	202	55.34	113	66.41
Avg.	179252.23	1075.15	3121.85	162.15	39.61	292.92	36.94

[a] run times in minutes on an Intel Celeron 2 GHz processor

limit of *heurVNS* has been reduced to $5x10^5$ iterations.

Table 3.3 gives the results generated by *heurVNS*. We provide values for the objective function $\hat{f}'(\int)$, total travel distance $\hat{c}(\int)$, total route duration $\hat{g}(\int)$, total passenger waiting time $\hat{l}(\int)$, total excess ride time with respect to direct ride time $\hat{r}(\int)$, the sum over early arrivals $\hat{e}(\int)$ (all other violations are 0 in our case), average ride time, and total vehicle waiting time. The percentage deviations for the objective function values $\hat{f}'(\int)$, provided by Bergvinsdottir (2004); Jørgensen et al. (2007), are given in the columns headed with "(%)". A negative percentage deviation indicates that the corresponding average value obtained by means of the VNS is better than the according value computed by the GA. *heurVNS* yields better results for all instances. Computation times are comparable: 37.59 minutes on average in case of *heurVNS* and 36.94 minutes in case of the GA.

In Table 3.3, we provide average values for total routing costs or the total distance traveled $\hat{c}(\int)$. When compared to the values given in Table 3.1, where only $\hat{c}(\int)$ is minimized, it becomes obvious that a combined objective function, including service quality oriented criteria, such as passenger waiting time or user ride time, entails higher routing costs. This trade-off between the two objectives is subject to investigation in the following chapter.

3.6 Summary

In this chapter a VNS for the static multi-vehicle DARP has been proposed. The basic algorithm is generic enough to be applicable for two different variants of the problem. The only change necessary is the evaluation function used in the local search and in the move

Table 3.3: *heurVNS* (average values over 5 runs) compared to GA

	total cost $\hat{f}'(\int)$	(%)	travel dist. $\hat{c}(\int)$	total dur. $\hat{g}(\int)$	pass. wait. $\hat{l}(\int)$	excess ride $\hat{r}(\int)$	early arrival $\hat{e}(\int)$	avg. ride time	veh. wait. time	CPU[a]
R1a	**3193.41**	-32.00	273.70	863.65	0.00	35.50	1.40	7.22	109.95	5.69
R2a	**15004.14**	-22.76	424.63	1858.89	0.16	99.64	196.86	7.80	474.26	10.00
R3a	**15755.32**	-75.87	780.32	2433.10	1.92	332.86	84.43	12.59	212.78	11.81
R5a	**21253.28**	-90.04	1017.88	3729.13	0.87	272.51	71.36	8.19	311.25	25.75
R9a	**14202.42**	-95.74	1041.34	3236.31	1.45	496.70	10.59	12.09	34.97	64.67
R10a	**24624.66**	-96.68	1382.01	4495.21	1.95	531.20	51.93	11.14	233.20	78.75
R1b	**2841.52**	-40.33	239.12	731.80	0.00	65.60	0.00	9.51	12.68	6.76
R2b	**5009.27**	-63.11	431.89	1402.00	0.00	26.04	1.54	6.04	10.12	16.26
R5b	**12569.40**	-87.19	953.73	3430.09	0.00	201.48	7.54	7.64	76.36	47.29
R6b	**15250.82**	-91.76	1222.64	4163.00	0.00	303.64	2.75	8.75	60.36	50.70
R7b	**4466.11**	-51.29	358.88	1133.22	0.00	55.74	8.18	9.45	54.34	9.98
R9b	**13455.38**	-91.98	964.44	3229.63	0.47	359.16	13.26	10.41	105.19	54.44
R10b	**16250.22**	-96.58	1306.72	4203.66	1.23	480.75	1.04	10.43	16.94	106.51
Avg.	12605.84	-71.95	799.79	2685.36	0.62	250.83	34.68	9.33	131.72	37.59

[a] run times in minutes on a Pentium D computer with 3.2 GHz

or not step.

In the first version (standard DARP) user inconvenience is only represented by a maximum ride time limit. In the objective function total routing costs, which correspond to the total distance traveled, are minimized. In this problem class the results by Cordeau and Laporte (2003b) are already very good. We were not able to improve them when considering average values over five random runs. However, taking the best out of these five runs, an average improvement of 0.32% was possible. Considering the best solutions across all parameter tuning experiments, an average improvement of 0.73% was achieved.

In the second version most constraints are relaxed to soft constraints and all aspects of user inconvenience are penalized. This version of the DARP has recently been proposed by Jørgensen et al. (2007), who also provide results using a genetic algorithm. In this problem class we achieved remarkable improvements of, on average, 71.95%.

The next steps will be the integration of a user related objective function (Chapter 4) and then additional real world constraints (Chapters 5 and 6).

4 Visualizing the trade-off between costs and user inconvenience

4.1 Introduction

The research work summarized in this chapter is based on the fact that in ambulance routing problems at least two conflicting objectives exist. On the one hand, the company's objective is to minimize costs. On the other hand, users seek the highest possible quality of service level. Low user inconvenience entails high quality of service and vice versa. Usually there exists a certain trade-off between cost minimization and the amount of inconvenience caused for the persons transported. On the company side it is thus important to know the costs of an increased level of service. In order to visualize this trade-off relationship, we define the multi-objective DARP (MO-DARP). Cost on the company side (e.g. the ARC) will be measured in terms of the total distance traveled by the vehicles and user inconvenience in terms of mean user ride time. The lower the mean user ride time the higher the overall quality of service for the persons transported. In order to avoid scenarios that result in mostly very short user ride times but one or two excessively long ones, maximum user ride time limits are also imposed. Our task is now to develop a decision support system providing the ambulance dispatcher with the different efficient transportation plans; including information regarding their respective quality of service level as well as their costs. As mentioned above, the task of choosing a transportation plan from the set of efficient transportation plans is to be left with the person in charge. The findings of this chapter have also been summarized in a scientific article (Parragh et al., 2009b). Many passages are taken from this article.

4.1.1 Solution attributes in multi-objective optimization

The term *efficient* refers to the set of Pareto optimal solutions in a multi-objective context. A transportation plan is Pareto optimal if there does not exist any other transportation plan that weakly dominates it. A transportation plan *weakly dominates* another transportation plan if it is better in at least one objective and not worse in any other objective. The image of a Pareto optimal transportation plan in objective space is called non-dominated point. Thus, the set of efficient solutions is the set of all Pareto optimal solutions. Its image in objective space is also referred to as non-dominated frontier or efficient frontier.

All solutions lying on the same Pareto frontier are incomparable across each other. This means that either solution is better in at least one objective and worse in at least one other objective than all other solutions of this frontier.

Within an efficient set one can further distinguish between *supported* and *non-supported* solutions. Supported efficient solutions are all those solutions that are optimal solutions of an aggregation of the multi-objective problem into a weighted sum single objective problem. For a two-objective problem the weighted sum objective function is given by,

$$\min \omega z_1(x) + (1 - \omega) z_2(x), \tag{4.1}$$

with $0 \leq \omega \leq 1$. Supported efficient solutions are located on the convex hull of the feasible set in objective space. By varying ω the whole set of supported solutions can be found. Non-supported efficient solutions, usually forming a major part of the Pareto frontier, are efficient solutions that are not optimal solutions of (4.1) for any weight combination $0 \leq \omega \leq 1$. They lie in the interior of the convex hull of the feasible set in objective space. Until today no theoretical characterization is known about non-supported efficient solutions. Consequently, no algorithm exists for generating in a straightforward way this set of solutions. In general non-supported efficient solutions are computed within an enumeration scheme based on a ranking algorithm or branch and bound algorithms.

4.1.2 Related work

The incorporation of multiple objectives is becoming more and more important in the field of vehicle routing, measured by the growing body of research emerging from that domain, (see, e.g., Jozefowiez et al., 2007a,b; Lacomme et al., 2006; Pacheco and Martí, 2006; Pasia et al., 2007a,b). Very recently a survey on multi-objective routing problems has been published by Jozefowiez et al. (2008). According to them the purpose of extending single objective problems to the multi objective case is the increased practical applicability; most logistical problems are not only cost driven. In our opinion, this is especially true when dealing with passenger or patient transportation.

Although there exists an abundant body of literature on the DARP, a major part focuses exclusively on the minimization of routing costs based on the distance traveled by the different vehicles used (Cordeau and Laporte, 2003b; Cullen et al., 1981; Dumas et al., 1989; Psaraftis, 1983a). In case the objective of minimum user inconvenience is also treated, this is either done by applying it instead of the minimum cost objective, or by means of a weighted sum objective function. Sexton and Bodin (1985a,b), e.g., entirely focus on user inconvenience. They minimize the difference between actual and desired delivery times and between actual and shortest possible ride times. Madsen et al. (1995) were the first to explicitly solve the DARP with multiple objectives. They resort to the weighted sum approach to incorporate cost as well as user related objectives into their optimization

procedure. The weighted sum of the following objectives is minimized: total driving time, number of vehicles, costs, total waiting time and user inconvenience in terms of the deviation from promised service times. Toth and Vigo (1997b) minimize costs and penalties for user inconvenience. Diana and Dessouky (2004) consider a weighted combination of the total distance traveled, excess user ride time (with respect to direct ride time) and vehicle idle times. As shown in the previous chapter, also Jørgensen et al. (2007) apply an aggregated objective function.

The combination of several objectives into one by means of a weighted sum objective function is at the expense of revealing the trade-off between the concurrent goals considered. Moreover, the decision maker's preference (represented as a weight vector) has to be determined before executing the optimization procedure. Vincke (1992) and Stummer (1998) refer to this concept of a priori choice as American School as opposed to the French School of a posteriori choice across trade-off solutions. Early solution concepts belonging to the French School in multi-objective optimization predominantly belong to the field of genetic algorithms. Following the pioneering work of Schaffer (1988), more recent successful and widely applied solution paradigms involve the NSGA-II by Deb et al. (2002) and the SPEA by Zitzler et al. (2001); Zitzler and Thiele (1999), two genetic algorithms using the Pareto dominance criterion. Furthermore, there is the P-ACO by Doerner et al. (2004) an ant colony optimization based procedure, the Multi-Objective Simulated Annealing (MOSA) by Ulungu (1993); Ulungu and Teghem (1992), the Pareto Simulated Annealing (P-SA) by Czyzak and Jaszkiewicz (1996, 1998), and the Multi Objective Tabu Search (MOTS) by Gandibleux et al. (1997), to name a few. For an overview of the different solution methods applied in Pareto multi-objective optimization we refer to Ehrgott and Gandibleux (2004).

About a decade ago Mladenovic and Hansen (1997) introduced the VNS concept and Glover and Laguna (1997) sketched the PR idea. VNS yielded promising results for the single objective problem in the previous chapter and PR has shown to be effective as intensification mechanism in the context of single-objective combinatorial optimization (Aiex et al., 2003, 2005; Ghamlouche et al., 2004; Resende and Ribeiro, 2003). Recent applications of PR in the context of vehicle routing and location routing respectively involve the work of Ho and Gendreau (2006) and Prins et al. (2006). In the first implementation it is combined with a tabu search heuristic, in the latter with a GRASP. Very recently PR has also been successfully incorporated into solution concepts in the field of multi-objective optimization (see Gandibleux et al., 2004; Pasia et al., 2007c, 2006; Przybylski et al., 2008).

The multi-objective two-phase framework we use to combine VNS and PR, distinguishing between supported and non-supported solutions, is inspired by the work of Przybylski et al. (2008). They solve the bi-objective assignment problem using a two-phase concept, which improves the original version by Ulungu and Teghem (1995).

4.1.3 Contribution

We propose a two-phase metaheuristic based method that provides the ambulance dispatcher with all possible trade-off solutions that can be found. Phase one aims at generating the supported part of the Pareto frontier, phase two at the computation of the non-supported part. Phase one consists of a weighted sum based solution method. In this context the VNS idea is used. A heuristic weighted sum algorithm might generate sub-optimal solutions not part of the set of efficient supported solutions. Some of them may, nevertheless, be efficient. These solutions belong to the set of non-supported efficient solutions. Phase two of our solution approach is a heuristic module using ideas from PR. It aims at generating all those solutions not found by the weighted sum based VNS.

The contribution of this chapter is fourfold. First, we define the MO-DARP as a true multi-objective problem. Second, we develop an efficient heuristic solution procedure, combining two state-of-the-art concepts. Third, we embed an existing branch and cut algorithm into the ϵ-constraint framework to generate the exact efficient sets for small to medium-sized instances for comparison purposes. And last but not least, we provide an in-depth analysis of different major design elements in multi-objective PR.

The remainder of this chapter is organized as follows. First, a formal definition of the MO-DARP is given, followed by a detailed description of the solution methods used. Finally computational results for two sets of benchmark instances are discussed.

4.2 Problem definition

In contrast to the standard DARP which has been formulated as a 3-index program in the previous chapter, the MO-DARP will be formulated as a 2-index program. The necessary notation is given in the following. The MO-DARP is also modeled on a complete directed graph $G = (V, A)$ and for each arc (i, j) a non-negative travel cost c_{ij} is considered. A total of n customer requests are to be served by m vehicles with a capacity of C. The set of all pickup vertices is denoted by P, the set of all delivery vertices by D. Every vehicle starts at the origin depot 0 and ends its route at the destination depot $2n+1$ ($V = P \cup D \cup \{0, 2n + 1\}$). Note that origin and destination depot can nevertheless be at the same location. At every pickup vertex a certain number of passengers ($q_i > 0$) waits to be transported. The demand at every delivery vertex is equal to $q_{n+i} = -q_i$. Again, either the pickup vertex (origin) or the delivery vertex (destination) is associated with a time window $[e_i, l_i]$, depending on the type of request. At each vertex loading or unloading operations last for a given service time d_i. In order to limit user inconvenience caused by long detours, a maximum user ride time $\bar{L}$ has to be respected. By means of $\bar{L}$ time windows are set for all origins/destinations, that were not associated with a time window initially. They are constructed relative to the given time window of the respective request. Beginning of service B_i at each vertex i are to be

determined. The ride time of a client is calculated by $L_i = B_{n+i} - (B_i + d_i)$. The variable Q_i gives the load when leaving vertex i, and the binary routing variables x_{ij} evaluate to one if arc (i, j) is traversed by a vehicle. The mathematical problem formulation proposed in Ropke et al. (2007) adapted to the two-objective case is:

$$\min z_1(x) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{4.2}$$

$$\min z_2(x) = \frac{1}{n} \sum_{i \in P} L_i \tag{4.3}$$

subject to:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in P \cup D \tag{4.4}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in P \cup D \tag{4.5}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{4.6}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{4.7}$$

$$x_{ij} = 1 \Rightarrow \quad B_j \geq B_i + d_i + t_{ij} \qquad \forall i \in V, j \in V \tag{4.8}$$

$$x_{ij} = 1 \Rightarrow \quad Q_j \geq Q_i + q_j \qquad \forall i \in V, j \in V \tag{4.9}$$

$$L_i = B_{n+i} - (B_i + d_i) \qquad \forall i \in P \tag{4.10}$$

$$e_i \leq B_i \leq l_i \qquad \forall i \in V \tag{4.11}$$

$$t_{i,n+i} \leq L_i \leq \bar{L} \qquad \forall i \in P \tag{4.12}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{C, C + q_i\} \qquad \forall i \in V \tag{4.13}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{4.14}$$

The objective functions given in (4.2) and (4.3) minimize routing costs and mean user ride time, respectively. Constraints (4.4) and (4.5) ensure that every vertex is visited exactly once. Inequalities (4.6) limit the number of vehicles leaving the depot. Constraints (4.7) take care of precedence relations between origins and destinations (Ropke et al., 2007; Ruland and Rodin, 1997). The set $\mathcal{S}$ is defined as the set of all vertex subsets $S \subseteq V$, such that $0 \in S$, $2n+1 \notin S$, $n+i \in S$ and $i \notin S$ for some $i \in P$. Consistency with respect to time and load variables is guaranteed by inequalities (4.8) and (4.9). Constraints (4.10) and (4.12) ensure that maximum user ride times are respected. Time window compliance is taken care of by (4.11). Inequalities (4.13) guarantee that the maximum vehicle load is not exceeded. Note that subtour elimination is implicitly taken care of by constraints (4.8) assuming that $(t_{ij} + d_i) > 0$ for all i and j.

Algorithm 4.1 Solution framework

phase 1 (repeat for several weight combinations)
weighted sum based VNS // aims at finding supported solutions
phase 2 (repeat until some stopping criterion is met)
path relinking (PR) // aims at finding non-supported as well as supported solutions not found by the weighted sum method
local search (LS) // improves solutions found by PR

4.3 Solution framework

In order to generate the whole Pareto frontier of the MO-DARP we chose to develop a two-phase solution method. Phase one aims at generating a subset of the supported part of the Pareto frontier by means of a weighted sum based VNS. The VNS is run several times for different weight combinations. Some of the solutions found may indeed be efficient supported solutions, some may be non-supported efficient solutions, and some may be non-efficient. Phase two consists of a PR module departing from the approximation of the efficient set obtained by phase one. To improve on the solutions found along the different paths a local search algorithm, i.e. an iterative improvement procedure, is used. The whole framework is depicted in Algorithm 4.1.

4.3.1 Solution evaluation

In a heuristic context, the second objective gives rise to the issue of correctly evaluating arrival, waiting, and beginning of service times at each vertex. An evaluation procedure, setting the beginning of service at each vertex in the best possible way by using the notion of forward time slacks, was proposed by Savelsbergh (1992). Cordeau and Laporte (2003b) propose an eight step evaluation scheme for the DARP based on the forward time slack. This evaluation scheme has also been used in Chapter 3.

The ride times obtained by this procedure are quite often equal to the exact ride times. However, this is not always the case. Therefore, we use a slightly different evaluation scheme to obtain a better approximation of the minimum ride times for a given route configuration. Let A_i denote the arrival time (initially $B_i := \max\{A_i, e_i\}$), $D_i := B_i + d_i$ the departure time, and $W_i := B_i - A_i$ the waiting time at vertex i. The changes made to the original procedure only affect the computation of the forward time slack F_i at origin vertices. In Cordeau and Laporte (2003b) it is computed as follows,

$$F_i = \min_{i \le j \le q} \left\{ \sum_{i < p \le j} W_p + (\min\left\{l_j - B_j, \bar{L} - P_j\right\})^+ \right\}, \tag{4.15}$$

Our modified version of the forward time slack does not consider the above buffer $\mathcal{B}_j = (\min\left\{l_j - B_j, \bar{L} - P_j\right\})^+$ at destinations whose origins lie before i. This is done to avoid a

ride time increase caused by postponing the arrival at destination vertices. Moreover, at all other vertices the buffer can be set to $\hat{\mathcal{B}}_j = (l_j - B_j)^+$. The second term $(\bar{L} - P_j)$ does not need to be considered, since shifting the beginning of service at i to a later moment in time has no impact on the ride time of users that have not been picked up yet. This results in

$$\hat{F}_i = \min_{i \leq j \leq q} \begin{cases} \left(\sum_{i<p\leq j} W_p\right) & \text{if } j \in \{n+1, \ldots, 2n\} \text{ and } D_{j-n} < D_i \text{ ,} \\ \left(\sum_{i<p\leq j} W_p + \hat{\mathcal{B}}_j\right) & \text{else.} \end{cases} \tag{4.16}$$

Note that this evaluation scheme is more restrictive in terms of feasibility than the evaluation procedure of Cordeau and Laporte (2003b). Consequently, the original version (for further details we refer to the previous chapter, Section 3.4.7) is used in the heuristic weighted sum based method of phase one. This is necessary since it starts with an initial solution that might be infeasible. Thus, attaining feasibility is of predominant importance in this phase. Ride times are only reevaluated at the end of the search. In the PR module we accept this trade-off for the advantage of obtaining better approximations of the true minimum ride times possible for the respective route. Feasibility is not a major issue in this phase since all seeding solutions passed to the PR module have to be feasible.

Deliberately increasing the beginning of service B_i at vertex i until at most the end of the time window l_i might yield waiting times W_i within the user defined time window. This should be questioned. However, an according restriction is not considered in the branch and cut algorithm (Ropke et al., 2007) that is used for comparison purposes. Thus, in order to allow fair comparison across all solution methods applied, it will not be integrated into the heuristic evaluation scheme either. A related issue (penalizing user waiting time while aboard a vehicle) is investigated in the following chapter.

In order to reduce the search space, see also Section 3.4.1, all solution procedures make use of pre-processing steps as described by Cordeau (2006). As in Chapter 3, at intermediate steps of the search constraint violations are allowed but penalized. The following evaluation function is applied,

$$\hat{f}(\int) = \omega \hat{z}_1(\int) + (1-\omega)\hat{z}_2(\int) + \hat{\alpha}\hat{q}(\int) + \hat{\gamma}\hat{w}(\int) + \hat{\tau}\hat{t}(\int). \tag{4.17}$$

The terms $\hat{z}_1(\int)$ and $\hat{z}_2(\int)$ represent the normalized routing costs and the normalized mean user ride time of solution $\int$, respectively ($\hat{z}_1(\int) = \frac{1}{\bar{K}} z_1$, $\hat{z}_2(\int) = \frac{1}{\bar{L}} z_2$) and $\bar{K} = \sum_i [\max_j(c_{ij})]$. Only accessible arcs are considered, i.e. arcs that are not forbidden after having applied the pre-processing steps. The parameter $0 \leq \omega \leq 1$ is the weight associated with objective one. The terms $\hat{\alpha}, \hat{\gamma}$, and $\hat{\tau}$ are penalization parameters for load violation $\hat{q}(\int) = \sum_{i \in P \cup D} (Q_i - C)^+$, time window violation $\hat{w}(\int) = \sum_{i \in V} (B_i - l_i)^+$, and ride time violation $\hat{t}(\int) = \sum_{i \in P} (L_i - \bar{L})^+$, respectively. In contrast to Chapter 3, duration violation is not considered. Considering a maximum route duration limit is not necessary here; the

Algorithm 4.2 Phase one: *heurVNSws*

for $\omega = \{1.0, 0.9, \ldots, 0.0\}$ **do**
  initial solution // determine an initial solution $\int$ and set $\hat{k} := 1$
  repeat
    shaking // determine a solution $\int'$ in $\hat{N}_{\hat{k}}(\int)$
    local search // apply local search to $\int'$ to yield $\int''$
    move or not // if $\int''$ meets the acceptance requirements the incumbent solution $\int$ is replaced by $\int''$ and $\hat{k} := 1$, otherwise $\hat{k} := (\hat{k} \mod \hat{k}_{max}) + 1$; if $\int''$ is feasible and better than $\int_{best}$, set $\int_{best} := \int''$
  until some *stopping criterion* has been met
  if $\int_{best}$ is a new non-dominated solution **then**
    add it to the partial Pareto frontier
  end if
end for

time window at the depot(s) is constructed in such a way that the maximum route duration cannot be exceeded. In phase one, the VNS, the penalty parameters are adjusted dynamically throughout the search. They are updated as in *heurVNS* (see Chapter 3, Section 3.4.7) with the only difference that the term to update them $\hat{d}$ is fixed to $\hat{d} = 0.05$. In phase two, the PR module, the penalty terms are fixed at $\hat{\alpha} = \hat{\gamma} = \hat{\tau} = 10$; the focus is put on feasibility in this phase.

4.3.2 Request insertion

As in Chapter 3, best insertion of a request always refers to inserting origin and destination of the respective request one by one in the best possible way into their new route. The notion of *critical vertices*, as described in Cordeau and Laporte (2003b), is used. The critical vertex is the pickup (delivery) vertex if the request is inbound (outbound), i.e. the vertex with the tight time window. This vertex is inserted first at the best possible position into its new route. Then, the non-critical vertex is inserted in accordance with the critical vertex at the best possible position.

4.3.3 Phase one: variable neighborhood search

Phase one is based on the VNS of Chapter 3 (*heurVNS*). It will be denoted as *heurVNSws*. Several modifications were made to the original version. First, *heurVNSws* is run for 11 different weight combinations ($\omega = \{1.0, 0.9, \ldots, 0.0\}$), see Algorithm 4.2. This entails a change in the construction of the initial solution, depending on the current weight combination. Furthermore, also the shaking phase, the local search frequency, and the stopping criterion have been subject to modifications. All changes made to the original procedure are described in the following.

4.3.3.1 Initial solution

In contrast to *heurVNS*, an initial solution is constructed in two different ways, depending on the current weight combination. For the first weight combination ($\omega = 1$) a randomized route construction procedure is used. For all subsequent weight combinations ($\omega < 1$) either the best solution found during the previous run serves as initial solution, or the randomized route construction procedure is evoked. This choice depends on the results of the four preceding weight combinations. If in one of them a new best solution has been identified, the best solution of the previous run serves as initial solution. Otherwise, the randomized route construction procedure is used, in order to avoid local optimality.

As mentioned above, we use pre-processing procedures strengthening time windows and reducing the number of accessible arcs in the graph. Based on these the route construction procedure has been developed. In a first step, requests are put into a list in random order. The first request on the list is inserted into the first route (origin right after destination). All subsequent requests are inserted as follows. Across all routes already opened (consisting of at least one request) insertion is tried at the beginning and at the end of each route, without using in-accessible arcs. Then, the minimum cost insertion position across all these positions is determined. If no such position exists request-wise insertion (origin right after destination) without using in-accessible arcs is tried at intermediate route positions including empty routes. Routes are checked in random order. The respective request is inserted at the first position encountered during the search where no in-accessible arc has to be used. If no such position exists, it is inserted at the first position using one in-accessible arc.

The constructed routing plan is improved by means of the proposed intra-tour local search procedure. Note that only in-accessible arcs that were eliminated based on time window or ride time violations may be part of an initial solution.

4.3.3.2 Shaking

During the shaking phase, as in *heurVNS*, four neighborhood classes are considered: three classical neighborhoods and one new neighborhood. The first two are based on the move and the swap operator, respectively. The third is based on the ejection chain idea and the fourth is the new zero split neighborhood. In contrast to *heurVNS*, closeness is not considered when selecting an insertion route in the move neighborhood. A move neighborhood of size one (M1) thus corresponds to moving one request from its original route to a randomly chosen route. A move of size two (M2), three (M3), and four (M4) consists in moving at most two, three, and four requests, respectively. The swap neighborhood corresponds to the one employed in *heurVNS*: two random sequences of vertices are exchanged. In the swap neighborhood of size one (S1) a sequence of length one is swapped with a sequence of at most length one. The swap neighborhoods of size two (S2), three (S3), and four (S4) consist of swapping two sequences of at most length two, three, and four, respectively. Also the

chain neighborhood (C) and the zero split neighborhood (Z) correspond to the ones applied in *heurVNS*.

The four shaking operators are applied in the following order: M1 – S1 – C1 – M2– S2 – C2 – M3 – S3 – C3 – M4 – S4 – C4 – Z. In contrast to *heurVNS*, the move neighborhood of size one is considered to be the smallest neighborhood. The next larger neighborhoods are the swap neighborhood of size one and the chain neighborhood of size one in this order. After the chain neighborhood of size one the search continues with the move neighborhood of size two and so on, until the chain neighborhood of size four has been reached. The last neighborhood is the zero split neighborhood. Thus, in *heurVNSws* only a total of $\hat{k}_{max} = 13$ different neighborhoods are iteratively traversed during the search. In *heurVNS* the total number of neighborhoods employed was $\hat{k}_{max} = 19$.

4.3.3.3 Local search frequency

As in *heurVNS*, all shaking steps mainly focus on moving requests between routes (inter-tour). The local search procedure aims at improving each route individually (intra-tour); it reconsiders the position of every request within its route in the same way as described in the previous chapter. In contrast to *heurVNS*, whether a solution $\int'$ undergoes local search based improvement does not depend on its quality. In *heurVNSws* the local search improvement heuristic is applied every l_{LS} iterations to the newly constructed solution $\int'$ yielding $\int''$ (l_{LS} is chosen randomly in $\{5, \ldots, 20\}$ every time the local search procedure is evoked). Furthermore, it is applied to $\int'$ whenever this solution constitutes a new best solution $\int_{best}$.

4.3.3.4 Stopping criterion

In order to generate a bundle of trade-off solutions, *heurVNSws* is run for 11 weight combinations ($\omega = \{1.0, 0.9, \ldots, 0.0\}$). The first weight combination is run for 6x10^5 iterations. As described in Section 4.3.3.1, in most subsequent executions the best solution obtained with the previous weight combination serves as initial solution. Therefore, the maximum number of iterations for all other weight combinations ($\omega < 1$) has been set to 5x10^4 iterations.

4.3.4 Phase two: path relinking

As depicted in Algorithm 4.3, the PR module is initiated with the partial Pareto frontier $\mathcal{P}$ obtained in phase one. At every iteration, first, two solutions, serving as initial solution $\int_I$ and guiding solution $\int_G$, are randomly chosen from $\mathcal{P}$. Second, each route in $\int_I$ is mapped to a route in $\int_G$ (see Section 4.3.4.1). Third, a series of solutions is constructed that transforms $\int_I$ step by step into $\int_G$, making only small changes at every step, thus forming a so-called path. Thereafter, the solutions found along the path undergo local search based improvement, see Section 4.3.4.3. Finally, $\mathcal{P}$ is updated. The procedure terminates as soon

Algorithm 4.3 Phase two: path relinking

$\mathcal{P}$:= get_partial_frontier(); // from *heurVNSws*
repeat
  randomly choose $\int_I$ and $\int_G$ from $\mathcal{P}$;
  map each route in $\int_I$ to a route in $\int_G$;
  S_{path} := path_relinking($\int_I$,$\int_G$);
  local_search(S_{path});
  update $\mathcal{P}$;
until some stopping criterion has been met

as a predefined time limit has been reached. PR incorporates two major design elements. These are the way routes in $\int_I$ and $\int_G$ are mapped and how the path is to be constructed. Both are described in further detail in the following.

4.3.4.1 Mapping

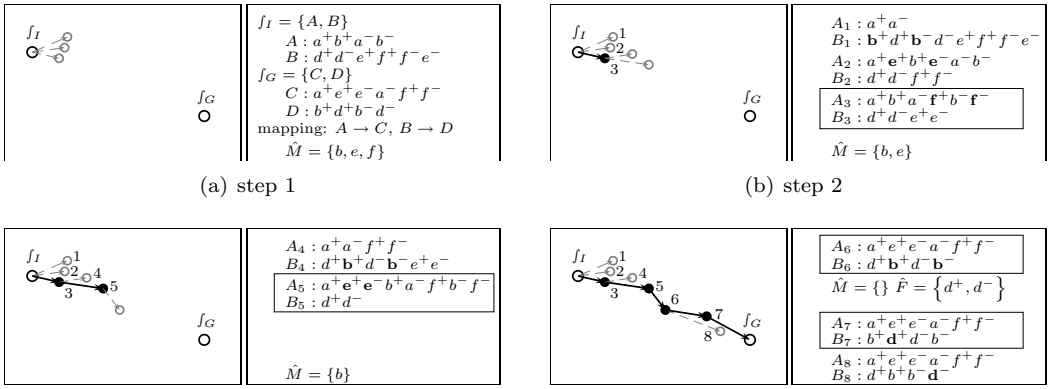
The first ingredient refers to the mapping across routes in initial and guiding solution. Three different strategies are implemented. The first mapping approach is a very simple one: routes in $\int_I$ and $\int_G$ are randomly mapped. However, also two more sophisticated procedures are pursued. They rely on a well-defined similarity measure between two routes: The second mapping approach applies as similarity measure the number of identical requests on two routes. In this case a high value indicates similarity. The similarity measure used in the third mapping strategy is based on the so-called *edit distance*. Its usage in the context of combinatorial permutation type problems, such as the traveling salesman problem or the vehicle routing problem, is discussed in Sörensen (2007). It is the minimum number of elementary operations, i.e. replacements, insertions, and deletions, that are needed to transform one route (part of $\int_I$) into another route (part of $\int_G$). Thus, low transformation costs indicate similarity, while high transformation costs indicate dissimilarity. The edit distance is calculated as depicted in Table 4.1. Six edit operations are needed to transform route A into route B.

Both similarity measures yield an array of similarity values e_{ij} from each route $i \in \int_I$ to

Table 4.1: Edit distance calculation

route A: abcdefg → route B: **bcfamo**			
search vertex	operation	#	current route
b	delete a	1	**b**cdefg
c	no operation	0	**bc**defg
f	delete d and e	2	**bcf**g
a	replace g with a	1	**bcfa**
m	insert m	1	**bcfam**
o	insert o	1	**bcfamo**
	edit distance:	**6**	

= number of elementary operations necessary



(a) step 1 (b) step 2

(c) step 3 (d) the whole path

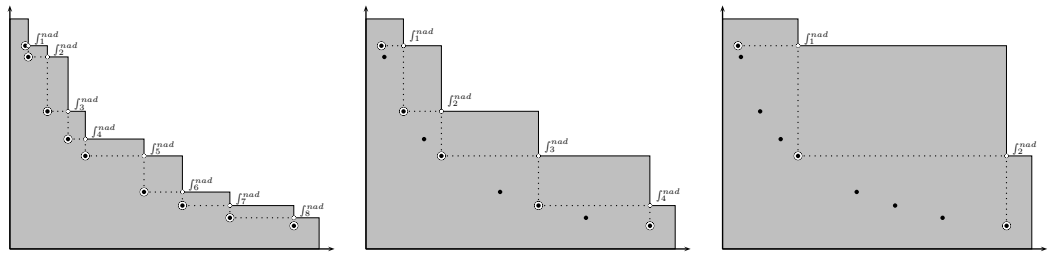
Figure 4.1: Path construction

every route $j \in \int_G$. Mapping across routes is then conducted in a greedy way. In a first step, the two routes that are most similar are mapped. High similarity is indicated by a large value e_{ij} in case of the number of equal requests. In case of the edit distance a small value signifies similarity. This procedure is repeated until there are no more routes to map. In case of a tie, both entries can be chosen with equal probability. Note that both, $\int_I$ and $\int_G$, always consist of exactly m routes (some of them may be empty).

4.3.4.2 Path construction

The second ingredient is the construction of the path itself. After having mapped each route part of $\int_I$ to a route part of $\int_G$, the set $\hat{M}$ is computed. $\hat{M}$ contains all those requests that are not on the correct route according to $\int_G$. This is depicted in Figure 4.1(a). Here $\int_I$ consists of routes A and B and $\int_G$ of routes C and D. Each request i consists of a pickup and delivery vertex pair $\{i^+, i^-\}$. Based on random mapping, route A is mapped to C and route B to D. This means that route A is to be transformed step by step into route C and route B into route D. All requests that need to be moved to yield this result are in $\hat{M}$. The first solution along the path is obtained by considering all possible moves of one request from its route in $\int_I$ to "its" route in $\int_G$. Again, insertion is done in the same way as described in Cordeau and Laporte (2003b), i.e. node-wise best insertion using the notion of critical vertices. The best of these moves constitutes the first solution on the path. To determine the *best* solution across all possible solutions, similar to Pasia et al. (2007c), the weighted sum evaluation function (4.17) is applied. We resort to a different random weight ω at every step along the path. In Figure 4.1(b) solution three is chosen based on this evaluation criterion, in Figure 4.1(c) solution five.

This procedure is repeated until $\hat{M}$ is empty. However, since best insertion is used, the end of the path might not have been reached yet. Consequently, all nodes being on the



(a) $k = 1$ (b) $k = 2$ (c) $k = 4$

Figure 4.2: Nadir points

wrong position with respect to $\int_G$ have to be detected. All these constitute the set $\hat{F}$, see Figure 4.1(d). Now all possible intra-tour moves are considered, i.e. moving each vertex $\in \hat{F}$ from its current position to its final position in $\int_G$. The term possible refers to the condition that the precedence constraint has to be respected at all times. The best move according to a random weight based evaluation function generates the next solution along the path. The procedure ends as soon as $\int_G$ has been reached.

4.3.4.3 Local search in the path relinking module

Basically the same intra-tour local search as in *heurVNSws* is used in the PR module. However, in the context of multi-objective optimization the concept of *improvement* becomes debatable. There are two options. Either, as in the path construction phase, random weight vectors can be defined in order to aggregate the different objective values; or, all objective values can be checked separately and only if the current incumbent solution is weakly dominated by the new solution, a new incumbent has been found. We adopt the latter approach. Furthermore, whenever an incomparable solution is encountered, it is checked if it belongs to the current Pareto frontier $\mathcal{P}$. If it is a new non-dominated solution $\mathcal{P}$ is updated.

Another design issue refers to the notion of *promising* solutions. The path constructed between initial and guiding solution might incorporate solutions that are very far away from the Pareto frontier in terms of objective values. Consequently, the question arises whether all solutions in S_{path} should undergo local search or not. Promising solution are all those solutions that lie within a certain distance from the current Pareto frontier. This *promising* area can be defined by means of Nadir points. Generally, the Nadir point or anti-ideal point is defined by the maximal objective values (in the context of minimization) attained by Pareto optimal solutions of a multi-objective optimization problem (see, e.g., Ehrgott and Tenfelde-Podehl, 2003). This notion can be extended to a set of Nadir points N^k defined by means of certain efficient solutions. In our case every k-th solution of the current Pareto frontier $\mathcal{P}$ ordered with respect to z_1 is used to constitute N^k. Let P^k contain every k-th solution of $\mathcal{P}$,

$N^k = \{\int_j^{nad} = (z_1^{nad}, z_2^{nad}) | z_1^{nad} = \max(z_1^i, z_1^{i+1}), z_2^{nad} = \max(z_2^i, z_2^{i+1}) | \int_i = (z_1^i, z_2^i) \in P^k\}$. This set N^k is used to define the promising area. In Figure 4.2 an example for $k = 1$, 2, and 4 is given. Black solid circles represent the current Pareto frontier $\mathcal{P}$ in objective space. Every k-th solution is part of the set P^k. Nadir points formed by means of P^k are depicted by empty circles. Promising solutions would be those that are located in the gray area. Thus, in general promising solutions can be defined as all those solutions that are non-dominated by a given set of Nadir points N^k.

4.4 An exact method for the MO-DARP

For the purpose of assessing the performance of the heuristic methods, we generate the exact Pareto frontiers of the MO-DARP by means of the ϵ-constraint framework (see, e.g., Laumanns et al., 2006). In the bi-objective ϵ-constraint framework the single objective prob-

Algorithm 4.4 ϵ-constraint framework

```
set upper bound for mean ride time objective z̄_2 := L̄
set ε := z̄_2 + Δ
while feasible region is not empty do
    solve min cost problem with the additional constraint z_2(x) ≤ ε − Δ to obtain z_1(x*)
    compute according minimum mean ride time z_2(x*); set ε := z_2(x*)
    add x* to the set of efficient solutions R
end while
```

lem, e.g. only considering z_1, is solved to optimality, adding a new constraint incorporating a bound on the other objective, e.g. $z_2 \leq \epsilon - \Delta$. With varying ϵ different solutions can be obtained. We implemented the branch and cut algorithm as described in Ropke et al. (2007) to solve the single objective problem with respect to z_1. Consequently, all solutions generated are part of the optimal Pareto frontier. The values for ϵ are chosen as depicted in Algorithm 4.4, in order to obtain the complete efficient set. The parameter Δ was set to $\Delta = 0.001$.

4.5 Computational experiments

All algorithms were implemented in C++. For the implementation of the branch and cut algorithm ILOG Concert Technology 2.5 and CPLEX 11.0 were used. All programs were run on a 3.2 GHz Pentium D computer with a memory of 4 GB. In the following we first describe the test instances used. Thereafter, the definition of the different quality indicators used to assess the solution quality of our algorithms is provided. Then, results of preliminary experiments for the selection of the best mapping strategy and Nadir point setting in the PR module are summarized. Finally, individual results for all instances are discussed.

4.5.1 Test instances

Two data sets (Cordeau, 2006; Ropke et al., 2007, see also Section 2.6 Cor06, Co06+), containing 24 randomly generated test instances each, were used for testing purposes. In both sets the number of requests n ranges from 16 to 96, the number of vehicles m from 2 to 8. The first $\frac{n}{2}$ requests are assumed to be outbound while the remaining $\frac{n}{2}$ are considered as inbound requests. The coordinates of the respective origin and destination vertices were randomly generated in the square $[-10, 10]^2$ according to a uniform distribution. The depot is located at the center of the square. Routing costs c_{ij} as well as travel times t_{ij} correspond to the Euclidean distance between the vertices i and j. In case of an outbound request a tight time window was generated for the destination vertex (l_{n+i} was chosen in the interval $[60, \hat{H}]$, $e_{n+i} = l_{n+i} - 15$, $\hat{H}$ being the end of the planning horizon). Inbound requests have a tight time window on the origin vertex (e_i was chosen in the interval $[0, \hat{H} - 60]$, $l_i = e_i + 15$). No time windows exist for origin (destination) vertices in case of outbound (inbound) requests. For data set A vehicle capacity was set to $C = 3$ and at every origin one passenger ($q_i = 1$, $i \in \{1, \ldots, n\}$, and $q_i = -1$, $i \in \{n + 1, \ldots, 2n\}$) waits to be transported. For data set B vehicle capacity was set to $C = 6$ and every request consists of up to six passengers (q_i is chosen randomly in $\{1, \ldots, 6\}$ and $q_{n+i} = -q_i$, $i \in \{1, \ldots, n\}$). Since the efficient frontier of instance b2-20 consists of only one compromise solution, results for this instance will not be reported.

4.5.2 Quality indicators

To evaluate approximations of the Pareto frontier generated by heuristic solution procedures different performance indicators have been proposed in the literature (see Knowles et al., 2006). Quality indicators of multi-objective metaheuristics following the Pareto approach can be categorized according to the principal aims they pursue, such as a good approximation of the set of Pareto-efficient solutions or the coverage of a broad range of the region of the Pareto frontier. We use three different quality indicators, each providing information with respect to a different aspect of the approximation sets obtained by our algorithms.

The hypervolume indicator I_H (Zitzler and Thiele, 1999) measures the hypervolume of the objective space that is weakly dominated by an approximation set. In Figure 4.3(a) the hypervolume weakly dominated by approximation set $\mathcal{A}$ corresponds to the gray region, the hypervolume weakly dominated by $\mathcal{R}$ to the lightgray field. As reference point serves the objective vector $(\bar{K}, \bar{L})$. All quality indicators are calculated based on normalized objective values. Thus, the reference point becomes $(1, 1)$. Higher I_H values are preferable.

The multiplicative version of the unary epsilon indicator I_ϵ (Zitzler et al., 2003) can be defined as follows. It is the minimum factor ϵ such that if every point in reference set $\mathcal{R}$ was multiplied by ϵ the resulting approximation set would be weakly dominated by

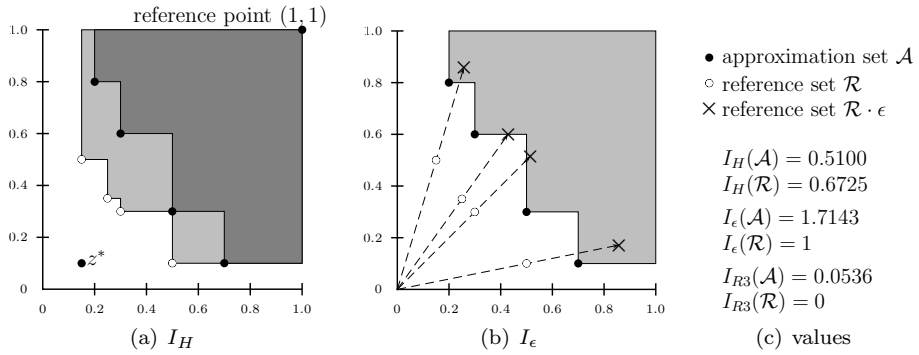


Figure 4.3: Quality indicators

approximation set $\mathcal{A}$,

$$I_\epsilon(\mathcal{A}) = I_\epsilon(\mathcal{A}, \mathcal{R}) = \inf \{\epsilon \in \mathbb{R} | \forall \int_2 \in \mathcal{R} \ \exists \int_1 \in \mathcal{A} \text{ so that } \int_1 \preceq_\epsilon \int_2\}. \tag{4.18}$$

The symbol $\preceq_\epsilon$ is defined as, $\int_1 \preceq_\epsilon \int_2 \Leftrightarrow \forall i | \hat{z}_i(\int_1) \leq \epsilon \cdot \hat{z}_i(\int_2)$. Reference set $\mathcal{R}$ is either equal to the true Pareto frontier or to a good approximation of it. In our case, for the smaller instances $\mathcal{R}$ was generated by means of the ϵ-constraint framework as described above. For the larger instances $\mathcal{R}$ is approximated by considering the union of all points obtained from the different procedures in any of the experiments and removing dominated points. In Figure 4.3(b) every point in reference set $\mathcal{R}$ has to be multiplied by at least $I_\epsilon = 1.7143$ to yield an approximation set that is weakly dominated by $\mathcal{A}$. Low I_ϵ values are preferable.

The R3 indicator I_{R3} Hansen and Jaszkiewicz (1998) incorporates the decision maker's preference into the evaluation procedure. Let Λ denote a set of weight vectors ($|\Lambda| = 10{,}000$, randomly generated) and let $u_\lambda(\int)$ denote the individual utility of solution $\int$ for some λ using the augmented Tchebycheff function:

$$u_\lambda(\int) = 1 - (\max_{1 \leq j \leq 2} \left\{\lambda_j | z_j^* - \hat{z}_j(\int)|\right\} + \rho \sum_{1 \leq j \leq 2} |z_j^* - \hat{z}_j(\int)|), \tag{4.19}$$

where z^* is the ideal point ($z_j^* = \min_{\int \in \mathcal{R}} \{\hat{z}_j(\int)\}$, see Figure 4.3(a)), $\rho = 0.01$ in our case, and j refers to the objectives. Furthermore, let $u^*(\lambda, \mathcal{A}) = \max_{\int \in \mathcal{A}} \{u_\lambda(\int)\}$, i.e. the maximum value obtained by utility function u_λ with weight vector λ across all solutions in approximation set $\mathcal{A}$. We can now compute the R3 Indicator using the following equation,

$$I_{R3}(\mathcal{A}) = I_{R3}(\mathcal{A}, \mathcal{R}) = \frac{\sum_{\lambda \in \Lambda} \left[u^*(\lambda, \mathcal{R}) - u^*(\lambda, \mathcal{A})\right] / u^*(\lambda, \mathcal{R})}{|\Lambda|}. \tag{4.20}$$

Small I_{R3} values are preferable. In Figure 4.3(c), the R3 indicator is $I_{R3}(\mathcal{A}, \mathcal{R}) = 0.0536$.

Table 4.2: Path relinking without local search

Mapping	Data set	I_H	I_ϵ	I_{R3}
Edit distance	A	*0.34454*	1.01832	0.00124
	B	*0.44139*	1.01218	0.00076
	avg.	**0.39297**	1.01525	0.00100
Random	A	0.34452	*1.01640*	0.00124
	B	0.44130	1.01404	0.00094
	avg.	0.39291	1.01522	0.00109
Equal requests	A	*0.34454*	1.01806	*0.00120*
	B	*0.44139*	*1.01206*	*0.00075*
	avg.	**0.39297**	**1.01506**	**0.00097**

4.5.3 Tuning the path relinking module

All PR experiments are based on the partial Pareto frontiers generated in phase one. Each PR parameter setting was given a maximum run time limit of 15 seconds. In PR two design decisions have to be taken. The first refers to the way $\int_I$ and $\int_G$ are chosen from $\mathcal{P}$. Here, initial tests with distance related selection mechanisms showed that, for the problem at hand, random selection clearly dominates all other selection mechanisms tried. Once $\int_I$ and $\int_G$ have been selected, the second design decision determines the mapping between the routes in $\int_I$ and $\int_G$. This design issue is investigated in the following.

In a first step, PR is run without the intra-tour local search step. In Table 4.2 results for the different mapping strategies are given on an aggregated level for each data set (A and B), as well as averages (avg.) over these. The best average values are marked in bold. The best values per data set are given in italic letters. In all tables results given for individual instances obtained from the heuristic algorithms are average values over ten random runs; results reported on group level correspond to average values over all the instances forming the respective group. All three quality measures indicate that mapping based on the number of equal requests is on average the best strategy ($I_H = 0.39297$, $I_\epsilon = 1.01506$, $I_{R3} = 0.00097$), closely followed by edit distance based mapping and random mapping. If the two data sets A and B are considered individually the ranking changes. For data set A random mapping is better than mapping based on the number of equal requests in one indicator (I_ϵ). Regarding the hypervolume indicator mapping based on the number of equal requests and edit distance based mapping are equally good. In terms of the R3 indicator mapping based on the number of equal requests outperforms the other two strategies. For data set B edit distance based mapping and mapping based on equal requests obtain the best values for the hypervolume indicator $I_H = 0.44139$. In terms of the unary epsilon indicator and the R3 indicator, mapping based on the number of equal requests should be chosen. Summarizing these results, it is not clear which mapping strategy performs best. Mapping based on the number of equal requests is a bit better than the other two strategies; indicating that a well defined similarity measure might be beneficial within a restrictive time limit.

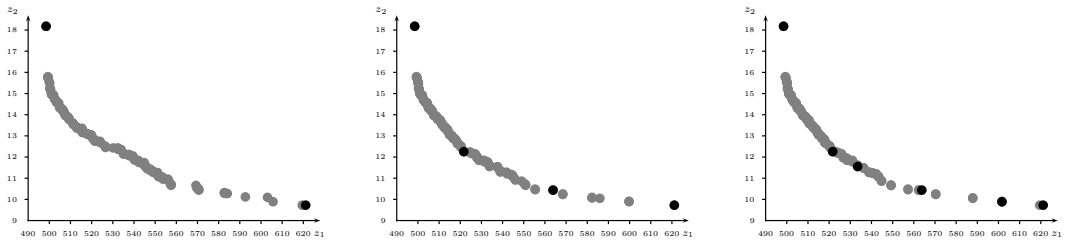
Table 4.3: Path relinking with local search

Nadir points	Mapping	I_H	I_ϵ	I_{R3}
none	Edit distance	0.39292	1.01608	0.00111
	Random	0.39279	1.01774	0.00127
	Equal requests	0.39294	1.01552	0.00107
	Total avg.	0.39288	1.01645	0.00115
$k = 1$	Edit distance	0.39304	1.01396	0.00095
	Random	0.39299	1.01416	0.00102
	Equal requests	**0.39305**	**1.01375**	**0.00092**
	Total avg.	*0.39303*	*1.01396*	*0.00096*
$k = 2$	Edit distance	0.39302	1.01422	0.00097
	Random	0.39299	1.01442	0.00103
	Equal requests	0.39304	1.01378	0.00094
	Total avg.	0.39302	1.01414	0.00098
$k = 4$	Edit distance	0.39300	1.01435	0.00098
	Random	0.39294	1.01498	0.00108
	Equal requests	0.39302	1.01410	0.00096
	Total avg.	0.39299	1.01448	0.00101

In a second step, the complete PR module including the local search step is tested. Table 4.3 gives the according results on a very aggregated level, i.e. average values across all instances, within a time limit of 15 seconds. As mentioned above, different sets of Nadir points were used in order to define which solutions are subject to local search based improvement. Within the time limit of 15 seconds the most restrictive setting ($k = 1$) yields the best results across all quality measures (marked in italic letters in Table 4.3). This confirms our assumption that solutions that are already close to the efficient frontier are more likely to be improved to efficient solutions by local search. If the time limit is increased $k = 1$ and $k = 2$ yield results of similar quality. Here, the intuition is that if the time limit is not very restrictive, some solutions, initially a bit further away from the current frontier $\mathcal{P}$, can still be improved such that they become part of $\mathcal{P}$.

Also the different mapping strategies are further investigated. Within the run time limit of 15 seconds together with local search, mapping based on the number of equal requests yields the best results (with the Nadir points setting $k = 1$) across all parameter combinations. It yields a hypervolume value of $I_H = 0.39305$, a unary epsilon value of $I_\epsilon = 1.01375$, and an R3 value of $I_{R3} = 0.00092$, marked in bold in Table 4.3. Edit distance based mapping and random mapping are on place two and three respectively.

However, the picture changes a bit when the time limit is increased. In case of a time limit of 30 seconds, the three mapping strategies perform almost equally well, each obtaining good results for a different quality indicator. The intuition is that paths obtained by edit distance based mapping and random mapping tend to be longer; also routes that are not very similar in terms of the number of equal requests per route may be mapped. Consequently, the search space is increased. Moreover, using random mapping, if the same pair of solutions is linked more than once, different routes can be mapped. This increased diversification may



(a) $n_{seed} = 2$ (b) $n_{seed} = 4$ (c) $n_{seed} = 6$

Figure 4.4: Instance a5-40: Pareto frontiers based on different number of seeding solutions

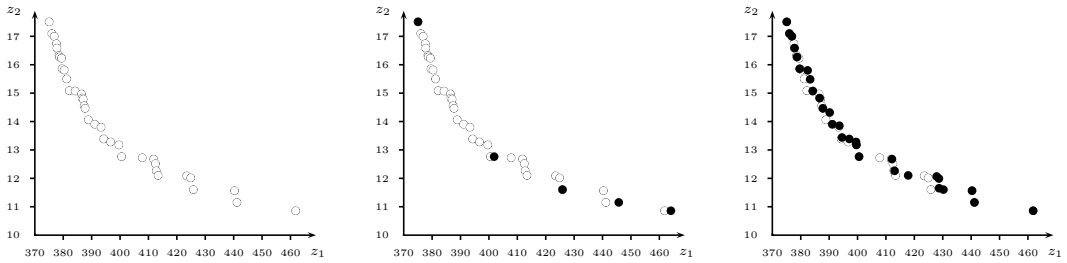
cause random mapping to be beneficial if even more time is given to the procedure.

Summarizing, within the time limit of 15 seconds a mapping strategy that is based on a well defined similarity measure is beneficial with respect to solution quality. In our case the overall best results are obtained using mapping based on the number of equal requests and the Nadir point setting $k = 1$.

4.5.4 Final results

Phase one yields Pareto frontiers consisting of two to ten solutions. This number is usually slightly correlated to the total size of the complete efficient set. Thus, the next step consists in investigating the impact of the number of seeding solutions n_{seed} passed on from phase one to phase two. Three different scenarios are tested. Pareto frontiers based on two, four, and six seeding solutions derived from phase one are computed with the PR module. This is illustrated in Figure 4.4 for instance a5-40. Comparison shows that $n_{seed} = 6$ yields the best approximation. However, also four and only two initial solutions seem to be sufficient to generate a well spread frontier, only slightly inferior to the one obtained with six seeding solutions. Thus, phase two seems to be rather robust with respect to the number of seeding solutions.

This is confirmed by the average results obtained based on different n_{seed} settings for data sets A and B. In all experiments results for individual instances obtained from the heuristic algorithms are given as average values over ten random runs. Those results that are given on group level correspond to average values over all the instances forming the respective group. Table 4.4 gives the average values over the quality indicators per data set and number of seeding solutions; using the Nadir point setting $k = 1$ and mapping based on the number of equal requests. $n_{seed} = 2$ indicates that exactly two seed points are passed on from phase one. In the setting $n_{seed} \leq 4$, at most four seed points are used. If the total number of seeding solutions of phase one is smaller than four, this number is used. The same applies to setting $n_{seed} \leq 6$. From the values obtained it can be derived that the more seed points are available the better the obtained approximation of the efficient set tends to be. Setting $n_{seed} \leq 6$ on average yields the best results in terms of two quality



(a) ϵ-constraint (b) phase one (c) phase two

Figure 4.5: Instance a4-24: Pareto frontiers obtained by the different solution methods

indicators ($I_\epsilon = 1.01708, I_{R3} = 0.00113$). The best values in terms of the hypervolume indicator are obtained with setting $n_{seed} \leq 4$. However, even with only two seed points results of high quality can be generated. Summarizing these results, if time does not permit the computation of a larger number of seed points, two seeding solutions are sufficient to initialize the PR module and obtain a good approximation of the efficient set.

Finally, the gains of phase two with respect to phase one are analyzed. The different phases are illustrated in Figure 4.5 using the example of instance a4-24. The best settings as identified above are used in phase two (mapping based on the number of equal requests and the Nadir point setting $k = 1$). Figure 4.5(a) depicts the exact efficient frontier obtained by means of the ϵ-constraint method for instance a4-24. Figures 4.5(b) and 4.5(c) show the Pareto frontiers generated by means of phase one, the weighted sum based VNS, and phase two, the PR module, respectively. Evidently, a good approximation of the true efficient set can be generated by the proposed two-phase method.

Tables 4.5 and 4.6 give the results for data sets A and B respectively. The first columns are devoted to the description of the reference sets used. Hypervolume values marked with an asterisk indicate that the reference set for this instance corresponds to the exact efficient set. These are obtained by means of the ϵ-constraint method. Data set B is a bit more tightly constrained. Therefore, larger instances can be solved to optimality. The largest

Table 4.4: Path relinking and local search: varying seed points

Seeds	Data set	I_H	I_ϵ	I_{R3}
$n_{seed} = 2$	A	0.34329	1.03360	0.00323
	B	0.44087	1.02513	0.00223
	avg.	0.39208	1.02937	0.00273
$n_{seed} \leq 4$	A	*0.34421*	1.02392	0.00204
	B	*0.44130*	1.01617	0.00125
	avg.	**0.39275**	1.02004	0.00165
$n_{seed} \leq 6$	A	0.34341	*1.02086*	*0.00143*
	B	0.44106	*1.01329*	*0.00083*
	avg.	0.39223	**1.01708**	**0.00113**

Table 4.5: Results data set A

	reference set $\mathcal{R}$		phase one				phase two (15 sec.)		
Instance	I_H	CPU[a]	I_H	I_ϵ	I_{R3}	CPU[a]	I_H	I_ϵ	I_{R3}
a2-16	0.24563*	3.18	0.24236	1.05228	0.00442	2.28	0.24560	1.00894	**0.00000**
a2-20	0.29859*	5.77	0.29612	1.06215	0.00394	5.39	0.29853	1.00903	0.00011
a2-24	0.25644*	24.38	0.25463	1.04840	0.00298	6.19	0.25636	1.00574	0.00017
a3-18	0.29820*	6.30	0.29591	1.04756	0.00417	1.58	**0.29820**	1.00207	**0.00000**
a3-24	0.33340*	20.82	0.33076	1.04708	0.00285	2.98	0.33266	1.02131	0.00102
a3-30	0.26859*	161.63	0.26599	1.03896	0.00256	4.53	0.26765	1.01324	0.00076
a3-36	0.30610*	236.23	0.30000	1.05358	0.00440	6.23	0.30397	1.01490	0.00143
a4-16	0.26532*	4.22	0.26477	1.03517	0.00132	0.93	0.26530	1.00530	**0.00000**
a4-24	0.31843*	70.13	0.31572	1.04604	0.00232	2.01	0.31802	1.00896	0.00035
a4-32	0.32544*	867.08	0.32241	1.04390	0.00326	3.28	0.32473	1.00915	0.00029
avg.	0.29161	139.98	0.28887	1.04751	0.00322	3.54	0.29110	1.00986	0.00041
a4-40	0.37348	-	0.36851	1.08286	0.00626	4.65	0.37115	1.02050	0.00172
a4-48	0.35044	-	0.34640	1.04360	0.00333	8.16	0.34828	1.01372	0.00105
a5-40	0.39900	-	0.39649	1.03686	0.00264	3.88	0.39819	1.01092	0.00061
a5-50	0.35990	-	0.35628	1.04826	0.00377	6.16	0.35790	1.01622	0.00153
a5-60	0.36872	-	0.36432	1.05101	0.00384	8.50	0.36615	1.01659	0.00151
a6-48	0.39911	-	0.39485	1.04426	0.00301	3.94	0.39639	1.01719	0.00087
a6-60	0.36020	-	0.35636	1.05867	0.00432	5.92	0.35828	1.01687	0.00133
a6-72	0.39405	-	0.38887	1.04999	0.00440	9.91	0.39044	1.02821	0.00258
a7-56	0.38829	-	0.38376	1.05438	0.00376	3.74	0.38575	1.01897	0.00123
a7-70	0.38371	-	0.37814	1.05415	0.00438	6.89	0.38010	1.02334	0.00184
a7-84	0.39269	-	0.38753	1.06172	0.00461	10.03	0.38931	1.02681	0.00216
a8-64	0.42151	-	0.41636	1.06810	0.00481	4.27	0.41864	1.02071	0.00171
a8-80	0.42015	-	0.41630	1.05638	0.00389	6.97	0.41765	1.02416	0.00187
a8-96	0.38599	-	0.38086	1.05826	0.00445	10.70	0.38240	1.02864	0.00230
Total avg.	0.34639	-	0.34265	1.05182	0.00374	5.38	0.34465	1.01590	0.00110

[a] run time in minutes
* exact Pareto frontier from ϵ constraint method

instance in terms of requests that can be solved within data set A is instance a3-36, in case of data set B it is instance b4-40. Column *CPU* gives the run times needed to obtain the respective efficient frontiers. For the smallest instances run times are rather low. However, with an increasing number of requests, run times increase rapidly. Moreover, individual run times per Pareto optimal solution within the ϵ-constraint framework vary considerably. Pareto solutions close to the two ends of the efficient frontier can be generated rather quickly while intermediate solutions tend to consume more computation time.

The results from the ϵ-constraint method are compared to those obtained by phase one, i.e. only applying *heurVNSws*, as well as to the whole two-phase procedure. The quality indicators show that good approximations of the Pareto frontiers can be obtained in reasonable time (run times for the whole procedure can be computed by adding 0.25 minutes to those of phase one). The run times needed by phase one are strongly correlated to the requests per vehicle ratio. This is due to the local search step in *heurVNSws*. Its run time depends on the lengths of the respective routes. Consequently, less run time is needed if the requests per vehicle ratio is low. For instance b3-18, b3-24, and b4-16, only solutions part of the exact efficient sets were found in all ten random runs, indicated by $I_\epsilon = 1$ and $I_{R3} = 0$. The values obtained on average for the unary epsilon indicator for both data sets show that the Pareto frontiers generated by the two-phase procedure are very close to the exact

Table 4.6: Results data set B

	reference set $\mathcal{R}$		phase one				phase two (15 sec.)		
Instance	I_H	CPU[a]	I_H	I_ϵ	I_{R3}	CPU[a]	I_H	I_ϵ	I_{R3}
b2-16	0.29978*	2.33	0.29966	1.00714	0.00021	4.00	0.29971	1.00654	0.00014
b2-24	0.34765*	4.45	0.34718	1.02058	0.00087	9.44	0.34756	1.00779	0.00021
b3-18	0.38801*	2.03	0.38774	1.01712	0.00068	2.80	**0.38801**	**1.00000**	**0.00000**
b3-24	0.42200*	2.17	0.42193	1.00603	**0.00000**	4.40	**0.42200**	**1.00000**	**0.00000**
b3-30	0.38286*	16.25	0.38134	1.04024	0.00282	6.10	0.38250	1.00623	0.00041
b3-36	0.40767*	21.38	0.40471	1.03351	0.00236	9.57	0.40688	1.00267	0.00007
b4-16	0.35958*	0.10	**0.35958**	**1.00000**	**0.00000**	1.60	**0.35958**	**1.00000**	**0.00000**
b4-24	0.43711*	16.45	0.43548	1.02719	0.00098	3.29	0.43650	1.01069	0.00041
b4-32	0.46682*	38.88	0.46459	1.04024	0.00192	4.77	0.46559	1.00997	0.00049
b4-40	0.42236*	220.05	0.42125	1.02084	0.00145	8.07	0.42171	1.00834	0.00071
avg.	0.39338	32.41	0.39235	1.02129	0.00113	5.40	0.39301	1.00522	0.00024
b4-48	0.48289	-	0.48028	1.03403	0.00201	14.30	0.48098	1.01507	0.00054
b5-40	0.45364	-	0.45111	1.02971	0.00184	4.81	0.45209	1.00951	0.00057
b5-50	0.45604	-	0.45307	1.03709	0.00233	8.67	0.45376	1.01342	0.00083
b5-60	0.47437	-	0.47126	1.03996	0.00246	12.09	0.47205	1.01483	0.00100
b6-48	0.45949	-	0.45757	1.03006	0.00210	5.00	0.45859	1.01260	0.00086
b6-60	0.47256	-	0.46933	1.03254	0.00221	8.54	0.46987	1.01669	0.00120
b6-72	0.48997	-	0.48695	1.03587	0.00249	12.72	0.48752	1.01555	0.00120
b7-56	0.45466	-	0.45165	1.03958	0.00264	5.76	0.45250	1.01295	0.00084
b7-70	0.52285	-	0.51931	1.03786	0.00253	9.91	0.51996	1.01607	0.00116
b7-84	0.46734	-	0.46398	1.04101	0.00301	13.12	0.46459	1.02006	0.00167
b8-64	0.50267	-	0.49962	1.05694	0.00378	6.09	0.50050	1.01926	0.00134
b8-80	0.50287	-	0.50009	1.04535	0.00343	9.47	0.50078	1.02257	0.00180
b8-96	0.51349	-	0.50945	1.05125	0.00325	14.54	0.51029	1.02629	0.00173
Total avg.	0.44290	-	0.44074	1.03148	0.00197	7.78	0.44146	1.01161	0.00075

[a] run time in minutes
* exact Pareto frontier from ϵ constraint method

frontiers, where known (data set A: $I_\epsilon(avg.) = 1.00986$, data set B: $I_\epsilon(avg.) = 1.00522$).

Whenever the efficient set is unknown the quality measures also indicate that the Pareto frontiers obtained are close to the best approximation constructed across all computational experiments. It can be assumed that these are very close to the true efficient sets.

In case of data set A, compare Table 4.5, phase two improves the total average results of phase one by 0.6% with respect to the hypervolume indicator. The distance to the best value of the unary epsilon indicator is decreased by more than 60%. The R3 indicator value can be reduced by the factor 3.4. The percentage gain of the hypervolume indicator refers to the additional space on average that is weakly dominated by the reference sets. The improvement in terms of the unary epsilon indicator can be interpreted as increased closeness to the reference set. A decrease in R3 indicates that the difference between the utility obtained from the reference set and the utility obtained from the approximations can on average be reduced. For data set B, very similar results are obtained, see Table 4.6.

4.6 Summary

In this chapter we have developed a solution procedure to solve the dial-a-ride problem with two objectives. An existing branch and cut algorithm is used in the ϵ-constraint

framework to generate exact Pareto frontiers for small to medium-sized instances of two sets of benchmark instances. To deal with larger instances a heuristic two-phase procedure has been designed. Phase one consists of a weighted sum based VNS heuristic. In phase two a PR module is seeded with the partial Pareto frontier computed in phase one. Various design aspects of this rather new method in the field of multi-objective optimization have been discussed. The results obtained indicate that a wise mapping mechanism is beneficial with respect to solution quality. Furthermore, within a low time limit the notion of Nadir points is used to define promising solutions that should be subject to local search based improvement. Also the impact of the number of seeding solutions, derived from phase one passed on to phase two, has been investigated. Even with only two seeding solutions a well spread Pareto frontier can be generated. Finally, comparison of the results computed with the two-phase heuristic to those obtained from the ϵ-constraint method shows that the heuristic procedure is able to produce high-quality approximation sets within reasonable time.

The next step will consist in integrating additional real world constraints.

5 Introducing heterogeneous patients and vehicles

5.1 Introduction

In Chapter 3 of this book, a rather standard DARP has been solved. In Chapter 4, based on this standard version, the two-objective DARP has been introduced and a tailor-made heuristic as well as an exact solution method has been implemented. In the current chapter again the standard version of the DARP is extended. Instead of another objective, several real world motivated constraints are added to the basic model. They are derived from the real world ambulance routing problem situation faced by the ARC in the field of patient transportation. We denote this problem as heterogeneous DARP (HDARP). The term heterogeneous refers to heterogeneous vehicles (vehicles with different capacities) as well as heterogeneous passengers (patients demand different modes of transportation).

The ARC distinguishes three patient types. A patient may demand to be transported seated, on a stretcher, or in a wheelchair. In addition, an accompanying person may be present. The ARC disposes of two different types of vehicles. Each type provides different capacities for four modes of transportation (staff seat, patient seat, stretcher, wheelchair place). In the following these are referred to as resources. Resource 0 refers to staff seats, resource 1 to patient seats, resource 2 to stretchers, and resource 3 to wheelchair places. Each passenger can only be transported by a vehicle that disposes of the appropriate resource. Usually an accompanying person has to use a staff seat; a seated patient a patient seat; a patient needing a stretcher a stretcher; and a patient in a wheelchair a wheelchair place. However, certain so-called up-grading conditions apply; an accompanying person may use the patient seat, in case there is no vacant staff seat available, or, in case neither of the two is currently available, he/she can sit on the stretcher. According to Austrian law, a seated patient cannot use a staff seat, he/she may, however, sit on the stretcher, in case there is no additional patient seat available. Patients that demand a stretcher can only be transported on a stretcher. Patients in wheelchairs can only be transported by vehicles providing space for a wheelchair. As in the other two versions of the DARP dealt with so far, users specify time windows for either the pickup or the drop off location. Also ride time as well as maximum route duration limits have to be respected. The objective is to generate a transportation plan serving all patient transportation requests without violating any of the

above requirements at minimum routing costs. Furthermore, an option to penalize vehicle waiting time with passengers aboard shall be considered. The HDARP discussed in this chapter represents a further step towards the real world situation. Only the assignment of drivers and additional staff members to vehicles, where needed, is not considered. These aspects will be integrated in the subsequent chapter.

5.2 Related work

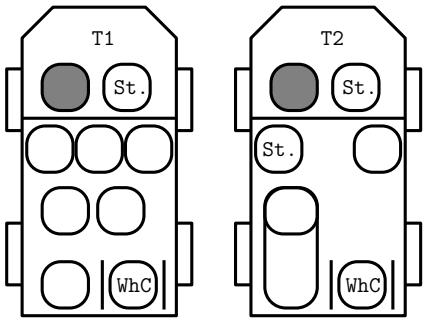
Previous publications, considering heterogeneous versions of the DARP, involve, e.g., the work of Toth and Vigo (1997b). The heterogeneity considered refers to two modes of transportation (seated passengers and passengers in wheelchairs) and several different types of vehicles. The authors devise a parallel insertion heuristic and a tabu thresholding algorithm for this version of the DARP. Another heterogeneous version of the DARP is described in Melachrinoudis et al. (2007). Several different types of vehicles in terms of capacity limits but only one mode of transportation are considered. The solution method developed is a tabu search algorithm. Heterogeneous vehicles in terms of different capacity limits for one mode of transportation in the context of the DARP are also considered in the work of Rekiek et al. (2006). The proposed problem is solved by a grouping genetic algorithm.

In a dynamic environment, Beaudry et al. (2008) adapt the tabu search heuristic developed by Cordeau and Laporte (2003b) to solve a heterogeneous DARP that arises in large hospitals. It involves transportation requests demanding three different modes of transportation (seated, on a bed, or in a wheelchair) and several different types of vehicles. Another implementation of a computer based planning system, considering hospital specific constraints such as multi-dimensional capacities, for the dynamic problem, in a large German hospital, is reported in Hanne et al. (2007).

The remainder of this chapter is organized as follows. First, the HDARP will be defined in further detail by means of a 3-index and a 2-index mathematical problem formulation. Each of these two formulations will serve as the basis for a branch and cut algorithm. Then, the VNS developed in Chapter 3, will be adapted to the HDARP to compute heuristic upper bounds. The two branch and cut algorithms and the VNS have been tested on random instances. Their results will be analysed.

5.3 Problem definition

In the following, first, the HDARP is described in further detail and the employed notation is explained. Then, the two mathematical problem formulations, a 3-index, and a 2-index formulation for the HDARP are proposed.



(a) Vehicle type 1 (b) Vehicle type 2

Figure 5.1: Vehicle types at the ARC

5.3.1 Notation

A set K of m heterogeneous vehicles has to serve all n transportation requests. Each vehicle $k \in K$ is associated with a vector $C^{s,k}$ that gives the amount of resource s available on vehicle k. The ARC disposes of two basic vehicle types. Type 1 (T1) provides 1 staff seat, 6 patient seats, and 1 wheelchair place. Type 2 (T2) provides 2 staff seats, 1 patient seat, 1 stretcher, and 1 wheelchair place (see Figure 5.1). Each route has to start at the start depot 0 and end at the end depot $2n + 1$, respecting a route duration limit T^k.

As all previously discussed versions of the DARP, the HDARP is modeled on a complete directed graph $G = (V, A)$ where V is the set of all vertices and A the set of all arcs. For each arc (i, j) and each vehicle k a non-negative travel cost c_{ij}^k and a non-negative travel time t_{ij}^k is considered. A total amount of n customer requests, each consisting of a pickup and delivery vertex pair $\{i, n + i\}$ have to be served. The set of pickup vertices is given by $P = \{1, \ldots, n\}$, the set of delivery vertices by $D = \{n + 1, \ldots, 2n\}$. At every pickup vertex one patient waits to be transported. This patient may demand one of three different modes of transportation. Passengers may have to be transported seated ($q_i^1 = 1$), on a stretcher ($q_i^2 = 1$), or in a wheelchair ($q_i^3 = 1$). Each patient may be accompanied by a friend, relative or nurse ($q_i^0 = 1$). The demand at every delivery vertex is equal to $q_{n+i}^s = -q_i^s$ for all $s \in R = \{0, 1, 2, 3\}$. Every user either specifies a time window $[e_i, l_i]$ for the pickup (origin) or the drop off (destination) location and beginning of service has to start within this time window. In case a vehicle arrives too early, it has to wait until service is possible. A maximum passenger ride time $\bar{L}$ is also considered, in order to keep quality of service at a reasonably high level. At each vertex loading or unloading operations last for a given service time d_i. Thus, the set of all vertices is given by $V = P \cup D \cup \{0, 2n + 1\}$, and the set of all arcs by $A = \{(i, j) | i \in V \setminus \{2n + 1\}, j \in V \setminus \{0\}, i \neq j\}$.

As mentioned above, so-called up-grading conditions apply. Patients demanding to be transported seated may use a patient seat or the stretcher. Patients demanding a stretcher can only be transported on a stretcher. The same applies to wheelchair passengers. Accom-

Table 5.1: Transportation mode up-grading options

Passenger type	staff seat	patient seat	stretcher	wheelchair place
accompanying person	X	X	X	
seated patient		X	X	
patient on stretcher			X	
patient in wheelchair				X

panying persons, however, may use a staff seat, a patient seat, or the stretcher, if no other transportation mode is available. Table 5.1 gives on overview of the different upgrading options.

During the optimization process the decision variables (listed below) are determined.

$$x_{ij}^k = \begin{cases} 1, & \text{if arc } (i,j) \text{ is traversed by vehicle } k \\ 0, & \text{else,} \end{cases}$$

$$y_i^k = \begin{cases} 1, & \text{if vehicle } k \text{ arrives with passengers at vertex } i \\ 0, & \text{else,} \end{cases}$$

B_i^k ... beginning of service of vehicle k at vertex i,
A_i^k ... arrival time of vehicle k at vertex i,
$\hat{W}_i^k$... waiting time of vehicle k with passengers aboard at vertex i,
$Q_i^{s,k}$... load of vehicle k of resource s when leaving vertex i,
L_i^k ... ride time of user i on vehicle k.

5.3.2 A 3-index formulation

The mathematical program, covering all of the above described real world conditions is based on the DARP formulation by Cordeau (2006),

$$\min \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}^k x_{ij}^k + \rho \sum_{k\in K}\sum_{i\in P\cup D} \hat{W}_i^k \tag{5.1}$$

subject to:

$$\sum_{k\in K}\sum_{j\in P\cup D} x_{ij}^k = 1 \qquad \forall i \in P, \tag{5.2}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{n+i,j}^k = 0 \qquad \forall i \in P, k \in K, \tag{5.3}$$

$$\sum_{j \in V} x_{0j}^k = 1 \qquad \forall k \in K, \tag{5.4}$$

$$\sum_{i \in V} x_{i,2n+1}^k = 1 \qquad \forall k \in K, \tag{5.5}$$

$$\sum_{i \in V} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall j \in P \cup D, k \in K, \tag{5.6}$$

$$x_{ij}^k = 1 \Rightarrow \quad Q_j^{s,k} \geq Q_i^{s,k} + q_j^s \qquad \forall i, j \in V, k \in K, s \in R, \tag{5.7}$$

$$\sum_{s'=s}^{2} Q_i^{s',k} \leq \sum_{s'=s}^{2} C^{s',k} \qquad \forall i \in V, k \in K, s \in R \setminus \{3\}, \tag{5.8}$$

$$Q_i^{3,k} \leq C^{3,k} \qquad \forall i \in V, k \in K, \tag{5.9}$$

$$Q_i^{s,k} \geq 0 \qquad \forall i \in V, k \in K, s \in R, \tag{5.10}$$

$$M y_i^k \geq \sum_{s \in R} (Q_i^{s,k} - q_i^s) \qquad \forall i \in P \cup D, k \in K, \tag{5.11}$$

$$x_{ij}^k = 1 \Rightarrow \quad A_j^k = B_i^k + d_i + t_{ij}^k \qquad \forall i, j \in V, k \in K, \tag{5.12}$$

$$B_i^k \geq A_i^k \qquad \forall i \in V, k \in K, \tag{5.13}$$

$$y_i^k = 1 \Rightarrow \quad \hat{W}_i^k \geq B_i^k - A_i^k \qquad \forall i \in P \cup D, k \in K, \tag{5.14}$$

$$L_i^k = B_{n+i}^k - (B_i^k + d_i) \qquad \forall i \in P, k \in K, \tag{5.15}$$

$$B_{2n+1}^k - B_0^k \leq T^k \qquad \forall k \in K, \tag{5.16}$$

$$e_i \leq B_i^k \leq l_i \qquad \forall i \in V, k \in K, \tag{5.17}$$

$$t_{i,n+i} \leq L_i^k \leq \bar{L} \qquad \forall i \in P, k \in K, \tag{5.18}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i, j \in V, k \in K, \tag{5.19}$$

$$y_i^k \in \{0, 1\} \qquad \forall i \in P, k \in K, \tag{5.20}$$

$$\hat{W}_i^k \geq 0 \qquad \forall i \in P \cup D, k \in K. \tag{5.21}$$

The objective function (5.1) minimizes total routing costs and penalizes waiting time when passengers are aboard the vehicle. The penalty term ρ will have to be defined by the ARC. Constraints (5.2) and (5.3) guarantee that each request is served exactly once and that each origin-destination pair is visited by the same vehicle. Equalities (5.4) – (5.6) ensure that each vehicle starts at and returns to the depot at the end of its route.

Consistency with respect to resource and load variables is guaranteed by constraints (5.7) – (5.9). Up-grading constraints for resources 0, 1, and 2 are given in (5.8). They guarantee that capacity restrictions regarding these resources are not violated and that each patient demanding resource 0, 1 or 2 can only be loaded if there is either enough capacity of the resource demanded or another one with a higher number (0 = staff seat, 1 = patient seat, 2 = stretcher). Constraints (5.9) guarantee that patients demanding resource 3 (wheelchair place) can only be transported if there is enough capacity of resource 3.

Equalities (5.12) define the arrival times for each vertex. Constraints (5.13) guarantee

that beginning of service only starts after having arrived at vertex i. Waiting times are set in (5.14). Waiting is only penalized if there is someone aboard the vehicle when arriving at vertex i. This is ensured by the way y_i^k are set in (5.11). Note that constraints (5.12) and (5.13) also take care of subtour elimination given that $(t_{ij} + d_i) > 0$ for all $i, j \in V, i \neq j$.

Equalities (5.15) define the ride time of each user. Total route duration is limited by (5.16), time window and maximum ride time compliance is ensured by (5.17) and (5.18).

5.3.2.1 Non-linear constraints

All non-linear formulations, given in (5.7) and (5.12), can be linearized (adapted from Cordeau, 2006). Load propagation inequalities (5.7) can be rewritten as

$$Q_j^{s,k} \geq (Q_i^{s,k} + q_j^s) - \mathcal{W}_{ij}^{s,k}(1 - x_{ij}^k), \tag{5.22}$$

with

$$\mathcal{W}_{ij}^{s,k} \geq \max\left\{\hat{C}^{s,k}, \hat{C}^{s,k} + q_i^s\right\} \quad \text{and} \quad \hat{C}^{s,k} = C^{s,k} + \sum_{s'=s}^{2} C^{s',k}.$$

As shown by Desrochers and Laporte (1991) they can then be lifted into

$$Q_j^{s,k} \geq (Q_i^{s,k} + q_j^s) - \mathcal{W}_{ij}^{s,k}(1 - x_{ij}^k) + (\mathcal{W}_{ij}^{s,k} - q_i^s - q_j^s)x_{ji}^k \tag{5.23}$$

for the DARP by taking the reverse arc into account. Time consistency constraints (5.12) are replaced by,

$$A_j^k \leq B_i^k + d_i + t_{ij}^k + l_{2n+1}(1 - x_{ij}^k), \tag{5.24}$$

$$A_j^k \geq B_i^k + d_i + t_{ij}^k - (l_i + d_i + t_{ij}^k)(1 - x_{ij}^k), \tag{5.25}$$

and those inequalities responsible for setting the waiting time correctly (5.14) by

$$\hat{W}_i^k \geq (B_i^k - A_i^k) - l_i(1 - y_i^k). \tag{5.26}$$

5.3.2.2 Variable aggregation

Since travel times are equal across all vehicles, t_{ij}^k can be replaced by t_{ij} and aggregate time variables A_i, B_i, L_i, and W_i can be used at all vertices except the two depots 0 and $2n+1$. Furthermore, y_i^k can be substituted by y_i. Thus, constraints (5.12), defining the arrival time at each vertex, are replaced by

$$x_{0j}^k = 1 \;\Rightarrow\quad A_j = B_0^k + d_i + t_{ij} \qquad\qquad \forall j \in V, k \in K, \tag{5.27}$$

$$\sum_{k \in K} x_{ij}^k = 1 \Rightarrow \quad A_j = B_i + d_i + t_{ij} \qquad \forall i \in P \cup D, j \in P \cup D, \tag{5.28}$$

$$x_{i,2n+1}^k = 1 \Rightarrow \quad A_{2n+1}^k = B_i + d_i + t_{ij} \qquad \forall i \in V, k \in K, \tag{5.29}$$

inequalities (5.13), ensuring that the beginning of service variables are correctly set, by

$$B_0^k \geq A_0^k \qquad \forall k \in K, \tag{5.30}$$

$$B_i \geq A_i \qquad \forall i \in P \cup D, \tag{5.31}$$

$$B_{2n+1}^k \geq A_{2n+1}^k \qquad \forall k \in K, \tag{5.32}$$

$$\tag{5.33}$$

constraints (5.14) and (5.11), responsible for defining a correct lower bound on the waiting time with passengers aboard, by

$$y_i = 1 \Rightarrow \quad \hat{W}_i \geq B_i - A_i \qquad \forall i \in P \cup D, \tag{5.34}$$

$$M y_i \geq \sum_{k \in K} \sum_{s \in R} (Q_i^{s,k} - q_i^s \sum_{j \in V} x_{ij}^k) \qquad \forall i \in P \cup D, \tag{5.35}$$

$$\tag{5.36}$$

and time window (5.17) and ride time constraints (5.15) and (5.18) by

$$e_0 \leq B_0^k \leq l_0 \qquad \forall k \in K, \tag{5.37}$$

$$e_i \leq B_i \leq l_i \qquad \forall i \in V, \tag{5.38}$$

$$e_{2n+1} \leq B_{2n+1}^k \leq l_{2n+1} \qquad \forall k \in K, \tag{5.39}$$

$$L_i = B_{n+i} - (B_i + d_i) \qquad \forall i \in P, \tag{5.40}$$

$$t_{i,n+i} \leq L_i \leq \bar{L} \qquad \forall i \in P. \tag{5.41}$$

The objective function becomes

$$\min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij}^k x_{ij}^k + \rho \sum_{i \in P \cup D} \hat{W}_i. \tag{5.42}$$

5.3.3 A 2-index formulation

When comparing the 3-index based branch and cut algorithm for the DARP of Cordeau (2006) to the 2-index based branch and cut algorithms proposed in Ropke et al. (2007), the 2-index versions obviously outperform the earlier 3-index based method. Given the additional complexity of our problem, it can be assumed that a branch and cut algorithm based on a 2-index formulation will also perform better than a branch and cut algorithm based on a 3-index program. However, the reformulation of the HDARP as a 2-index program is not as

straight forward as one may assume, given the heterogeneous vehicle fleet and the resulting differing capacity limits, that depend on which vehicle travels along which arc. How this issue can be resolved is described in the following.

In order to account for the heterogeneous nature of the vehicle fleet, artificial origin and destination depots for each vehicle have been introduced. A similar approach is followed, e.g., in Baldacci et al. (2007) in the context of the heterogeneous fleet size and mix vehicle routing problem. Let $D_o = \{2n+1, ..., 2n+m\}$ denote the set of vehicle origin depots and $D_d = \{2n+m+1, ..., 2n+2m\}$ the set of destination depots, the vertex set V is redefined as $V = P \cup D \cup D_o \cup D_d$ and the arc set as $A = \{(i,j) : i \in V \setminus D_d, j \in V \setminus D_o, i \neq j\}$. The demand/supply at the artificial depots is set to zero, $q^s_{2n+k} = q^s_{2n+m+k} = 0$ for all $k \in K, s \in R$. The total maximum vehicle capacity shall be defined as $\hat{C} = \sum_s \max_k C^{s,k}$.

To impose precedence and pairing constraints, let $\mathcal{S}$ denote the set of all vertex subsets $S \subseteq V$, such that for all $k \in K$ the respective origin depot $2n+k \in S$ and the destination depot $2n+m+k \notin S$, and there is at least one request i for which $i \notin S$ and $n+i \in S$. For the correct pairing of the artificial depots, define $\mathcal{U}$ as the set of all vertex subsets $U \subseteq V$, such that for exactly one $k \in K$ the origin depot $2n+k \in U$ and the destination depot $2n+m+k \notin U$, while for all other $l \in K \setminus \{k\}$ origin depots $2n+l \notin U$ and destination depots $2n+m+l \in U$.

Furthermore, to incorporate the according vehicle capacity constraints, let $\mathcal{H}$ denote the set of infeasible paths with respect to load violations $F = \{j_1, ..., j_h\}$, such that $j_1 \in D_o$ and let $A(F)$ denote the arc set of F. Omitting superscript k from all decision variables, the HDARP can be formulated as the following 2-index program (inspired by Ropke et al., 2007),

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \rho \sum_{i \in P \cup D} \hat{W}_i \quad , \tag{5.43}$$

subject to:

$$\sum_{i \in V \setminus D_d} x_{ij} = 1 \qquad \forall j \in P \cup D \cup D_d, \tag{5.44}$$

$$\sum_{j \in V \setminus D_o} x_{ij} = 1 \qquad \forall i \in P \cup D \cup D_o, \tag{5.45}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - (m+1) \qquad \forall S \in \mathcal{S}, \tag{5.46}$$

$$\sum_{i \in U} \sum_{j \notin U} x_{ij} \geq 1 \qquad \forall U \in \mathcal{U}, \tag{5.47}$$

$$x_{ij} = 1 \;\Rightarrow\quad Q_j \geq Q_i + \sum_{s \in R} q^s_i \qquad \forall i, j \in V, \tag{5.48}$$

$$\max\left\{0, \sum_{s\in R} q_i^s\right\} \leq Q_i \leq \min\left\{\hat{C}, \hat{C} + \sum_{s\in R} q_i^s\right\} \qquad \forall i \in V, \tag{5.49}$$

$$My_i \geq Q_i - \sum_{s\in R} q_i^s \qquad \forall i \in P \cup D, \tag{5.50}$$

$$\sum_{i=1}^{h-1} \sum_{l=i+1}^{h} x_{j_i,j_l} \leq |A(F)| - 1 \qquad \forall F \in \mathcal{H}, \tag{5.51}$$

$$x_{ij} = 1 \Rightarrow \quad A_j = B_i + d_i + t_{ij} \qquad \forall i, j \in V, \tag{5.52}$$

$$B_i \geq A_i \qquad \forall i \in V, \tag{5.53}$$

$$y_i = 1 \Rightarrow \quad \hat{W}_i \geq B_i - A_i \qquad \forall i \in P \cup D, \tag{5.54}$$

$$L_i = B_{n+i} - (B_i + d_i) \qquad \forall i \in P, \tag{5.55}$$

$$B_{2n+k} - B_{2n+m+k} \leq T^k \qquad \forall k \in K, \tag{5.56}$$

$$e_i \leq B_i \leq l_i \qquad \forall i \in V, \tag{5.57}$$

$$t_{i,n+i} \leq L_i \leq \bar{L} \qquad \forall i \in P, \tag{5.58}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, \tag{5.59}$$

$$y_i \in \{0, 1\} \qquad \forall i \in P, \tag{5.60}$$

$$\hat{W}_i \geq 0 \qquad \forall i \in V. \tag{5.61}$$

The objective function (5.43) minimizes total routing costs and penalizes waiting time with passengers aboard a vehicle. Constraints (5.44) and (5.45) guarantee that every vertex $i \in P \cup D$ is entered and left and that exactly one arc leaves (enters) every origin (destination) depot, i.e. $i \in D_o$ ($i \in D_d$). Precedence and pairing restrictions are guaranteed by (5.46) and (5.47). As shown by Ropke et al. (2007) inequalities (5.46), initially designed for the single vehicle case, also apply in a multi vehicle context. Since one depot per vehicle is considered in our mathematical model, the original right hand side $|S|-2$ has to be replaced by $|S|-(m+1)$. Suppose the set $S \in \mathcal{S}$ is $S = \{2n+1, \ldots, 2n+m, n+i\}$. From each start depot one arc has to leave S (i.e. m arcs). One of these arcs will be part of a path leading to $i \notin S$. If pairing and precedence constraints are respected, this path has to contain one arc leading back to $n+i \in S$. Finally, it has to arrive at one of the end depots, demanding another arc to be traversed leaving S and connecting to some end depot $2n+m+k \notin S$. Thus, $m+1$ arcs have to leave S, reducing the number of arcs that can be used within S to $|S|-(m+1)$. Constraints (5.47) are depicted in Figure 5.3.3. Here, d^{1+}, d^{2+}, and d^{3+} denote the origin depots of vehicles 1, 2, and 3, respectively, while d^{1-}, d^{2-}, and d^{3-} denote the corresponding destination depots. At least one arc has to leave set U (here U consists of d^{1+}, d^{2-}, and d^{3-} and probably some other vertices) and enter set $V \setminus U$ (represented as an arrow), in order to ensure the correct pairing of the depots (here d^{1+} and d^{1-}).

Aggregate loading variables are introduced and set by constraints (5.48). These are needed for the definition of y_i, defined in (5.50). The actual loading restrictions are taken care of

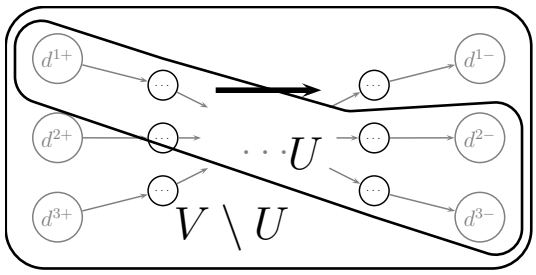


Figure 5.2: Pairing of artificial origin and destination depots

by inequalities (5.51); on every path $F \in \mathcal{H}$, each starting at one of the depots, a loading restriction is violated. Constraints of this type are known as tournament inequalities.

Arrival times are set in (5.52), beginning of service and waiting times in (5.53) and (5.54). Equalities (5.55) determine each user's ride time. Compliance with maximum route duration restrictions, time windows, and user ride time limits is guaranteed by (5.56) – (5.57).

5.3.3.1 Non-linear constraints

All non-linear inequalities are reformulated as shown for the 3-index model. Thus, load propagation constraints (5.48) are substituted by

$$Q_j \geq (Q_i + \sum_{s \in R} q_j^s) - \mathcal{W}_{ij}(1 - x_{ij}) \tag{5.62}$$

with

$$\mathcal{W}_{ij} \geq \max \left\{ \hat{C}, \hat{C} + \sum_{s \in R} q_i^s \right\},$$

and lifted into

$$Q_j \geq (Q_i + \sum_{s \in R} q_j^s) - \mathcal{W}_{ij}(1 - x_{ij}) + (\mathcal{W}_{ij} - \sum_{s \in R} q_i^s - \sum_{s \in R} q_j^s) x_{ji}, \tag{5.63}$$

by taking the reverse arc into account. Finally, arrival time inequalities (5.52) are replaced by,

$$A_j \leq (B_i + d_i + t_{ij}) + \max_k \{l_{2n+m+k}\} (1 - x_{ij}), \tag{5.64}$$

$$A_j \geq (B_i + d_i + t_{ij}) - (l_i + d_i + t_{ij})(1 - x_{ij}), \tag{5.65}$$

and vehicle waiting time inequalities (5.54) by

$$W_i \geq (B_i - A_i) - l_i(1 - y_i). \tag{5.66}$$

5.4 Valid inequalities

In the following section, the different valid inequalities that can be used to strengthen the above formulations are discussed. If not stated otherwise, $x_{ij} = \sum_k x_{ij}^k$ in the 3-index formulation.

5.4.1 Strengthened bounds on time and load variables

Bounds on time variables can be strengthened as follows (Cordeau, 2006):

$$B_i \geq e_i + \sum_{j \in V \setminus \{i\}} \max\{0, e_j - e_i + d_j + t_{ji}\}\, x_{ji} \tag{5.67}$$

$$B_i \leq l_i + \sum_{j \in V \setminus \{i\}} \max\{0, l_i - l_j + d_i + t_{ij}\}\, x_{ij} \tag{5.68}$$

In case of the 3-index formulation also bounds on load variables are strengthened (adapted from Cordeau, 2006). Let $orig(i)$ be the set that contains the origin of i, if i is a destination, otherwise the empty set, lower bounds can be strengthened as follows,

$$Q_i^{s,k} \geq \max\{0, q_i^s\} + \sum_{j \in V \setminus \{i, orig(i)\}} \max\left\{0, q_j^s\right\} x_{ji}^k, \tag{5.69}$$

$$\tag{5.70}$$

In case of upper bounds, one has to distinguish between origins and destinations. If vertex i is an origin ($1 \leq i \leq n$),

$$Q_i^{s,k} + \sum_{s'=s+1}^{2} Q_i^{s',k} \leq \hat{C}^{s,k} - \left(\hat{C}^{s,k} - \max_{j \in V \setminus \{i\}} \left\{\hat{q}_j^s\right\} - \hat{q}_i^s\right) x_{0i}^k - \sum_{j \in V \setminus \{i\}} \max\left\{0, \hat{q}_j^s\right\} x_{ij}^k, \tag{5.71}$$

with

$$\hat{C}^{s,k} = C^{s,k} + \sum_{s'=s+1}^{2} C^{s',k} \quad \text{and} \quad \hat{q}_i^s = q_i^s + \sum_{s'=s+1}^{2} q_i^{s'},$$

applies. If i is a destination ($n+1 \leq i \leq 2n$),

$$Q_i^{s,k} + \sum_{s'=s+1}^{2} Q_i^{s',k} \leq \min\left\{\hat{C}^{s,k}, \hat{C}^{s,k} + \hat{q}_i^s\right\} - \left(\hat{C}^{s,k} - \max_{j \in V \setminus \{i\}} \left\{\hat{q}_j^s\right\} - \hat{q}_i^s\right) x_{0i}^k, \tag{5.72}$$

is used. The last term used in (5.71) cannot be used to strengthen (5.72). Consider $\hat{C}^{s,k} = 1$, if $\hat{q}_i^s = -1$ and $\hat{q}_j^s = 1$ then $Q_i^{s,k} + \sum_{s'=s+1}^{s} Q_i^{s',k}$ would have to be ≤ -1 which is clearly not valid, although visiting j after i is feasible with respect to capacity limits.

5.4.2 Subtour elimination constraints

Standard subtour elimination constraints are given by $x(S) \leq |S| - 1$ for $S \subset P \cup D$. As shown by Cordeau (2006) these constraints can be lifted in several ways, exploiting the fact that each origin i must be visited before its destination $n + i$. Let $x(S) = \sum_{i,j \in S} x_{ij}$, $S = \{i_1, i_2, \ldots, i_h\} \subseteq P \cup D$, $\bar{S} = \{i \in P \cup D | i \notin S\}$. The predecessors of S are denoted as $\pi(S) = \{i \in P | n + i \in S\}$, its successors as $\sigma(S) = \{n + i \in D | i \in S\}$. The following four sets of inequalities are valid for the DARP (Cordeau, 2006) and are also applicable in the context of the HDARP:

$$x(S) + \sum_{i \in \bar{S} \cap \sigma(S)} \sum_{j \in S} x_{ij} + \sum_{i \in \bar{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} \leq |S| - 1, \quad (5.73)$$

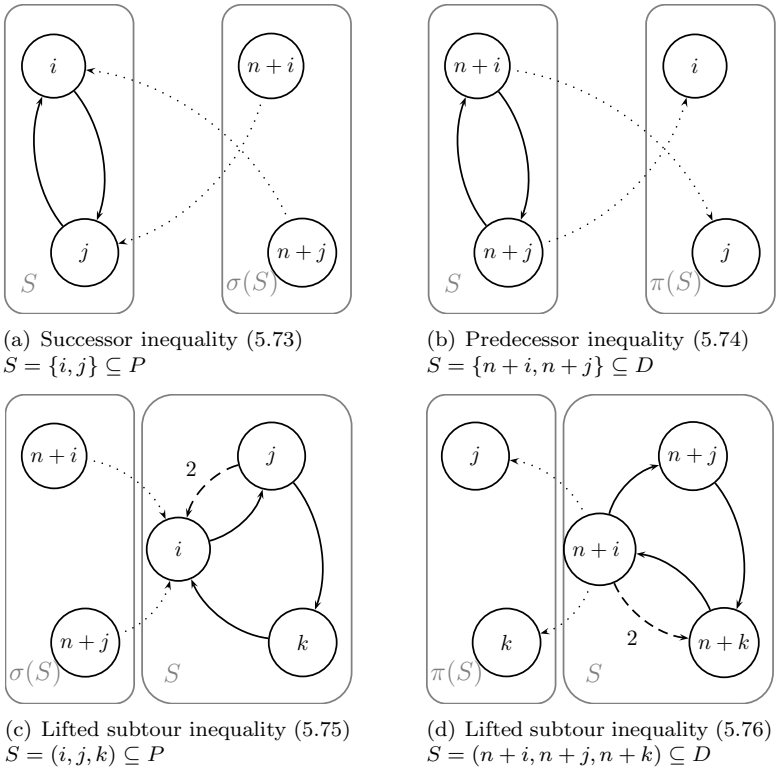
$$x(S) + \sum_{i \in S} \sum_{j \in \bar{S} \cap \pi(s)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in \bar{S} \setminus \pi(S)} x_{ij} \leq |S| - 1, \quad (5.74)$$

$$\sum_{j=1}^{h-1} x_{i_j, i_{j+1}} + x_{i_h, i_1} + 2 \sum_{j=2}^{h-1} x_{i_j, i_1} + \sum_{j=3}^{h-1} \sum_{l=2}^{j-1} x_{i_j, i_l} + \sum_{n+i_p \in \bar{S} \cap \sigma(S)} x_{n+i_p, i_1} \leq h - 1, \quad (5.75)$$

$$\sum_{j=1}^{h-1} x_{i_j, i_{j+1}} + x_{i_h, i_1} + 2 \sum_{j=3}^{h} x_{i_1, i_j} + \sum_{j=4}^{h} \sum_{j=3}^{j-1} x_{i_j, i_l} + \sum_{i_p \in \bar{S} \cap \pi(S)} x_{i_1, i_p} \leq h - 1. \quad (5.76)$$

The first two inequalities were originally introduced for the precedence constrained asymmetric TSP by Balas et al. (1995). They are also referred to as predecessor and successor inequalities, respectively. The ordering of the vertices in S does not play a role here. In the second two inequalities the ordering of the vertices is important. They are based on those of Grötschel and Padberg (1985) for the asymmetric TSP.

All four inequalities are illustrated in Figure 5.3. Dotted and dashed arcs represent liftings. In Figure 5.3(a) an example for successor inequality (5.73) is given. The set $S = \{i, j\}$ consists of two origin vertices. The successors of these two are their destinations $\sigma(S) = \{n + i, n + j\}$. Clearly, at most one ($|S| - 1 = 1$) of the four arcs shown in the figure can be used in a feasible solution. The same is true for the example of (5.74) in Figure 5.3(b). For inequalities (5.75) and (5.76) the reasoning is a bit different. Instead of sets, sequences of vertices are considered. Such that a sequence does not result in a cycle, at most $h - 1$ arcs can be used, given that the sequence consists of h arcs. If a reverse arc is used between two vertices, the vertices connected by this arc should no longer be connected to the rest of the sequence. This reduces the number of arcs that can be used in the sequence by two. That is way a reverse arc (dashed in the Figure) is counted twice. Furthermore, in



(a) Successor inequality (5.73)
$S = \{i, j\} \subseteq P$

(b) Predecessor inequality (5.74)
$S = \{n + i, n + j\} \subseteq D$

(c) Lifted subtour inequality (5.75)
$S = (i, j, k) \subseteq P$

(d) Lifted subtour inequality (5.76)
$S = (n + i, n + j, n + k) \subseteq D$

Figure 5.3: Lifted subtour elimination constraints (adapted from Cordeau, 2006)

case of inequality (5.75), illustrated in Figure 5.3(c), if a destination (successor) $\notin S$ of some origin $\in S$ is connected to the first vertex $\in S$, the first vertex cannot be connected to the origin of this destination anymore, reducing the number of vertices that can be used in the sequence by one. The same reasoning, considering predecessors instead of successors of the vertices $\in S$, leads to inequalities (5.76). An according example is given in Figure 5.3(d).

5.4.3 Generalized order constraints

Let $U_1, \ldots, U_h$ be mutually disjoint subsets and let $i_1, \ldots, i_h \in P$ be users such that either $0, 2n + 1 \notin U_l$ (3-index formulation) or $2n + k, 2n + m + k \notin U_l$ for all $k \in K$ (2-index formulation) and $i_l, n + i_{l+1} \in U_l$ for $l = 1, \ldots, h$ ($i_h + 1 = i_1$). The following inequalities, proposed by Ruland and Rodin (1997), are valid for the (H)DARP,

$$\sum_{l=1}^{h} x(U_l) \leq \sum_{l=1}^{h} |U_l| - h - 1. \tag{5.77}$$

For the (H)DARP these can be lifted in two ways as shown by Cordeau (2006),

$$\sum_{l=1} x(U_l) + \sum_{l=2}^{h-1} x_{i_1,i_l} + \sum_{l=3}^{h} x_{i_l,n+i_l} \leq \sum_{l=1}^{h} |U_l| - h - 1, \tag{5.78}$$

$$\sum_{l=1}^{h} x(U_l) + \sum_{l=2}^{h-2} x_{n+i_1,i_l} + \sum_{l=2}^{h} x_{n+i_1,n+i_l} \leq \sum_{l=1}^{h} |U_l| - h - 1. \tag{5.79}$$

5.4.4 Strengthened infeasible path constraints

Let $\mathcal{F}$ denote the set of infeasible paths. $A(F)$ is the arc set of F and $V(F)$ the vertex set, the following inequalities are valid for the (H)DARP (Ropke et al., 2007),

$$\sum_{(i,j)\in A(F)} x_{ij}^k \leq |A(F)| - 1 \qquad \forall F \in \mathcal{F}, k \in K. \tag{5.80}$$

If $F = (j_1, \ldots, j_h)$ denotes an infeasible path, they can be strengthened into so-called tournament constraints (see Ascheuer et al., 2000a; Ropke et al., 2007),

$$\sum_{i=1}^{h-1} \sum_{l=i+1}^{h} x_{j_i,j_l}^k \leq |A(F)| - 1. \tag{5.81}$$

If F links a vertex pair $\{i, n+i\}$, and $F = (i, j_1, \ldots, j_h, i+n)$ is infeasible due to time window or ride time constraints, (5.80) can be lifted in the following way,

$$x_{i,j_1}^k + \sum_{l=1}^{h-1} x_{j_l,j_{l+1}}^k + x_{j_h,n+i}^k \leq |A(F)| - 2. \tag{5.82}$$

If both path $F = (j_1, \ldots, j_h)$ and the reverse path $F' = (j_h, \ldots, j_1)$ are infeasible, the following inequality can be applied (Ropke et al., 2007),

$$\sum_{i=1}^{h-1} (x_{j_i,j_{i+1}}^k + x_{j_{i+1},j_i}^k) \leq h - 1. \tag{5.83}$$

These inequalities are generated for each vehicle in turn in case of the 3-index formulation. In case of the 2-index formulation x_{ij}^k has to be replaced by x_{ij} and only time related infeasibilities are considered since these restrictions are homogeneous across all vehicles.

5.4.5 Fork constraints

This family of inequalities, proposed by Ropke et al. (2007) in the context of the PDPTW and the DARP, also applies the notion of infeasible paths (regarding time infeasibilities). Here, they are eliminated by taking a whole bundle of such infeasible paths sharing some

common arcs into account. Let $F = (j_1, \ldots, j_h)$ be a feasible path. By adding a vertex at the beginning and at the end of this path an infeasible path is generated. A bundle of infeasible paths is defined by (i, F, l) being infeasible for every $i \in \hat{S}$ and $l \in \hat{T}$, $\hat{S}, \hat{T} \subset V$, resulting in the so-called fork inequality,

$$\sum_{i \in \hat{S}} x_{i,j_1} + \sum_{i=1}^{h-1} x_{j_i,j_{i+1}} + \sum_{l \in \hat{T}} x_{j_h,l} \leq h. \tag{5.84}$$

These inequalities can be lifted in two ways. In the first, an additional set of vertices $\hat{T}_i$ is appended to each vertex $j_i \in \{j_1, \ldots, j_{h-1}\}$ resulting in sequences $(k, j_1 \ldots j_i, l)$ that are infeasible for every $k \in \hat{S}$, $l \in \hat{T}_i$, $1 \leq i \leq h$. This so-called outfork inequality is given by

$$\sum_{i \in \hat{S}} x_{i,j_1} + \sum_{i=1}^{h-1} x_{j_i,j_{i+1}} + \sum_{i=1}^{h} \sum_{l \in \hat{T}_i} x_{j_i,l} \leq h. \tag{5.85}$$

In the second, a set of vertices $\hat{S}_i$ is added before each vertex $j_i \in \{j_2, \ldots, j_h\}$, leading to sequences $(k, j_i \ldots j_h, l)$ that are infeasible for every $k \in \hat{S}_i$, $l \in \hat{T}$, $1 \leq i \leq h$. This so-called infork inequality is given by

$$\sum_{k \in \hat{S}_i} \sum_{i=1}^{h} x_{k,j_i} + \sum_{i=1}^{h-1} x_{j_i,j_{i+1}} + \sum_{l \in \hat{T}} x_{j_h,l} \leq h. \tag{5.86}$$

5.4.6 Adapted rounded capacity inequalities (only 3-index formulation)

Let $\hat{R}(S)$ denote the minimum number of vehicles necessary to visit all nodes in $S \subseteq P \cup D$, the constraint $x(\Delta(S)) \geq 2\hat{R}(S)$ is a valid inequality for the DARP (Cordeau, 2006), $\Delta(S) = \sum_k \sum_{i \notin S} \sum_{j \in S} x_{ij}^k + \sum_k \sum_{i \in S} \sum_{j \notin S} x_{ij}^k$. A lower approximation of $\hat{R}(S)$ is given by $\lceil q(S)/C \rceil$, where $q(S) = \sum_{i \in S} q_i$.
This so-called rounded capacity inequality can be adapted to the HDARP as follows. Let $\hat{C}^{s,k}$ denote the cumulative capacity limit for resource s on vehicle k and be $q^s(S) = \sum_{i \in S}(q_i^s + \sum_{s'=s+1}^{2} q_i^{s'})$, the constraint $x(\Delta(S)) \geq 2\lceil q^s(S) / \max_k \hat{C}^{s,k} \rceil$ is a valid rounded capacity inequality for the HDARP.

5.4.7 Strengthened capacity inequalities (only 3-index formulation)

Introduced in Ropke et al. (2007), the strengthened capacity cuts for the DARP can be applied to the HDARP. Let $\hat{S}$,$\hat{T} \subseteq P \cup D$ denote two disjoint subsets such that $q(S) > 0$. Furthermore, let $\hat{U} = \pi(\hat{T}) \setminus (\hat{S} \cup \hat{T})$, the strengthened capacity inequality for the DARP is

$$x(\hat{S}) + x(\hat{T}) + x(\hat{S} : \hat{T}) \leq |\hat{S}| + |\hat{T}| - \left\lceil \frac{q(\hat{S}) - q(\hat{U})}{C} \right\rceil, \tag{5.87}$$

with $x(\hat{S} : \hat{T}) = \sum_{i \in \hat{S}} \sum_{j \in \hat{T}} x_{ij}$. In case of the HDARP $q(.)$ has to be replaced by $q^s(.)$ as defined above, and instead of C, $\max_k \hat{C}^{s,k}$ has to be used.

5.4.8 Reachability constraints (only 2-index formulation)

This group of inequalities, introduced in Laysgaard (2006) and used in the context of the DARP by Ropke et al. (2007), is based on the notion of conflicting vertices. Vertices are said to be conflicting if they cannot be served by the same vehicle. For every vertex $i \in P \cup D$ let $A_i^- \subset A$ define the minimum arc set such that any feasible sequence from an origin depot $2n + k$ $(k \in K)$ to vertex i can be constructed only using arcs $\in A_i^-$. Furthermore, let $A_i^+ \subset A$ denote the minimum arc set such that any feasible sequence from i to a destination depot $2n + m + k$ $(k \in K)$ only traverses arcs $\in A_i^+$. Finally, let $\hat{T}$ be a set of conflicting vertices. $A_{\hat{T}}^- = \cup_{i \in \hat{T}} A_i^-$ defines the reaching arc set of $\hat{T}$ and $A_{\hat{T}}^+ = \cup_{i \in \hat{T}} A_i^+$ the reachable arc set of $\hat{T}$. The following two inequalities can be defined,

$$x(\Delta^-(\hat{S}) \cap A_{\hat{T}}^-) \geq |\hat{T}|, \tag{5.88}$$

$$x(\Delta^+(\hat{S}) \cap A_{\hat{T}}^+) \geq |\hat{T}|, \tag{5.89}$$

with $\Delta^-(\hat{S}) = \sum_{i \notin \hat{S}} \sum_{j \in \hat{S}} x_{ij}$ and $\Delta^+(\hat{S}) = \sum_{i \in \hat{S}} \sum_{j \notin \hat{S}} x_{ij}$.

5.5 Branch and cut algorithms

In the following section, first, the branch and cut methodology is described. Then, the pre-processing steps employed and the different separation procedures are briefly discussed.

5.5.1 The branch and cut framework

As briefly explained in Chapter 2, branch and cut algorithms combine the branch and bound and the cutting plane idea. Every IP can be reformulated as a LP by dropping the integrality constraints for all variables. In our case, these are (5.19) and (5.20) in the 3-index, and (5.59) and (5.60) in the 2-index formulation. The optimal solution to the LP relaxation will yield a lower bound for the IP.

The branch and cut algorithm departs from the solution of the LP relaxation, considering only a reasonable subset of the original constraints. Typically, all constraint families of exponential size are not included. In our case these are the pairing constraints (5.46) and (5.47), and the infeasible path constraints (5.46) and (5.47) in the 2-index program introduced above. Also all families of valid inequalities that are only used to strengthen the model are added in a cutting plane fashion. Separation algorithms will then check the current solution for violations of the omitted constraints and valid inequalities. In case of omitted constraints, the separation procedures have to correspond to exact procedures such

that, if one of the omitted constraints is violated, one can be sure that it will be identified. In case of additional valid inequalities, which are not needed to ensure feasibility, also heuristic separation procedures can be employed.

In case at least one violated constraint is detected by the separation procedures, these are added to the current LP in the form of cuts and the updated LP is solved again. This process is repeated until the separation procedures fail to detect additional violated constraints. In this case, the optimal solution to the LP relaxation of the original mixed IP has been found. If this solution is integer, the optimal solution to the mixed IP has been found.

Otherwise, the problem is decomposed into two new problems. As in branch and bound, by branching on a variable that is associated with a fractional value in the current solution, e.g. on $x_{1,3}$, if $x_{1,3} = 0.4$. Branching refers to the generation of two child nodes. At one of the child nodes an LP is built with an additional constraint setting a lower bound on the chosen variable. This lower bound is equal to the fractional value of the variable ceiled to the next integer, e.g. $x_{1,3} \geq 1$. At the other child node another LP is built with an additional constraint setting an upper bound on the chosen variable. This upper bound is equal to the fractional value of the chosen variable floored to the next integer, e.g. $x_{1,3} \leq 0$. Then each of the two LPs is solved in the same way as the first LP relaxation; cuts are added until no more violated inequalities can be detected. In case the optimal solution is not integer, the child node serves as a new parent node for two new child nodes in the tree. The optimal solution to the original mixed IP is the best of the first two thus recursively solved problems. This summary of the branch and cut method is widely based on Naddef and Rinaldi (2002). Further details and additional references can be found in the same paper.

To accelerate the solution process several pre-processing steps can be performed prior to starting the optimization procedure. These steps refer to graph pruning and time window tightening techniques. Furthermore, some "easy" cuts can be generated in advance to strengthen the formulation (initial cut pool); and, in order to break symmetry, in case of the 3-index formulation, variable fixing techniques can be employed; e.g. in the homogeneous fleet case, those requests that cannot be served by the same vehicle are fixed to different vehicles. Our branch and cut implementations are based on Cordeau (2006) and Ropke et al. (2007). The employed pre-processing steps and the different separation procedures are briefly described in the following.

5.5.2 Pre-processing

In a first step, the graph pruning and time window tightening techniques as described by Cordeau (2006), see also Section 3.4.1 in Chapter 3, adapted to the HDARP are applied.

Then an initial cut pool is generated. This pool contains all cuts part of the initial pool of inequalities described in Cordeau (2006): strengthened bounds on time and load variables,

subtour elimination constraints for $|S| = 2$, four particular cases of precedence constraints, generalized order constraints for $h = 2$ and $|U_1| = |U_2| = 2$, infeasible path constraints of the form $x_{ij} + x_{j,n+j}x_{n+j,n+i} \leq 1$ if this path violates the ride time limit of request i. Furthermore, four inequalities for each pair of incompatible users $i, j \in P$ and every node $l \in P \cup D$ are added to the pool:

$$\begin{aligned} x_{il} + x_{li} + x_{lj} + x_{jl} &\leq 1 \\ x_{il} + x_{li} + x_{l,n+j} + x_{n+j,l} &\leq 1 \\ x_{n+i,l} + x_{l,n+i} + x_{lj} + x_{jl} &\leq 1 \\ x_{n+i,l} + x_{l,n+i} + x_{l,n+j} + x_{n+j,l} &\leq 1 \end{aligned}$$

Finally, in case of the 3-index based branch and cut algorithm, variable fixing methods are used. If the vehicle fleet of an instance is homogeneous the same variable fixing procedures as in Cordeau (2006) are applied. If the fleet is heterogeneous variables can be fixed as follows. If only one vehicle with resource 2 or 3 (stretcher or wheelchair) is available, all users demanding these modes of transportation can be fixed to the corresponding vehicle. In any case, if some vehicle does not provide these resources all users demanding them can be forbidden on this vehicle. Furthermore, if two users are identified as incompatible for one vehicle, a constraint guaranteeing that only one of the two can use this vehicle is appended to the model.

5.5.3 Separation heuristics

In both algorithms, violated subtour elimination and generalized order constraints are separated by means of several (meta)heuristics as described in Cordeau (2006). For the separation of strengthened infeasible path inequalities, an enumerative procedure as described in Ropke et al. (2007) is used. Due to heterogeneous fleet requirements, these constraints are checked and generated for each vehicle in turn in case of the 3-index formulation. The same applies to violated fork constraints which are also determined by means of enumeration procedures (Ropke et al., 2007). Regarding the 2-index formulation, only time related infeasibilities are considered. These are homogeneous across all vehicles.

In case of the 3-index based branch and cut algorithm, as in Cordeau (2006), for the separation of rounded capacity inequalities, a tabu search algorithm is employed. To detect violated strengthened capacity cuts, the heuristics as described by Ropke et al. (2007) are used, i.e. a randomized construction heuristic and another tabu search algorithm.

For the 2-index branch and cut algorithm additional separation procedures need to be devised. Precedence and pairing constraints, given in (5.46) are separated similar to Ropke et al. (2007). The implementation of Goldberg's algorithm for solving the individual max-flow-problems is used.

The pairing constraints in the 2-index model regarding the introduced artificial depots are separated by a similar procedure. Every origin depot together with all other destination depots are in turn used as the super source node and its corresponding destination depot together with all other origin depots as the super sink node. If the max flow on the network (arc capacities are defined by the current values of the x_{ij} variables) is smaller than one, the according depot pairing constraint, given in (5.47), is generated.

For the detection of infeasible paths regarding heterogeneous vehicles and resources violating a capacity restriction in the 2-index formulation, another enumerative procedure is applied. Every path starting at one of the start depots is extended on all arcs with some flow ($x_{ij} > 0$) and checked for capacity violations. If the capacity is exceeded at some vertex i, the corresponding inequality (5.51) is generated. Path extension ends as soon as one of the destination depots has been reached.

As in Ropke et al. (2007), to separate reachability cuts in the 2-index based algorithm, in a first step, for each vertex $i \in P \cup D$ the arc sets A_i^+ and A_i^- are computed. Based on these arc sets conflicting vertex sets are identified (if there is no path that contains a certain vertex j leading from one of the start depots to i, or from i to one of the destination depots, i and j are conflicting vertices). After having determined all conflicting pairs, conflicting sets of larger cardinality are generated on the basis of conflicting pairs. Obviously, at most sets with a cardinality of m are considered. If sets of higher cardinality existed, the respective problem instance would be infeasible. At each fractional solution encountered at some node of the branch and bound tree, each conflicting set $\hat{T}$ is considered in turn. In order to identify violated inequalities (5.88), a maximum flow problem between D_o and $\hat{T}$ is solved using only arcs $\in A_{\hat{T}}^-$. In order to do so, an artificial source, that is connected to each vertex $\in D_o$ with arcs of infinite capacity, and an artificial sink, connecting all vertices $\in \hat{T}$ with arcs of infinite capacity to this sink, have to be introduced. If the minimum cut's capacity obtained is smaller than $|\hat{T}|$, the according reachability cut is generated. Similarly, to identify violated inequalities (5.89), a max flow problem between $\hat{T}$ and D_d is solved using only arcs $\in A_{\hat{T}}^+$.

5.5.4 Heuristic upper bounds

In order to accelerate the optimization process initial upper bounds are calculated by means of an adapted version of *heurVNS*, designed for the standard DARP in Chapter 3. The current version (*heurVNShet*) differs from *heurVNS* in several ways.

First, a new evaluation function that accommodates the characteristics of the HDARP is devised. Besides routing costs and constraint violations, also the total waiting time with passenger aboard has to be considered,

$$\hat{f}(\int) = \hat{c}(\int) + \rho\hat{v}(\int) + \sum_{s \in R} \hat{\alpha}_s \hat{q}_s(\int) + \hat{\beta}\hat{d}(\int) + \hat{\gamma}\hat{w}(\int) + \hat{\tau}\hat{t}(\int). \tag{5.90}$$

The routing costs of a solution $\int$ are given by $\hat{c}(\int)$, the total vehicle waiting time with passengers aboard by $\hat{v}(\int)$, weighted by ρ. The terms $\hat{q}_s(\int)$ refer to the different resource violations $s \in R$. Each of these is penalized by a separate penalization parameter $\hat{\alpha}_s$. All other terms correspond to the penalization terms used in *heurVNS*. Note that as in *heurVNS*, a solution $\int$ can only become a new best solution $\int_{best}$ if $\hat{q}_s(\int) = \hat{d}(\int) = \hat{w}(\int) = \hat{t}(\int) = 0$ for all $s \in R$.

Second, a regret insertion procedure for the construction of the initial solution has been designed. The first m requests are each assigned to a different vehicle. All subsequent requests are inserted as follows. A regret value for each request is calculated: the best insertion position for each request on each route is determined and the corresponding evaluation function values are calculated; the regret value for a given request corresponds to the difference in terms of evaluation function value between its two best insertion positions. In every iteration the request associated with the largest regret value is inserted at its best position.

Third, only 13 different neighborhoods are considered (S1 – M1 – C1 – S2 – M2 – C2 – S3 – M3 – C3 – S4 – M4 – C4 – Z, S = Swap neighborhood, M = Move Neighborhood, C = Chain neighborhood, Z = Zero split neighborhood, see Chapter 3). Initial tests with 19 neighborhoods showed that, as in Chapter 4, 13 neighborhoods are sufficient.

Further, the above mentioned graph pruning techniques are only applied after the first feasible solution has been found. And last but not least, as stopping criterion a limit of $5\text{x}10^5$ iterations is employed.

5.6 Computational experiments

All programs were implemented in C++. In the branch and cut algorithms CPLEX 11.0 together with Concert Technology 2.5 were used. All experiments were carried out on a 3.2 GHz Pentium D computer with a memory of 4 GB. All solution procedures have been tested on three artificial data sets. These are based on an existing data set from the literature, enriched with the different real world characteristics described above. In the following, first, the characteristics of the generated test instances are described. Then, the results obtained are discussed. This part is split into two sections. The first summarizes the results obtained by means of the two branch and cut algorithms, the second presents the solutions obtained by means of *heurVNShet*. Both procedures are first tested with $\omega = 0$ (waiting is not penalized) and then with $\omega = 100$.

5.6.1 Test instances

The test instances used are based on the 12 “A” instances proposed by Cordeau (2006) for the standard DARP, containing 12 instances with 2 to 4 vehicles and 16 to 48 requests. In

Table 5.2: Characteristics of test instances

data set	probability for patient to be seated	on stretcher	in wheelchair	probability for AP	fleet
U	1.00	0.00	0.00	0.00	hom. (T0)
E	0.50	0.25	0.25	0.10	hom. (T2)
I	0.83	0.11	0.06	0.50	het. (T1, T2)

AP = Accompanying Person, hom. = homogeneous, het. = heterogeneous
T0: 3 patient seats
T1: 1 staff seats, 6 patient seats, 1 wheelchair place
T2: 2 staff seats, 1 patient seat, 1 stretcher, 1 wheelchair place

all instances time window length $l_i - e_i = 15$ minutes, maximum user ride time $\bar{L} = 30$ minutes, and service time $d_i = 3$ minutes for all users (see also entry "Cor06" in Table 2.4 in Chapter 2 for a summary). For each instance three instances with different degrees of heterogeneity were generated.

Heterogeneous users were introduced based on the probabilities given in Table 5.2. Every transportation request consists of at most one patient. In data set "U" only seated passengers and no accompanying persons are considered. In data set "E" half of the patients are considered as seated patients, 25% as patients on a stretcher, and 25% as patients in a wheelchair; 10% are assumed to be accompanied by someone. Eventually, in data set "I" users are transformed into seated patients, patients on a stretcher, and wheelchair passengers, according to the true distribution across all static transports carried out by the ARC in the city of Graz. Furthermore, half of them are assumed to be accompanied by someone.

Regarding the vehicle types employed in the different instances, in data set "U" a homogeneous fleet setting with only vehicles of type T0 is considered. Vehicle type T0 provides space for three seated passengers. In data set "E" a homogeneous vehicle fleet consisting of T2 vehicles, disposing of 2 staff seats, 1 patient seat, 1 stretcher, and 1 wheelchair place, is used; while for data set "I" a heterogeneous vehicle fleet has been generated. Here the original number of vehicles has been randomly divided into T1 and T2 vehicles such that at least one vehicle of each type is available. Vehicle types T1 and T2 are derived from data provided by the ARC regarding their vehicle fleet.

5.6.2 Branch and cut results

Table 5.3 provides the results obtained by the 3-index and the 2-index branch and cut algorithms, ignoring the penalization option regarding waiting with passengers aboard ($\rho = 0$). The two algorithms will be denoted as *3indexBC* and *2indexBC* in the following. The table contains the following information. Columns "z" give the proven optimal objective value (marked by an asterisk) or the best feasible solution found. Furthermore, the best lower bound identified within the time limit (bestLB), run times in seconds (CPU), the number of nodes visited as well as the number of cuts added are provided. The number of cuts refers

to the number of user cuts generated during the optimization process. Cuts generated by CPLEX are not counted. In the last column (heurUB) the employed initial upper bounds, computed by means of one run of *heurVNShet*, can be found. For all experiments reported in this section a run time limit of four hours was used. When comparing the results obtained by the two branch and cut algorithms, *2indexBC* clearly outperforms *3indexBC*. *2indexBC* finds the optimal solution to 29 out of 36 test instances within the 4-hour time limit, while *3indexBC* only finds 13. Only in one case (instance a4-24 of data set "E"), *3indexBC* finds the optimal solution and proves it to be optimal, while *2indexBC* does not. In *2indexBC* on average five times more cuts are generated but only half of the number of nodes are explored. In many cases the optimal solution is already found at the root node in *2indexBC*, while this is never the case for *3indexBC*. The conclusion that can be drawn from these results is that, as expected, the 2-index based formulation leads to a more efficient branch and cut algorithm. Therefore, in the following experiment only *2indexBC* is considered.

Table 5.4 provides the results obtained with *2indexBC* and $\rho = 100$. This means that waiting with passengers aboard is penalized rather severely. All results thus obtained do not contain any waiting time with passengers aboard. Penalizing waiting with passengers aboard makes the problem slightly more difficult to solve. Instead of 29 out of 36 instances, 28 out of 36 instances can be solved to optimality within the 4-hour time limit.

5.6.3 Heuristic results

Finally, Table 5.5 gives the results obtained by means of *heurVNShet* over five random runs with $\rho = 0$ and $\rho = 100$. For each data set the best value, the mean value, the deviation from the optimal solution, where known, the run time in seconds, and the total waiting time with passengers aboard (averaged over 5 runs) are given. *heurVNShet* is able to cope with both settings. The largest deviation from the optimal solution in case of $\rho = 0$ is 1.41%, taking average values over 5 runs. When considering the best values out of these 5 runs, the maximum gap is equal to 0.74%. In most cases the optimum solution is found. The largest deviation in case of $\rho = 100$, taking average values over 5 runs, is 1.06%. The largest gap, taking the best solution values out of these 5 runs, is equal to 0.27%. This indicates that for *heurVNShet*, penalizing waiting with passengers aboard makes the problem slightly easier to solve. In both versions run times are very low, less than 2 minutes on average. When comparing the amount of waiting time contained in the average solutions, setting $\rho = 0$ results in, on average, 18.98 minutes of waiting, while setting $\rho = 100$ results in no waiting for the passengers. Only for one instance (a4-32 data set "I") one of the 5 random runs yielded a solution with passenger waiting time (0.08 seconds). When comparing average routing costs for $\rho = 0$ and $\rho = 100$ a cost increase of about 2.5% can be observed. When comparing best values out of 5 runs, the largest cost increase amounts to almost 6% (instance a4-48 of data set "E"). The implications from a company perspective are that only moderate

Table 5.3: *3indexBC* vs. *2indexBC* ($\rho = 0$)

	3indexBC					*2indexBC*					
	z	bestLB	CPU[a]	nodes	cuts	z	bestLB	CPU[a]	nodes	cuts	heurUB
U											
a2-16	294.25*	294.25	44.05	58	152	294.25*	294.25	1.13	0	44	294.25
a2-20	344.83*	344.80	1674.30	6751	2531	344.83*	344.83	2.59	0	112	344.83
a2-24		415.17	14492.70	13632	2391	431.12*	431.12	8.54	0	161	431.12
a3-18	300.48*	300.45	4376.70	16350	933	300.48*	300.48	4.55	0	249	300.48
a3-24		334	14516.40	7380	595	344.83*	344.83	7.62	0	422	347.42
a3-30		467.25	14490	3953	433	494.85*	494.85	9.83	0	575	494.85
a3-36		553.05	14570.80	3917	351	583.19*	583.19	105.05	0	1099	584.44
a4-16	282.68*	282.67	319.42	284	241	282.68*	282.68	5.61	0	430	282.68
a4-24	375.02*	375.02	709.89	187	240	375.02*	375.02	5.60	0	347	378.13
a4-32		432.78	14520.60	2084	291	485.50*	485.50	30.67	0	1056	487.81
a4-40		502.74	14468.60	88	339	557.69*	557.63	8328.46	9723	6293	582.26
a4-48		571.27	14523.20	48	340	668.82	664.64	14542.60	5076	5164	709.47
$\overline{U}$		406.12	9058.89	4561	736		429.92	1921.02	1233	1329	436.48
E											
a2-16	331.16*	331.16	37.54	14	101	331.16*	331.13	284.17	4908	1461	331.16
a2-20	347.03*	347.03	441.26	1523	817	347.03*	347.03	8.06	0	150	347.03
a2-24		421.63	14537.70	7198	1255	450.25*	450.21	891.24	5743	1298	450.25
a3-18	300.63*	300.63	4412.52	13053	612	300.63*	300.63	4.28	0	304	300.63
a3-24		340.65	14528.70	6721	505	344.91*	344.91	10.15	0	505	346.22
a3-30		484.26	14514.80	3199	445	500.58*	500.53	1608.63	4898	2600	500.58
a3-36		565.55	14554.10	2165	312	583.19*	583.19	101.40	0	1133	585.94
a4-16	285.99*	285.99	194.69	31	155	285.99*	285.99	759.27	7664	1832	285.99
a4-24	383.84*	383.84	1321.24	112	383	383.84	380.48	14471.60	19808	22083	390.87
a4-32		459.87	14499.90	870	562		488.14	14494.10	4733	12610	508.51
a4-40		534.35	14716.30	43	516		558.09	14518.50	2390	17918	609.08
a4-48		593.36	14625.30	0	635		665.02	14539.50	1476	11919	704.07
$\overline{E}$		420.69	9032.00	2911	525		436.28	5140.08	4302	6151	446.69
I											
a2-16	294.25*	294.25	92.17	59	148	294.25*	294.25	0.88	0	25	294.25
a2-20	355.74*	355.74	10142.80	42432	3728	355.74*	355.71	115.44	1103	428	355.74
a2-24		421.75	14553.30	13825	1324	431.12*	431.12	7.06	0	236	431.12
a3-18		297.02	14538.70	17172	651	302.17*	302.15	30.78	187	515	302.17
a3-24		332.49	14528.50	5570	354	344.83*	344.83	6.78	0	436	344.83
a3-30		472.35	14490.80	2896	481	494.85*	494.85	9.50	0	574	494.85
a3-36		566.23	14503.70	1173	263	618.63	592.69	14500.10	8742	17455	625.90
a4-16	299.05*	299.05	360.75	804	296	299.05*	299.05	36.51	35	606	299.05
a4-24	375.07	372.07	14499.20	4221	244	375.02*	375.02	5.23	0	391	375.07
a4-32		419.60	14497.50	1073	225	486.93*	486.93	40.63	0	1093	498.48
a4-40		499.03	14525.60	23	188	557.69*	557.63	2600.02	2093	4692	583.53
a4-48		567.51	14630.20	32	276	678.59	663.30	14498.00	1820	9526	708.68
$\overline{I}$		408.09	11780.27	7440	682		433.13	2654.25	1165	2998	442.81
$\overline{UEI}$		411.64	9957.05	4971	648		433.11	3238.45	2233	3493	441.99

[a] run times in seconds

cost increases will lead to improved solutions from a customer perspective. What remains to be seen is how this relationship translates into larger problem instances. Possibly the difference between the two extremes, only minimizing costs and avoiding user waiting time when aboard a vehicle, will increase.

Table 5.4: *2indexBC* ($\rho = 100$)

	2indexBC					
	z	bestLB	CPU[a]	nodes	cuts	heurUB
U						
a2-16	300.17*	300.17	0.92	0	63	300.17
a2-20	345.74*	345.74	2.43	0	117	345.74
a2-24	448.07*	448.07	15.67	0	147	448.07
a3-18	316.78*	316.78	3.59	0	204	316.78
a3-24	346.97*	346.97	11.55	0	516	347.37
a3-30	501.68*	501.68	23.46	0	549	502.39
a3-36	598.53*	598.53	21.55	0	646	599.55
a4-16	282.68*	282.68	5.54	0	357	282.68
a4-24	386.38*	386.38	5.73	0	398	386.38
a4-32	493.15*	493.15	83.48	0	1200	493.15
a4-40	557.94*	557.90	98.35	0	1204	560.31
a4-48	707.89	697.88	14467.20	4514	4132	711.12
$\overline{U}$		439.66	1228.29	376	794	441.14
E						
a2-16	331.16*	331.13	92.46	1406	657	331.16
a2-20	347.89*	347.89	4.76	0	77	347.89
a2-24	461.93*	461.89	231.05	1310	582	462.82
a3-18	316.78*	316.78	5.99	0	224	316.78
a3-24	347.05*	347.05	17.32	0	637	347.45
a3-30	501.68*	501.68	27.00	0	617	504.15
a3-36	604.35*	604.30	390.94	447	1820	606.08
a4-16	291.55	288.70	14479.30	76702	10520	291.55
a4-24	386.38*	386.38	5.66	0	336	386.38
a4-32		491.66	14490.20	4836	9864	507.73
a4-40		560.00	14507.70	2387	21330	590.19
a4-48		697.28	14498.10	2355	7316	717.50
$\overline{E}$		444.56	4895.87	7454	4498	450.81
I						
a2-16	300.17*	300.17	0.99	0	39	300.17
a2-20	356.64*	356.61	28.06	42	205	356.64
a2-24	450.37*	450.37	30.04	21	251	450.37
a3-18	318.47*	318.47	17.17	0	311	318.47
a3-24	346.97*	346.97	16.57	0	644	347.37
a3-30	501.68*	501.68	10.45	0	478	501.68
a3-36	631.12	612.35	14511.20	9386	18371	633.29
a4-16	301.81*	301.78	198.41	1564	1244	302.23
a4-24	386.38*	386.38	7.35	0	396	386.38
a4-32	494.59	493.58	14488.90	18973	9994	499.06
a4-40	559.45*	559.42	223.67	51	1589	561.35
a4-48		698.39	14516.60	2125	8138	721.40
$\overline{I}$		443.85	3670.78	2680	3472	448.20
$\overline{UEI}$		434.35	3264.96	3503	2920	438.38

[a] run times in seconds

5.7 Summary

In this chapter the first step towards modeling the true ambulance routing problem of the ARC has been done by introducing heterogeneous vehicles and users into the standard DARP. Two problem formulations have been devised. When used in a branch and cut framework, the 2-index formulation clearly outperforms the 3-index based one in all three data sets considered. The variable neighborhood search heuristic of Chapter 3 has also

Table 5.5: *heurVNShet* (5 runs)

	$\rho = 0$						$\rho = 100$					
	avg.	%	best	%	CPU[a]	waiting	avg.	%	best	%	CPU[a]	waiting
U												
a2-16	294.25	0.00	294.25	0.00	68.20	0.57	300.17	0.00	300.17	0.00	70.20	0.00
a2-20	344.83	0.00	344.83	0.00	133.80	13.25	345.74	0.00	345.74	0.00	128.80	0.00
a2-24	431.12	0.00	431.12	0.00	187.80	22.21	448.07	0.00	448.07	0.00	202.40	0.00
a3-18	300.48	0.00	300.48	0.00	45.40	12.19	316.78	0.00	316.78	0.00	47.40	0.00
a3-24	345.26	0.12	344.83	0.00	86.80	14.47	347.37	0.11	347.37	0.11	86.60	0.00
a3-30	495.11	0.05	494.85	0.00	105.60	15.67	502.46	0.16	501.68	0.00	111.20	0.00
a3-36	583.89	0.12	583.30	0.02	162.60	55.06	602.23	0.62	599.02	0.08	175.20	0.00
a4-16	282.68	0.00	282.68	0.00	26.00	0.00	282.68	0.00	282.68	0.00	26.80	0.00
a4-24	375.04	0.00	375.02	0.00	50.80	30.90	386.92	0.14	386.38	0.00	51.20	0.00
a4-32	488.27	0.57	486.88	0.28	86.00	11.79	496.05	0.59	493.15	0.00	84.60	0.00
a4-40	565.58	1.41	561.80	0.74	130.60	0.75	563.87	1.06	559.45	0.27	130.00	0.00
a4-48	680.98		673.64		253.80	50.57	717.58		711.12		0.00	0.00
$\overline{U}$	432.29		431.14		111.45	18.95	442.49		440.97		92.87	0.00
E												
a2-16	331.16	0.00	331.16	0.00	65.60	0.00	331.16	0.00	331.16	0.00	67.20	0.00
a2-20	347.03	0.00	347.03	0.00	120.00	13.25	347.89	0.00	347.89	0.00	111.80	0.00
a2-24	450.25	0.00	450.25	0.00	160.40	16.03	462.57	0.14	461.93	0.00	172.60	0.00
a3-18	300.63	0.00	300.63	0.00	47.60	9.27	316.78	0.00	316.78	0.00	49.00	0.00
a3-24	345.59	0.20	344.91	0.00	76.20	14.47	347.37	0.09	347.05	0.00	80.60	0.00
a3-30	501.41	0.17	500.58	0.00	107.60	7.24	503.70	0.40	501.68	0.00	106.40	0.00
a3-36	583.79	0.10	583.19	0.00	161.60	58.43	607.10	0.46	605.46	0.18	161.80	0.00
a4-16	285.99	0.00	285.99	0.00	25.00	20.00	291.55	0.00	291.55	0.00	24.40	0.00
a4-24	384.03	0.05	383.84	0.00	52.60	10.88	386.64	0.07	386.38	0.00	51.00	0.00
a4-32	504.79		502.52		83.00	15.88	509.76		507.72		81.00	0.00
a4-40	588.40		585.64		121.00	12.94	595.81		590.19		121.00	0.00
a4-48	681.80		675.37		252.20	49.48	717.95		715.62		285.00	0.00
$\overline{E}$	442.07		440.93		106.07	18.99	451.52		450.28		109.32	0.00
I												
a2-16	294.25	0.00	294.25	0.00	68.40	0.57	300.36	0.06	300.17	0.00	71.40	0.00
a2-20	355.74	0.00	355.74	0.00	141.80	13.25	356.64	0.00	356.64	0.00	137.40	0.00
a2-24	431.12	0.00	431.12	0.00	211.00	22.21	450.37	0.00	450.37	0.00	207.20	0.00
a3-18	302.17	0.00	302.17	0.00	47.20	12.19	318.62	0.05	318.47	0.00	50.20	0.00
a3-24	344.99	0.05	344.83	0.00	83.60	14.47	347.55	0.17	347.37	0.11	87.20	0.00
a3-30	495.13	0.06	494.85	0.00	106.80	15.67	501.68	0.00	501.68	0.00	111.60	0.00
a3-36	619.64		618.58		170.60	43.11	630.61		627.39		207.80	0.00
a4-16	299.05	0.00	299.05	0.00	27.00	20.00	302.06	0.08	301.81	0.00	26.40	0.00
a4-24	376.19	0.31	375.07	0.01	51.60	26.81	386.89	0.13	386.38	0.00	51.40	0.00
a4-32	488.64	0.35	486.93	0.00	88.00	10.08	498.99		497.07		84.80	0.02
a4-40	563.34	1.01	561.35	0.66	132.20	2.36	563.07	0.65	561.35	0.34	132.20	0.00
a4-48	687.44		680.43		262.40	47.17	717.98		713.21		303.60	0.00
$\overline{U}$	438.14		437.03		115.88	18.99	447.90		446.83		122.60	0.00
$\overline{UEI}$	437.50		436.37		111.13	18.98	447.31		446.03		108.26	0.00

[a] run times in seconds

been adapted to this problem version. High quality solutions are computed within short computation times. These results suggest that the application of the proposed method is also suitable for larger (real world) instances. Furthermore, also the impact of penalizing vehicle waiting time with passengers aboard has been investigated. Heuristic results for the two extreme settings have been compared. On the one hand, only considering the cost part in the objective function, on the other hand, setting the penalization term to a rather high value. The latter setting results in solutions without user waiting time when aboard

a vehicle. When comparing the results, it can be observed that the latter setting yields average routing costs that are only 2.5% higher than in the minimum cost setting. The maximum increase amounts to about 6%. This indicates that from a company perspective, avoiding waiting time with users aboard a vehicle does not lead to a significant cost increase.

The next and final step is to integrate, in addition to heterogeneous users and vehicles, the assignment of drivers and additional personnel to the different vehicles.

6 Solving the real world problem

6.1 Introduction

The dial-a-ride problem variant considered in this chapter represents the final step towards reality. It tries to model the static real world problem situation, as, e.g., faced by the ARC, in the field of patient transportation. The ARC has to devise routing plans for their ambulances for all those requests that are known ahead of time. We define the according problem as the HDARP with driver related constraints (dHDARP). The heterogeneous nature of the problem has already been investigated in Chapter 5. In the current chapter we try to cover all real world characteristics. These are, as before, heterogeneous passengers and vehicles and, an aspect not treated so far, the assignment of drivers and additional personnel to the different vehicles.

We thus consider two different types of vehicles, three patient types (seated, on a stretcher, in a wheelchair), and accompanying persons. As in Chapter 5, up-grading conditions apply for accompanying persons and seated patients. Users specify time windows for either the pickup or the drop off location; and in some cases a second staff member has to be aboard the vehicle.

Eventually, the ARC has to assign drivers to vehicles (usually there are fewer drivers than vehicles available) and driver working regulations have to be integrated into the planning process. These regulations refer to maximum shift lengths and mandatory breaks. The aim is to construct a routing plan that is of minimum routing cost while respecting service related criteria, expressed in terms of time windows, as well as labor regulations.

We address this complex problem situation with a column generation algorithm that computes lower bounds. These bounds are used to asses the quality of the developed heuristic solution method, which is again based on VNS, integrating the findings of the previous chapters.

The contribution of this chapter is fourfold. First, we devise a problem formulation for the dHDARP. It simultaneously takes routing and driver deployment decisions into account. This increased integration of two planning decisions is possible since the time frame within which regular patient transports are carried out corresponds to the drivers' working shifts. Second, a column generation framework is designed. The central new aspect lies in the way the maximum route duration limit together with a non-empty time window at the start depot is treated in the dynamic programming part. Third, an efficient tailor-made metaheuristic

algorithm for the specified problem is presented. And last but not least, the integration of the two methods into a collaborative scheme is investigated. The current chapter has also been summarized in form of a technical report (Parragh et al., 2009a). Many of the following passages are therefore either similar or identical to those of the technical report.

6.2 Related Work

In a related research domain, a recent work by Xu et al. (2003) deals with a practical pickup and delivery problem. They take multiple vehicle types, multiple time windows and several compatibility constraints into account. These compatibility constraints refer to orders that can only be handled by certain vehicle types and those that cannot be shipped together. In addition, first-in-first-out loading rules and driver working regulations are considered. This complex problem situation is successfully solved by a heuristic column generation algorithm; instead of an exact pricing procedure several heuristics are employed. In their case, the solution obtained at the root node is quite often already integer. If this is not the case, a MIP is solved on the set of obtained columns. The developed method yielded high quality solutions for the described problem. Column generation integrated into a branch and cut framework has also been successfully applied to the standard PDPTW in Ropke and Cordeau (2008), outperforming an earlier branch and cut algorithm by Ropke et al. (2007), in terms of the maximum problem size that can be handled. These results indicate that the use of a column generation based algorithm is a promising venue to follow. The combination of column generation and a local search derived method has only recently been successfully implemented by Danna and Lepape (2005), yielding a so-called collaborative scheme. This chapter is based on these findings.

6.3 Problem formulation

In the following the basic notation needed to formulate the dHDARP is given. Thereafter, two different problem formulations are presented; a 3-index based model and a more compact set partitioning type formulation. The latter will serve as the basis for the proposed column generation framework.

6.3.1 Notation

As all other problem versions considered so far, the dHDARP is modeled on a complete directed graph $G = (V, A)$. For each arc (i, j) a non-negative travel cost c_{ij} and a non-negative travel time t_{ij} is considered. A total amount of n customer requests, each consisting of a pickup and delivery vertex pair $\{i, n + i\}$ have to be served. The set of pickup vertices is given by $P = \{1, \ldots, n\}$, the set of delivery vertices by $D = \{n + 1, \ldots, 2n\}$. At every

pickup vertex one patient waits to be transported. This patient may demand one of three different modes of transportation. Passengers may have to be transported seated ($q_i^1 = 1$), on a stretcher ($q_i^2 = 1$), or in a wheelchair ($q_i^3 = 1$). Each patient may be accompanied by a friend, relative or nurse ($q_i^0 = 1$). The demand at every delivery vertex is equal to $q_{n+i}^s = -q_i^s$ for all $s \in R = \{0, 1, 2, 3\}$. In addition, a patient may need additional personnel, besides the driver, on the vehicle (in our case a civilian servant). If this is the case $a_i = 1$ and 0 otherwise. Every user either specifies a time window $[e_i, l_i]$ for the pickup (origin) or the drop off (destination) location and beginning of service has to start within this time window. Maximum passenger ride times $\bar{L}_i$ are also considered, in order to keep quality of service at a reasonably high level. This is done by artificially constructing a time window at the origin (destination) relative to the time window given at the corresponding destination (origin). At each vertex loading or unloading operations last for a given service time d_i.

A set K of m heterogeneous vehicles has to serve all n transportation requests. Each vehicle $k \in K$ is associated with a vector $C^{s,k}$ that gives the amount of resource s available on vehicle k. As mentioned in the previous chapter, the ARC disposes of two basic vehicle types. Type 1 (T1) provides 1 staff seat, 6 patient seats, and 1 wheelchair place. Type 2 (T2) provides 2 staff seats, 1 patient seat, 1 stretcher, and 1 wheelchair place. Patients demanding to be transported seated may use a patient seat or the stretcher. Patients demanding a stretcher can only be transported on a stretcher. The same applies to wheelchair passengers. Accompanying persons, however, may use a staff seat, a patient seat or the stretcher, if no other transportation mode is available. Each route has to start at the start depot 0 within a prespecified time window and end at the end depot $2n + 1$, respecting a route duration limit T. This limit is based on Austrian labor regulations. Driver working shifts are limited to 8.5 hours per day including a (lunch) break of $H = 30$ minutes that has to start within a given time window $[e_H, l_H]$. The lunch break can be held at every vertex. In addition, only a certain number of drivers m_d (usually $m_d < m$) and only a limited number of additional personnel (civilian servants, i.e. employees serving their alternative service) m_c are available (We assume here that a civilian servant can be either male or female; although in Austria only the male part of the population has to either do military or alternative service.) A civilian servant can only work during morning or afternoon periods on a vehicle. If a civilian servant is needed on the vehicle in the morning, he/she has to be returned to the depot at noon $2n+2$ within a certain time window $[e_{2n+2}, l_{2n+2}]$. If a civilian servant is on the vehicle in the afternoon he/she has to be picked up at the noon depot within the time window. In case there are more drivers available than actually needed to serve all requests, the excess drivers may be employed for civilian servant duties, serving only half of the day on a vehicle.

Thus, the set of all vertices is given by $V = P \cup D \cup \{0, 2n + 1, 2n + 2\}$, and the set of all arcs by $A = \{(i, j) : i \in V \setminus \{2n + 1\}, j \in V \setminus \{0\}, i \neq j\}$.

During the optimization process the following decision variables are determined:

$$x_{ij}^k = \begin{cases} 1, & \text{if arc } (i,j) \text{ is traversed by vehicle } k, \\ 0, & \text{else,} \end{cases}$$

$$z_0^k = \begin{cases} 1, & \text{if a driver is assigned to vehicle } k, \\ 0, & \text{else,} \end{cases}$$

$$z_1^k = \begin{cases} 1, & \text{if a civilian servant is assigned to vehicle } k \text{ in the morning,} \\ 0, & \text{else,} \end{cases}$$

$$z_2^k = \begin{cases} 1, & \text{if a civilian servant is assigned to vehicle } k \text{ in the afternoon,} \\ 0, & \text{else,} \end{cases}$$

$$v_i^k = \begin{cases} 1, & \text{if the lunch break is held at vertex } i, \\ 0, & \text{else,} \end{cases}$$

$w \in \{0, \ldots, m_d\}$ number of drivers that serve as additional civilian servants,

$B_i^k \ldots$ beginning of service of vehicle k at vertex i,
$W_H^k \ldots$ waiting time until lunch break on vehicle k,
$Q_i^{s,k} \ldots$ load of vehicle k of resource s when leaving vertex i.

6.3.2 A 3-index formulation

The aim of the ARC is the minimization of total routing costs,

$$\min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k, \tag{6.1}$$

subject to several constraints that reflect the above stated real world conditions. Those given below guarantee that each request is served exactly once (6.2) and that each origin-destination pair is visited by the same vehicle (6.3). Flow conservation is taken care of by equalities (6.4). The subsequent equalities (6.5) and (6.6) ensure that, if a driver is assigned to a vehicle, the vehicle starts at and returns to the depot at the end of its route,

$$\sum_{k \in K} \sum_{j \in P \cup D} x_{ij}^k = 1 \qquad \forall i \in P, \tag{6.2}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{n+i,j}^k = 0 \qquad \forall i \in P, k \in K, \tag{6.3}$$

$$\sum_{i \in V} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall j \in P \cup D \cup \{2n+2\}, k \in K, \tag{6.4}$$

$$\sum_{j \in V} x_{0j}^k = z_0^k \qquad \forall k \in K, \tag{6.5}$$

$$\sum_{i \in V} x_{i,2n+1}^k = z_0^k \qquad \forall k \in K, \tag{6.6}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in V, k \in K, \tag{6.7}$$

$$z_0^k \in \{0,1\} \qquad \forall k \in K. \tag{6.8}$$

Civilian servant related conditions form another block of constraints. Equalities (6.9) make sure that the noon depot is used if a civilian servant is assigned to the respective vehicle for morning or afternoon periods. This is necessary to either pick up or drop off the civilian servant at the beginning or at the end of his/her shift. If a civilian servant is assigned to a vehicle he/she demands a staff seat. This is modeled in inequalities (6.10) – (6.12). Furthermore, a user demanding a civilian servant aboard the vehicle can only be visited if a civilian servant is on the vehicle, see constraints (6.13).

$$\sum_{i \in V} x_{i,2n+2}^k = \max\left\{z_1^k, z_2^k\right\} \qquad \forall k \in K, \tag{6.9}$$

$$Q_0^{0,k} \geq z_1^k \qquad \forall k \in K, \tag{6.10}$$

$$x_{i,2n+2}^k = 1 \;\Rightarrow\; Q_{2n+2}^{0,k} \geq Q_i^{0,k} - z_1^k + z_2^k \qquad \forall k \in K, \tag{6.11}$$

$$x_{i,2n+2}^k = 1 \;\Rightarrow\; Q_{2n+2}^{s,k} \geq Q_i^{r,k} \qquad \forall k \in K, s \in R \setminus \{0\}, \tag{6.12}$$

$$z_1^k + z_2^k \geq a_i \sum_{j \in V} x_{ij}^k \qquad \forall i \in V, k \in K, \tag{6.13}$$

$$z_1^k, z_2^k \in \{0,1\} \qquad \forall k \in K. \tag{6.14}$$

Consistency with respect to resource and load variables is guaranteed by constraints (6.15)–(6.18). Inequalities (6.15) ensure load propagation from one vertex to the other. Up-grading constraints for resources 0, 1, and 2 are given in (6.16). They guarantee that capacity restrictions regarding these resources are not violated and that each patient demanding resource 0, 1 or 2 can only be loaded if there is either enough capacity of the resource demanded or another one with a higher number (0 = staff seat, 1 = patient seat, 2 = stretcher).
Suppose an empty T2 vehicle (2 staff seats, 1 patient seat, 1 stretcher, 1 wheelchair place) visits two origin locations in a row. At each location a seated patient with an accompanying person has to be picked up. The first seated passenger will fill the patient seat, the accompanying person the first staff seat. If it was not possible to have seated patients sit on the stretcher, the second origin location could not be visited. However, the given up-grading conditions allow seated passengers to sit on the stretcher. Therefore, origin location two can be visited. The seated passenger will use the stretcher and the accompanying person the second staff seat. Suppose the same two origin locations were visited by an empty T1 vehicle (1 staff seat, 6 patient seats, 1 wheelchair place). In this case the first seated patient would again use a patient seat and the accompanying person the staff seat. At the second location there is no more empty staff seat available. However, again due to the given up-grading possibilities, the accompanying person can use a patient seat. Thus, after visiting location two, three patient seats and the staff seat will be occupied.

Finally, constraints (6.17) guarantee that patients demanding resource 3 (wheelchair place) can only be transported if there is enough capacity of resource 3:

$$x_{ij}^k = 1 \Rightarrow \quad Q_j^{s,k} \geq Q_i^{s,k} + q_j^s \qquad \forall i,j \in V \setminus \{2n+2\}, k \in K, s \in R, \tag{6.15}$$

$$\sum_{s'=s}^{2} Q_i^{s',k} \leq \sum_{s'=s}^{2} C^{s',k} \qquad \forall i \in V \setminus \{2n+2\}, k \in K, s \in R \setminus \{3\}, \tag{6.16}$$

$$Q_i^{3,k} \leq C^{3,k} \qquad \forall i \in V, k \in K, \tag{6.17}$$

$$Q_i^{s,k} \geq 0 \qquad \forall i \in V, k \in K, s \in R. \tag{6.18}$$

Inequalities (6.19) and (6.20) define the beginning of service for each vertex. If vertex i is chosen for the lunch break ($v_i = 1$), in addition to the service time associated with this vertex, the vehicle will stay at this vertex until the lunch break time window starts (W_H^k) and then until the lunch break has been concluded. These constraints also take care of subtour elimination given that $(t_{ij} + d_i) > 0$ for all $i, j \in V, i \neq j$. Total route duration is limited by (6.21). If a driver is assigned to a vehicle he/she has to stop for a lunch break. This is imposed by constraints (6.22).

$$x_{ij}^k - v_i^k = 1 \Rightarrow \quad B_j^k \geq B_i^k + d_i + t_{ij} \qquad \forall i,j \in V, k \in K, \tag{6.19}$$

$$x_{ij}^k + v_i^k = 2 \Rightarrow \quad B_j^k \geq B_i^k + d_i + H + W_H^k + t_{ij} \qquad \forall i,j \in V, k \in K, \tag{6.20}$$

$$B_{2n+2}^k - B_0^k \leq T \qquad \forall k \in K, \tag{6.21}$$

$$\sum_{i \in V} v_i^k \geq z_0^k \qquad \forall k \in K, \tag{6.22}$$

$$v_i^k \in \{0,1\} \qquad \forall i \in V, k \in K. \tag{6.23}$$

The different limits on the beginning of service are given in the following. Standard time window constraints are modeled in (6.24). Additional time related constraints provide new bounds on the beginning of service at those vertices where a civilian servant is required. Bounds for morning periods are given in (6.25) and for afternoon periods in (6.26). In case a civilian servant is present during both periods, no additional bounds are needed. Furthermore, constraints (6.27) guarantee that the lunch break starts within the lunch break time window $[e_H, l_H]$.

$$e_i \leq B_i^k \leq l_i \qquad \forall i \in V, k \in K, \tag{6.24}$$

$$(z_1^k - z_2^k)a_i \geq 1 \Rightarrow \quad B_0^k \leq B_i^k \leq B_{2n+1}^k \qquad \forall i \in V, k \in K, \tag{6.25}$$

$$(z_2^k - z_1^k)a_i \geq 1 \Rightarrow \quad B_{2n+1}^k \leq B_i^k \leq B_{2n+2}^k \qquad \forall i \in V, k \in K, \tag{6.26}$$

$$v_i^k = 1 \Rightarrow \quad e_H \leq B_i^k + d_i + W_H^k \leq l_H \qquad \forall i \in V, k \in K, \tag{6.27}$$

$$W_H^k \geq 0 \qquad \forall k \in K. \tag{6.28}$$

Finally, inequalities (6.29) and (6.30) limit the number of drivers and civilian servants that can be assigned to vehicles. Each driver who is used for civilian servant duties, instead of driving, can be employed for either morning or afternoon periods; increasing the number of civilian servants available and decreasing the number of drivers by the same amount (w):

$$\sum_{k \in K} z_0^k \leq m_d - w, \tag{6.29}$$

$$\sum_{k \in K} z_1^k + \sum_{k \in K} z_2^k \leq m_c + w, \tag{6.30}$$

$$w \in \{0, \ldots, m_d\}. \tag{6.31}$$

6.3.3 A set partitioning formulation

The above stated problem can be reformulated in a more compact way. Let Ω be the set of all feasible routes. Then, let $\mathcal{T}$ denote the set of different vehicle types and Ω_t the set of feasible routes of vehicle type t, $\Omega = \bigcup_t \Omega_t$. Furthermore, let m_t denote the number of vehicles of type t available; and let c_r be the cost of route $r \in \Omega$. The constants b_{ir} and g_r give the number of times vertex $i \in P$ is traversed by r and the number of civilian servants needed by route r, respectively. Finally, variables u_r evaluate to one if route r is used in the solution. The dHDARP can thus be formulated as the following Set Partitioning problem (SP):

$$\min \sum_{r \in \Omega} c_r u_r \tag{6.32}$$

subject to

$$\sum_{r \in \Omega} b_{ir} u_r = 1 \qquad \forall i \in P, \tag{6.33}$$

$$\sum_{r \in \Omega_t} u_r \leq m_t \qquad \forall t \in \mathcal{T}, \tag{6.34}$$

$$\sum_{r \in \Omega} u_r \leq m_d - w, \tag{6.35}$$

$$\sum_{r \in \Omega} g_r u_r \leq m_c + w, \tag{6.36}$$

$$w \geq 0, \tag{6.37}$$

$$u_r \in \{0, 1\} \qquad \forall r \in \Omega. \tag{6.38}$$

The objective function (6.32) minimizes the costs of the selected routes. Constraints (6.33) guarantee that every request is served exactly once. Inequalities (6.34) limit the number of vehicles of type t that can be used in the solution; (6.35) ensures that at most as many

vehicles as there are drivers available are used. Drivers that are not needed in a solution may be employed as additional civilian servants, thus increasing the number of civilian servants that can be used by w, see constraint (6.36). In order to compute a lower bound, the Linear relaxation of SP (LSP) is solved. LSP is obtained by replacing (6.38) by

$$u_r \geq 0 \qquad \forall r \in \Omega. \tag{6.39}$$

Due to the large size of Ω the above formulation will not be solved directly. Instead, a restricted version of this problem, considering only a small subset of columns $\Omega' \subset \Omega$, will be solved. Ω' is generated by solving LSP using column generation. In column generation LSP decomposes into a (restricted) master problem and $|\mathcal{T}|$ subproblems, one for each vehicle type. Let π_i, σ_t, λ, and ϕ be the dual variable values of constraint (6.33) for index i, of constraint (6.34) for index t, of constraint (6.35), and of constraint (6.36), respectively. The reduced cost of column u_r corresponding to trip $r \in \Omega$ is given by,

$$\bar{c}_r = c_r - \sum_{i \in r} \pi_i - \sigma_t - \lambda - g_r \phi. \tag{6.40}$$

Subproblem t corresponds to finding a single vehicle trip $r \in \Omega_t$ for a vehicle of type t such that its reduced cost $\bar{c}_r$ is minimum. It is subject to constraints (6.3)–(6.27), omitting superscript k, setting $z_0 = 1$, and replacing k by t in case of the capacity limits $C^{s,t}$. The index t refers to the vehicle type.

6.4 Solving the column generation subproblem

In order to find negative reduced cost columns we implemented a label setting shortest path algorithm and several heuristics. The use of heuristics, aiding the exact procedure in finding negative reduced cost paths, yields significant run time reductions (see Savelsbergh and Sol, 1998). Following the findings of Ropke and Cordeau (2008) our label setting algorithm only considers elementary paths, i.e. every vertex can only be covered exactly once in a path. This demands additional information to be stored at every label but yielded slightly better results than its non-elementary version as shown by Ropke and Cordeau (2008) for the PDPTW. The PDPTW is a special case of the dHDARP considered in this paper. In the following the elementary shortest path algorithm will be described in detail. Thereafter, the heuristics based on the label setting algorithm and the different other heuristic algorithms employed are briefly discussed.

6.4.1 The label setting algorithm

The labeling algorithm implemented is based on the one described by Ropke and Cordeau (2008) to solve the elementary shortest path problem with time windows, capacity, and

Algorithm 6.1 The labeling algorithm (source = 0, sink = $2n + 1$)

```
1: // initialization
2: get graph G = (V, A);
3: generate initial label at source node κ(0);
4: set list of unprocessed labels Γ := {κ(0)};
5: repeat
6:     κ := pop_first_label(Γ);
7:     set i := η(κ);
8:     // dominance
9:     if no label ∈ L_i dominates κ then
10:        L_i := L_i ∪ {κ};
11:        // extension
12:        for each arc (i, j) ∈ G leaving i do
13:            extend κ yielding κ' at vertex j;
14:            // elimination
15:            if κ' should not be eliminated then
16:                Γ := Γ ∪ κ';
17:            end if
18:        end for
19:    end if
20: until Γ = ∅
21: return path associated with best label ∈ L_{2n+1}
```

pickup and delivery. The subproblem we have to solve is also a constrained shortest path problem. It can be described as an elementary shortest path problem with time windows, heterogeneous capacity, pickup and delivery, and route duration, given its additional complexity due to the different modes of transportation available and the maximum route duration limit.

In Algorithm 6.1 the structure of our labeling algorithm to solve this shortest path problem is given. The shortest path between source (origin depot) and sink (destination depot) has to be found. In an *initialization* step, a first label at the start depot $\kappa(0)$ is generated (see Section 6.4.1.2). $\kappa(0)$ is put into the queue of unprocessed labels Γ (all labels in the queue are sorted according to ascending reduced costs). Thereafter we enter the repeat-until loop. In every iteration, the first label in Γ is taken from the queue. If it is not *dominated* by any other label at the same vertex (see Section 6.4.1.3), it is added to the set of labels at vertex i which is denoted as $\mathcal{L}_i$, and its *extension* is tried along each arc $(i, j) \in G$ leaving vertex i (see Section 6.4.1.2). In case the resulting label κ' is not eliminated in the subsequent label *elimination* check (see Section 6.4.1.5) it is added to Γ. This is repeated until Γ is empty. In this case the path corresponding to the best label $\in \mathcal{L}_{2n+1}$ is returned.

Wihtin a column generation framework it is usually not necessary to identify the best path or column. Thus, the run time of the labeling algorithm can be reduced, if it is stopped as soon as a certain number of negative reduced cost labels have been found at $2n + 1$.

Furthermore, it can be beneficial if, in addition to the path corresponding to the best label at vertex $2n+1$, all negative reduced costs paths ending at $2n+1$ are returned. We briefly come back to this issue in Section 6.4.2.1.

6.4.1.1 Lunch break and civilian servant requirements

In order to properly treat the lunch break requirement and the possibility of additional personnel on the vehicle (civilian servants) in the labeling algorithm, three additional artificial nodes are introduced into graph G: one, denoted by $2n+3$, for the morning civilian servant, one, denoted by $2n+4$, for the afternoon civilian servant, and one for the lunch stop, denoted by $2n+5$. If a path generated by means of dynamic programming contains the morning civilian servant node, a civilian servant is aboard the vehicle during the morning shift. If the afternoon civilian servant node is part of the path, a civilian servant is assigned to the vehicle in the afternoon. Every path has to contain the lunch node, with one exception: if the vehicle returns to the end depot before the end of the lunch time window and the lunch has not taken place yet, it is assumed, that it is held at the end depot. The vertex that is followed by the lunch node on the constructed graph is the one where the lunch break will be held.

In the graph $2n+3$ can only be visited from the origin depot 0 and $2n+4$ only from the noon depot $2n+2$. Travel times for these nodes are set to $t_{0,2n+3} = t_{2n+2,2n+4} = 0$, $t_{2n+3,j} = t_{0,j}$ and $t_{2n+4,j} = t_{2n+2,j}$ for all j. In case of the lunch node, travel times from all vertices to this vertex are set to $t_{i,2n+5} = 0$ for all i. Let now i_{2n+5} denote the node visited directly before the lunch node, the travel times from the lunch node are dynamically set to $t_{2n+5,j} = t_{i_{2n+5},j}$. The time windows of the civilian servant pickup nodes are set to the beginning and the end of the planning horizon, respectively. In case of the lunch node a time window is given, i.e. $e_{2n+5} = e_H$ and $l_{2n+5} = l_H$ (see above). The service times at the civilian servant nodes are set to $d_{2n+3} = d_{2n+4} = 0$ and to $d_{2n+5} = H$ in case of the lunch node. The load is set to $q^0_{2n+3} = q^0_{2n+4} = 1$ at the civilian servant nodes and to zero for all other resources of artificial nodes.

6.4.1.2 Label management

For each label the following data is stored: η - the vertex of the label, δ - the departure time at η, Q^s_{cum} - the cumulative load of resource s when leaving η, c_{cum} - the accumulated cost until η, $b \in \{0,1\}$ - whether a lunch stop has already been made, $\alpha \in \{0,1\}$ - whether a civilian servant is aboard the vehicle or not, $o \in \{0,1\}$ - whether the noon depot has already been visited or not, $\mathcal{V} \subseteq \{0,\ldots,2n+5\}$ - the set of vertices visited along the path, $\mathcal{O} \subseteq \{1,\ldots,n\}$ - the set of open requests, f - the forward time slack, w_{cum} - the accumulated waiting time, and a pointer to its parent label. The resources f and w_{cum} are needed to check whether the route duration limit can still be respected; f gives the maximum time

the departure at the noon depot can be shifted forward in time. Section 6.4.1.4 explains these resources in further detail.

Extension of a label κ along an arc $(\eta(\kappa), j)$ is only possible if the following holds,

$$\delta(\kappa) + t_{\eta(\kappa),j} \leq l_j, \tag{6.41}$$

$$Q^s_{cum}(\kappa) + q^s_j + \sum_{s'=s+1}^{2} q^{s'}_j \leq \hat{C}^{s,t} \ \forall s \in R, \tag{6.42}$$

$$\alpha(\kappa) \geq a_j, \tag{6.43}$$

$$(1 - b(\kappa))(\max\left\{\delta(\kappa) + t_{\eta(\kappa),j}, e_j\right\} + d_j) \leq l_H, \tag{6.44}$$

$$\max\left\{\max(\delta(\kappa) + t_{\eta(\kappa),j}, e_j) + d_j, (1 - b(\kappa))e_H\right\} + (1 - b(\kappa))H - e_0 - F^0_j \leq T, \tag{6.45}$$

$$j \notin \mathcal{V}(\kappa). \tag{6.46}$$

The matrix $\hat{C}^{s,t} = C^{s,t} + \sum_{s'=s+1}^{2} C^{s',t}$. The final time slack F^0_j at vertex j is given by $F^0_j = \min\left\{\min[f(\kappa), l_j - (\delta(\kappa) + t_{\eta(\kappa),j}) + w_{cum}(\kappa)], w_{cum}(\kappa) + \max(0, e_j - (\delta(\kappa) + t_{\eta(\kappa),j})\right\}$, i.e. the minimum over all forward time slacks and the total waiting time until vertex j (see Section 6.4.1.4 for further details). Condition (6.41) ensures time window feasibility - the departure from the previous vertex $\eta(\kappa)$ plus the travel time from $\eta(\kappa)$ to j has to be earlier or at most equal to the later time window at vertex j. According to condition (6.42) a path can only be feasibly extended along arc $(\eta(\kappa), j)$ if all loading restrictions are satisfied. Condition (6.43) states that if vertex j demands a civilian servant aboard the vehicle ($a_j = 1$), it can only be visited if a civilian servant is currently on the vehicle ($\alpha(\kappa) = 1$). In case the lunch node is not part of the path yet, extension along arc $(\eta(\kappa), j)$ is only possible if it can still be feasibly inserted after vertex j. This is taken care of by (6.44). Feasibility with respect to route duration is guaranteed by condition (6.45). For further details on this issue we refer to Section 6.4.1.4. Finally, the elementary path condition is ensured by (6.46). Moreover, κ and j may have to comply with one of the following conditions,

$$j \in D \wedge j - n \in \mathcal{O}(\kappa), \tag{6.47}$$

$$j = 2n + 1 \wedge \mathcal{O}(\kappa) = \emptyset \wedge \alpha - o \leq 0. \tag{6.48}$$

Condition (6.47) ensures that if j is a delivery, it can only be visited if the request is open, i.e. the corresponding pickup has already been visited. Condition (6.48) ensures that a label can only be extended to the end depot if there are no more open requests and if the noon depot was visited, in case a civilian servant is currently on the vehicle.

If a label can feasibly be extended along arc $(\eta(\kappa), j)$, a new label κ' is generated at

vertex j:

$$\eta(\kappa') = j, \tag{6.49}$$

$$\delta(\kappa') = \max\left\{\delta(\kappa) + t_{\eta(\kappa),j}, e_j\right\} + d_j, \tag{6.50}$$

$$Q^0_{cum}(\kappa') = \begin{cases} Q^0_{cum}(\kappa) - 1 & \text{if } j = 2n+2 \wedge \alpha(\kappa) = 1, \\ Q^0_{cum}(\kappa) + q^0_j + \sum_{s'=1}^{2} q^{s'}_j & \text{else,} \end{cases} \tag{6.51}$$

$$Q^s_{cum}(\kappa') = Q^s_{cum}(\kappa) + q^s_j + \sum_{s'=s+1}^{2} q^{s'}_j \quad \forall s \in R \setminus \{0\}, \tag{6.52}$$

$$c_{cum}(\kappa') = c_{cum}(\kappa) + \bar{c}_{\eta(\kappa),j}, \tag{6.53}$$

$$b(\kappa') = \begin{cases} 1 & \text{if } j = 2n+5, \\ b(\kappa) & \text{else,} \end{cases} \tag{6.54}$$

$$\alpha(\kappa') = \begin{cases} 1 & \text{if } j \in \{2n+3, 2n+4\}, \\ 0 & \text{if } j = 2n+2, \\ \alpha(\kappa) & \text{else,} \end{cases} \tag{6.55}$$

$$o(\kappa') = \begin{cases} 1 & \text{if } j = 2n+2, \\ o(\kappa) & \text{else,} \end{cases} \tag{6.56}$$

$$\mathcal{V}(\kappa') = \mathcal{V}(\kappa) \cup \{j\}, \tag{6.57}$$

$$\mathcal{O}(\kappa') = \begin{cases} \mathcal{O}(\kappa) \cup \{j\} & \text{if } j \in P, \\ \mathcal{O}(\kappa) \setminus \{j-n\} & \text{if } j \in D, \\ \mathcal{O}(\kappa) & \text{else,} \end{cases} \tag{6.58}$$

$$w_{cum}(\kappa') = w_{cum}(\kappa) + \max\left\{0, e_j - (\delta(\kappa) + t_{\eta(\kappa),j})\right\} \tag{6.59}$$

$$f(\kappa') = \min\left\{f(\kappa), w_{cum}(\kappa) + l_j - (\delta(\kappa) + t_{\eta(\kappa),j})\right\}. \tag{6.60}$$

At the origin depot (the first vertex along each path) these labels are initialized by setting $\delta(0) = e_0$, $Q^s_{cum}(0) = 0$ for all $s \in R$, $c_{cum} = 0$, $b(0) = 0$, $\alpha(0) = 0$, $o = 0$, $f(0) = l_0 - e_0$, and $w_{cum}(0) = 0$.

6.4.1.3 Dominance

For dominance a criterion similar to the one denoted as DOM1' in Ropke and Cordeau (2008) is used. Let $\mathcal{U}(\kappa)$ denote the set of unreachable requests of label κ, $\hat{\mathcal{U}} = \mathcal{V} \cup \left\{i \in P : \delta(\kappa) + t_{\eta(\kappa),i} > l_i\right\}$, according to this criterion a label κ dominates another label κ' if

$$\eta(\kappa) = \eta(\kappa'), \delta(\kappa) \leq \delta(\kappa'), c_{cum}(\kappa) \leq c_{cum}(\kappa'), \hat{\mathcal{U}}(\kappa) \subseteq \hat{\mathcal{U}}(\kappa'), \mathcal{O}(\kappa) \subseteq \mathcal{O}(\kappa'). \tag{6.61}$$

In addition, in our case, also the following has to hold,

$$\min\{f(\kappa), w_{cum}(\kappa)\} \geq \min\{f(\kappa'), w_{cum}(\kappa')\}, \tag{6.62}$$

$$b(\kappa) = b(\kappa'), \alpha(\kappa) = \alpha(\kappa'), o(\kappa) = o(\kappa'). \tag{6.63}$$

This means that the amount of time by which the departure from the origin depot can be shifted forward in time at label κ has to be at least as large as at label κ'. If the lunch node or the noon depot were already visited by the partial path represented by label κ the same has to be true for κ'. The same applies to whether a civilian servant is aboard the vehicle or not. Furthermore we only apply the dominance check to labels with $\eta \in V$.

6.4.1.4 Time windows at start depot and minimum route duration

Incorporating a route duration limit together with a non-empty time window at the origin depot into a label setting algorithm is a non-trivial task. One option, reviewed by Irnich (2007) and introduced by Desaulniers and Villeneuve (2000), consists in appending two resources to each label coupled by a max-term, denoted as $\dot{q}$ and $\dot{z}$. These are extended as follows:

$$\dot{q}(\kappa') = t_{\eta(\kappa),j} + \max\left\{\dot{q}(\kappa) - (t_{\eta(\kappa),j} + d_{\eta(\kappa)}), \dot{z}(\kappa) - l_j\right\} \tag{6.64}$$

$$\dot{z}(\kappa') = t_{\eta(\kappa),j} + \max\left\{\dot{z}(\kappa), \dot{q}(\kappa) - (t_{\eta(\kappa),j} + d_{\eta(\kappa)}) + e_j\right\} \tag{6.65}$$

Here, we would like to show that this is equivalent to using the notion of the forward time slack as developed by Savelsbergh (1992).

The argumentation of Desaulniers and Villeneuve (2000) is the following. Let $F = (0, 1, 2, \ldots, h)$ denote a feasible path, the earliest possible departure time $\tilde{e}_j$ at node j when traveling along U can be calculated as,

$$\tilde{e}_0 = e_0, \tag{6.66}$$

$$\tilde{e}_j = \max\{\tilde{e}_{j-1} + (d_{j-1} + t_{j-1,j}), e_j\} \qquad \forall j \in \{1, \ldots, h\}. \tag{6.67}$$

On the other hand, the latest arrival time $\tilde{l}_j$ at node j for which waiting can be avoided, can be computed by setting,

$$\tilde{l}_0 = l_0, \tag{6.68}$$

$$\tilde{l}_j = \min\left\{\tilde{l}_{j-1} + (d_{j-1} + t_{j-1,j}), l_j\right\} \qquad \forall j \in \{1, \ldots, h\}. \tag{6.69}$$

Let now F be a feasible (partial) path starting at the origin depot 0 and ending at node i. Then, let $s_i = \sum_{(k,l)\in F}(t_{kl} + d_k)$ denote the sum over "pure" travel times along F (including

service time but excluding waiting time), the following two parameters can be defined:

$$\tilde{q}_i = s_i - \tilde{l}_i, \tag{6.70}$$

$$\tilde{z}_i = \max\{s_i, \tilde{q}_i + \tilde{e}_i\}. \tag{6.71}$$

Then the minimum duration of path F (ending at i) is equal to $\max\{\tilde{z}_i, \tilde{q}_i + D_i\}$, D_i denoting the departure time from vertex i.

Now it can be shown that $\tilde{q}_i$ is equivalent to the forward time slack introduced by Savelsbergh (1992). Let $\tilde{f}_i^0$ denote the forward time slack from the origin depot until the end of the path F (here node i). It is the maximum amount of time by which the departure at the depot can be shifted forward without violating any other time window constraint. It is computed as follows (B_j denotes the beginning of service at node j),

$$\tilde{f}_i^0 = \min_{0 \le j \le i} \left\{ l_j - [B_0 + \sum_{p=1}^{j} (d_{p-1} + t_{p-1,p})] \right\}. \tag{6.72}$$

Let us assume that $e_0 = 0$ and therefore $B_0 = 0$, we obtain,

$$\tilde{f}_i^0 = \min_{0 \le j \le i} \left\{ l_j - \sum_{p=1}^{j} (d_{p-1} + t_{p-1,p}) \right\}. \tag{6.73}$$

Let us now rewrite $\tilde{l}_i$ as,

$$\tilde{l}_i = \min_{0 \le j \le i} \left\{ l_j + \sum_{p=j+1}^{i} (d_{p-1} + t_{p-1,p}) \right\}, \tag{6.74}$$

and replace $\tilde{l}_i$ by this formulation in $\tilde{q}_i$, we obtain,

$$\tilde{q}_i = \sum_{p=1}^{i} (d_{p-1} + t_{p-1,p}) - \min_{0 \le j \le i} \left\{ l_j + \sum_{p=j+1}^{i} (d_{p-1} + t_{p-1,p}) \right\} \tag{6.75}$$

$$= - \min_{0 \le j \le i} \left\{ l_j - \sum_{p=1}^{j} (d_{p-1} + t_{p-1,p}) \right\}. \tag{6.76}$$

This shows that $\tilde{q}_i = -\tilde{f}_i^0$. Furthermore, let us examine the minimum route duration time as defined by Desaulniers and Villeneuve (2000) given by $\max\{\tilde{z}_i, \tilde{q}_i + D_i\}$. We still assume that $B_0 = e_0 = 0$. In fact by substituting $\tilde{z}_i$ this term can be rewritten as $\max\{s_i, \tilde{q}_i + \tilde{e}_i, \tilde{q}_i + D_i\} = \max\{s_i, \tilde{q}_i + \max(\tilde{e}_i, D_i)\}$. If D_i is computed correctly it will always be greater than or equal to $\tilde{e}_i$ in our case. Therefore, $\max\{s_i, \tilde{q}_i + D_i\}$ should be valid. Now if we denote by $\tilde{w}_i$ the accumulated waiting time until node i, it is easy to

see that $D_i - \tilde{w}_i = s_i$. Thus, $\max\{s_i, \tilde{q}_i + D_i\} = D_i - \min(-\tilde{q}_i, \tilde{w}_i) = D_i - \min(\tilde{f}_i^0, \tilde{w}_i)$. This corresponds to what Cordeau and Laporte (2003b) use to compute the minimum route duration of a given route (see also Chapter 3).

Thus, to handle a time window at the start depot together with a route duration limit, the notion of forward time slacks as defined by Savelsbergh (1992) can be used. In order to do so, we use the additional resources w_{cum} and f (see above). To generalize what has been shown to the case of $e_0 > 0$, these resources are initialized, as already pointed out, with $f(0) = l_0 - B_0$ ($B_0 = e_0$) and $w_{cum}(0) = 0$.

6.4.1.5 Label elimination

Following the observations of Ropke and Cordeau (2008) and Ropke (2005), labels can be eliminated if the deliveries of open requests cannot be reached in a feasible way. As in Ropke and Cordeau (2008) and Ropke (2005) we consider sets of one and two deliveries and one set of three deliveries. The last set consists of the delivery that is furthest away from the current vertex, the delivery that is furthest away from these two, and the delivery that is furthest away from the current vertex and the two previously selected deliveries. If for one of these sets of deliveries no path can be found that serves all deliveries in the set in a feasible way, the label can be eliminated.

In addition to these sets of deliveries, we check whether there is a civilian servant on the vehicle and whether the noon depot has not been visited yet. If this is the case and the noon depot cannot be reached in a feasible way from the current vertex, the label can equally be eliminated.

6.4.2 Heuristic algorithms

To accelerate the column generation process, besides the exact dynamic programming algorithm, several heuristic procedures to generate reduced cost columns are used. These can be divided into two classes; those that are based on the dynamic programming approach; and those that rely on simple construction/improvement principles.

6.4.2.1 Heuristics based on the labeling algorithm

Two of the heuristic algorithms used to generate columns are based on the exact labeling algorithm. They were both also used by Ropke and Cordeau (2008) in the context of the PDPTW. The first heuristic (*LimLabels*) simply limits the number of labels that can be in the queue of unprocessed labels at a time. At first the limit is set to 500. If no negative reduced cost column can be found with this limit, the limit is increased to 1000. If again no negative reduced cost columns are generated the limit is set to 2000. The second heuristic (*LimGraph*) applies the exact labeling algorithm on a reduced graph. Two reduced graphs are used. In Graph G_5 every pickup and delivery vertex is only connected to the 5 closest

pickup vertices and the 5 closest delivery vertices. In addition, if not already present, connections between each pickup and its corresponding delivery are added and the start depot is connected to all pickup vertices and the morning civilian servant node; and the noon depot is connected to the afternoon civilian servant node. All delivery vertices are connected to the end depot. All vertices except the end depot are connected to the lunch node and the noon depot. The morning civilian servant node, the afternoon civilian servant node and the lunch node are connected to all vertices except the start depot. The second reduced Graph G_{10} is constructed in the same way. Instead of the 5 closest pickups and deliveries it considers the 10 closest.

6.4.2.2 Construction/improvement based heuristics

Four of the heuristics applied use construction and or improvement algorithms. Like the labeling algorithm derived heuristics described above, they are based on those used by Ropke and Cordeau (2008) to solve the PDPTW by means of branch-and-cut-and-price. Heuristic *ConstrHeur* is a simple construction heuristic that starts from every pickup and delivery vertex pair and iteratively adds requests by means of best insertion regarding reduced costs. Heuristic *RandConstrHeur* is also a construction heuristic but here randomized best insertion is used to insert additional requests. The randomization process favors requests that increase the reduced costs of the partial route the least. In both construction heuristics, every time a new request is inserted, the resulting route is checked whether it has negative reduced cost. If this is the case, the according column is generated. After the check, the route undergoes local search based improvement (see Chapter 3, Section 3.4.4), minimizing actual routing costs, and considering only moves that yield a feasible route. In case this results in another negative reduced cost route, the according column is again added to the pool. Heuristic *LNSCurrBasis* applies Large Neighborhood Search (LNS) (Shaw, 1998) on the routes in the current basis. LNS works as follows. In a removal step, up to 50% of the requests forming the respective route are randomly removed from it and in an insertion step requests are reinserted, considering all requests, using randomized best insertion as described above. These two steps are repeated until no further improvement can be found. Between 15 and 20 non-improving steps are performed. Heuristic *LNSRandConstr* simply improves all solutions obtained by the randomized construction algorithm with LNS.

6.5 The column generation framework

The whole column generation framework is depicted in Algorithm 6.2. During the initialization phase initial columns are generated (see Section 6.5.1) and added to Ω', and a number of pre-processing steps (see Section 6.5.2) are applied. Then, LSP is solved on Ω' and the dual variable values associated with the different constraints are retrieved. Based

Algorithm 6.2 The column generation framework

initialization. generate initial columns by VNS, introduce artificial columns and add them to Ω', do pre-processing (graph pruning, time window tightening).
repeat
  solve LSP on Ω'
  every 10 iterations apply VNS to the current LSP solution (collaborative scheme)
  generate new negative reduced cost columns heuristically (*ConstrHeur*, *LNSCurrBasis*, *RandConstrHeur*, *LNSRandConstr*, *LimLabels*, *LimGraph*)
  if no negative reduced cost columns are found **then**
    run exact label setting algorithm to find negative reduced cost columns
  end if
  add new negative reduced cost columns to Ω'
until no more negative reduced cost columns can be found
return optimal solution of LSP

on these values negative reduced cost columns are generated using the different heuristics in a certain sequence (see Section 6.5.3) and if all fail, the exact dynamic programming procedure. All new negative reduced cost columns are added to Ω' and the LSP is solved again. This is repeated until no new negative reduced cost column can be identified. In this case, the optimal solution to LSP has been found. Additionally, every 10 iterations the proposed VNS heuristic (see Section 6.6) is applied to the current solution of LSP, see Figure 6.1 for the whole framework. This so-called collaborative scheme is depicted in further detail in Section 6.5.4.

6.5.1 Initial columns

An initial set of columns is generated by means of a heuristic algorithm, namely a VNS. The VNS is described in detail in Section 6.6. A limit of $2\text{x}10^4$ iterations is applied no matter whether a feasible solution can be found within this time limit or not. Here, ascending moves are not allowed. In addition, one artificial column for each $i \in P$ is generated, having exactly one 1 at the entry corresponding to i and zeros at all other entries. These column are given a sufficiently large cost of M.

6.5.2 Pre-processing

Before starting the column generation algorithm we perform several pre-processing steps. These refer to time window tightening and graph pruning techniques. They are based on those described in detail by Cordeau (2006). In addition, we use the cyclic time window tightening steps as described in Desrochers et al. (1992); Kallehauge et al. (2005),

$$e_k := \max\left\{e_k, \min[l_k, \min_{(i,k)}(e_i + (t_{ik} + d_i))]\right\}, \tag{6.77}$$

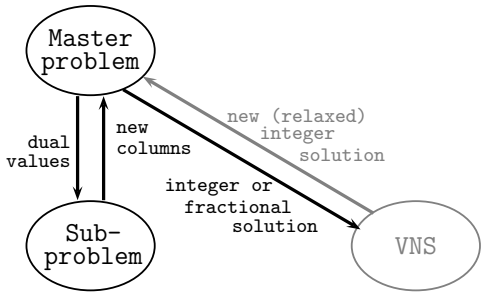


Figure 6.1: The collaborative scheme

$$e_k := \max\left\{e_k, \min[l_k, \min_{(k,j)}(e_j - (t_{kj} + d_k))]\right\}, \tag{6.78}$$

$$l_k := \min\left\{l_k, \max[e_k, \max_{(i,k)}(l_i + (t_{ik} + d_i))]\right\}, \tag{6.79}$$

$$l_k := \min\left\{l_k, \max[e_k, \max_{(k,j)}(l_j - (t_{kj} + d_k))]\right\}. \tag{6.80}$$

These steps are repeated until no further time window tightenings are possible. Note that only those arcs are considered that have not been eliminated in previous pre-processing steps. Eventually, also all those arcs are removed from the graph that would lead to an infeasible solution regarding vehicle capacity restrictions.

6.5.3 Pricing heuristics' sequence

The first pricing heuristic invoked to find new negative reduced cost columns ist selected using roulette wheel selection. Initially only *ConstrHeur* can be chosen. Its score is set to one, the score of all others to zero. Thereafter, in every column generation iteration the score of the heuristic that obtained one or more new negative reduced cost columns is increased by one. Thus, its probability to be chosen as the first heuristic to be tried is increased. The heuristics are ordered as in Ropke and Cordeau (2008): *ConstrHeur* - *LNSCurrBasis* - *RandConstrHeur* - *LNSRandConstr* - *LimLabels* - *LimGraph*. If *ConstrHeur* is chosen as the first heuristic but it fails to yield a new negative reduced cost column for either vehicle types, we switch to *LNSCurrBasis*. If *LNSCurrBasis* fails we switch to *RandConstrHeur* and so on. If *LNSCurrBasis* is chosen as the first heuristic and fails to obtain negative reduced cost columns, only the heuristics following *LNSCurrBasis* in the list are tried in the order above. This applies to all heuristics. If also *LimGraph* (the last one on the list) fails to generate negative reduced cost columns, the exact procedure is started. If this procedure also fails to generate negative reduced cost columns, the optimal solution of the relaxed problem (LSP) has been found. In case it is integer also the optimal solution of

SP has been found. In all labeling algorithms we stop as soon as 50 negative reduced cost columns have been generated. All labeling algorithms use a sorted queue of unprocessed labels. Labels are ordered according to increasing reduced cost.

LSP is resolved on Ω' every time at least one new negative reduced cost column for either vehicle type could be generated. Every heuristic tries to find negative reduced cost columns for T1 and T2 vehicles in alternating order. If new negative reduced cost columns for T1 vehicles are generated, before resolving LSP on the updated Ω', their validity for T2 vehicles is checked. In case they are valid, they are added to the column pool for T2 vehicles. Only then LSP is solved. This is not done if one of the heuristics yields new negative reduced cost columns for T2 vehicles; it is not very likely that columns for T2 vehicles are also valid for T1 vehicles.

6.5.4 Collaborative scheme

In addition, every 10 iterations (one iteration corresponds to finding one or more reduced cost columns, adding these columns to Ω', and solving LSP) the optimal solution of the current LSP is passed to the VNS. If there is still an artificial column in the basis the VNS resumes the search using the last incumbent of the previous run. If there are no more artificial columns in the basis, two different scenarios exist. Either, in case of an integer solution, it is passed on to the VNS as it is. Or, in case of a fractional solution, all duplicate requests are removed before passing it on to the VNS. Duplicate requests are kept on the route that is associated with the u_r closest to one. Empty routes are eliminated. In this case there can be more vehicles in use of a certain type, due to the fractionality of the solution, than actually available. However, it is still passed to the VNS keeping all increased limits on the number of routes per vehicle type and setting the number of drivers available to $\bar{m}_d = \max(m_d, m_{cg} - 1)$ (m_{cg} gives the number of routes used after having removed duplicate requests). Thus, at most as many drivers as there are currently needed minus one or, in case this number is smaller than the original limit, at most the original number of drivers can be used. A new best solution generated by the VNS might be infeasible regarding the number of vehicles of a certain vehicle type employed and in terms of the number of drivers, but no other constraint. This entails that the resulting columns might not be combinable as such but they are all feasible. If the solution obtained by the VNS is of lower cost (actual costs not reduced costs) than the current solution of the master problem after being converted into an integer solution, the corresponding routes are transformed into columns and added to the master. Here the VNS is run for 10^4 iterations and ascending moves are not considered, i.e. a deteriorating solution cannot become a new incumbent solution (see Section 6.6.3). This collaborative scheme is inspired by Danna and Lepape (2005). In contrast to Danna and Lepape (2005) we do not only use the collaborative local search method to improve the best integer solution found so far. We use it to improve the

Algorithm 6.3 *heurVNShetd*

initial solution // determine an initial solution $\int$ and set $\hat{k} := 1$
repeat
  shaking // determine a solution $\int'$ in the neighborhood $\hat{k}$ of $\int$
  local search // apply local search to $\int'$ yielding $\int''$
  move or not // if $\int''$ meets the acceptance requirements the incumbent solution $\int$ is replaced by $\int''$ and $\hat{k} := 1$, otherwise $\hat{k} := (\hat{k} \mod \hat{k}_{max}) + 1$; if $\int''$ is feasible and better than $\int_{best}$, set $\int_{best} := \int''$
until some stopping criterion has been met
return $\int_{best}$

current optimal solution, no matter whether it is integer of fractional.

6.6 The heuristic solution framework

Besides the column generation framework also a heuristic algorithm is employed. It is based on the VNS designed for the standard DARP in Chapter 3 (*heurVNS*). We will denote it as *heurVNShetd*. As in *heurVNS*, infeasibilities are allowed during the search but penalized. The following evaluation function is used:

$$\hat{f}(\int) = \hat{c}(\int) + \sum_{s \in R} \hat{\alpha}_s \hat{q}_s(\int) + \hat{\beta}\hat{d}(\int) + \hat{\gamma}\hat{w}(\int) + \hat{\zeta}\hat{a}(\int). \tag{6.81}$$

The term $\hat{c}(\int)$ gives the routing costs associated with solution $\int$. The terms $\hat{q}_s(\int)$, $\hat{d}(\int)$, $\hat{w}(\int)$, and $\hat{a}(\int)$ represent load violations ($\forall s \in R$), duration violation, time window violation and civilian servant violation (if there are more civilian servants needed in $\int$ than there are available), respectively. As in *heurVNS*, the according penalty parameters $\hat{\alpha}_s$, $\hat{\beta}$, $\hat{\gamma}$, and $\hat{\zeta}$ are dynamically adjusted throughout the search. Again, a solution $\int$ can only become a new best solution $\int_{best}$ if $\hat{q}_s(\int) = \hat{d}(\int) = \hat{w}(\int) = \hat{a}(\int) = 0$ for all $s \in R$. In contrast to *heurVNS*, pre-processing steps are not applied prior to starting the procedure. All design elements (see Algorithm 6.3) that have been subject to modification with respect to *heurVNS* are described in the following. The local search step corresponds to the one of *heurVNS*. It is also employed in the same frequency (see Chapter 3, Sections 3.4.4, 3.4.5).

6.6.1 Initial solution

In *heurVNShetd* a novel initialization procedure for the dHDARP is employed. It generates a possibly infeasible initial solution. In a first step the average number of requests, rounded to the next integer, per vehicle available is computed. Then all requests are inserted in the order they appear in the instance file into the routes, starting with the first route of the first type, opening the next route as soon as the average number of requests per vehicle has

been attained. If a request does not fit into a route due to a lack of the demanded resource on the current vehicle, it is inserted into the next vehicle route with this resource available. The procedure ends as soon as all requests have been inserted into some route. Finally all routes are checked whether a request demands a civilian servant aboard the vehicle. If this is the case, the noon depot is inserted at the best possible position and a civilian servant is assigned to the corresponding vehicle for the according shift (morning or afternoon).

6.6.2 Shaking

During the shaking phase four different neighborhood operators are employed: the first swaps two sequences of vertices, two neighborhoods are based on the move operator, and the last uses the ejection chain idea. In contrast to *heurVNS*, the zero split neighborhood is not considered.

In the swap neighborhood (S) two sequences of vertices are exchanged in the same way as described in Chapter 3. In contrast to all previous VNS implementations, two types of move neighborhoods are employed. The first move neighborhood (M) corresponds in large parts to the move neighborhood used in *heurVNS*. Insertion routes are either selected randomly or, the "closest" route in terms of spatial distance is taken. Since graph pruning techniques are not applied before starting *heurVNShetd* as a stand-alone method, a correction term does not need to be considered. When used in the above described collaborative scheme, graph pruning techniques have been applied. In this case, the same correction term as in Chapter 3 is used. The second move neighborhood (Mx) distinguishes itself from the first one in the way the insertion routes are selected. Here only random selection is employed. It thus corresponds to the move neighborhood employed in *heurVNSws* (see Chapter 4). The chain neighborhood (C) corresponds to the one introduced in Chapter 3.

In contrast to all previous problem versions, in case of the dHDARP, there may be more vehicles in use than drivers available. Therefore, a further design decision has to be taken. It refers to whether a neighborhood operator can move requests to vehicles currently not in use or not. Moving requests to vehicles currently not in use may result in a violation of the maximal number of drivers available. The following strategy is employed. In case of the swap neighborhood, the two routes are only chosen from all routes currently in use. Empty vehicles are not considered. In the first move neighborhood (M), at most one additional route that is currently not assigned to a driver may be chosen as insertion route. In the second move (Mx) and the chain neighborhood, all routes, also those currently not in use, are eligible for selection. Note that, in case $m_t > m_d$ for some $t \in \mathcal{T}$, m_t can be set to $m_t := m_d$.

Both move and also the chain neighborhood operators may such construct a solution with more vehicles in use than drivers available. Such a solution has to be "repaired". The repair procedure employed here simply chooses one route at random out of all non-

empty routes and redistributes the requests forming this route to other non-empty routes. Request insertion is done one by one in the best possible way, using the notion of critical vertices (Cordeau and Laporte, 2003b). The repair procedure is repeated until the number of vehicles in use meets the number of available drivers.

The shaking operators are applied in a similar order as in Chapter 3: S1 – M1 – C1 – S2 – M2 – C2 – S3 – M3 – C3 – S4 – M4 – C4 – Mx4 (the number given in addition to the neighborhood abbreviation indicates the neighborhood size). As in *heurVNSws* and *heurVNShet*, $\hat{k}_{max} = 13$ different neighborhoods are used. The first neighborhood ($\hat{k} = 1$) corresponds to applying shaking operator S1, whereas, e.g. in case of $\hat{k} = 6$ operator C2 will by used. In Chapter 3, the last neighborhood is the parameterless zero split neighborhood, which distributes requests forming a natural sequence to other routes. Here, Mx4 is the last neighborhood. In combination with the repair function it is the strongest diversification mechanism in place, regarding the number of request relocated together with the amount of change possible in terms of the number of vehicles of each vehicle type employed.

6.6.3 Move or not

As in *heurVNS*, the decision whether the search moves to the new solution $\int''$ or not is based on a simulated annealing type acceptance criterion. In addition to the acceptance criteria of Chapter 3, deteriorating solutions may be accepted before the first feasible solution has been identified; every solution that has an evaluation function value $\hat{f}(\int'') \leq 1.05\hat{f}(\int)$ is accepted with a probability of 1% until the first feasible solution has been found. Thereafter the same acceptance criteria as in *heurVNS* are employed.

6.6.4 Route evaluation

Every time a route is modified its routing cost and constraint violations have to be (re-)evaluated. Here the notion of the forward time slack, as shown for the label setting algorithm in Section 6.4.1.4, is applied. In the evaluation function, first, the best lunch location is determined. Thereafter, the forward time slack is calculated and the according constraint violations are calculated. In a separate procedure the noon depot is either inserted in the best possible way or removed, in case there are no more requests on the route that demand a civilian servant. In the local search step, the noon depot insertion/removal procedure is only invoked at the very end. The same applies to the request insertion routine. Whenever a request is removed from a route it is also invoked.

6.7 Computational experiments

All programs were implemented in C++. In the column generation framework the LP solver of CPLEX 11.1 together with Concert Technology 2.6 were used. All experiments

Table 6.1: Artificial instances - data

data set	probability for patient to be seated	on stretcher	in wheelchair	probability for AP	probability CS demanded	m_c	fleet
X	0.50	0.25	0.25	0.00	0.25	$\lceil m(0.5+\hat{\rho})\rceil$	2 T1, 4 T2
Y	0.25	0.25	0.50	0.10	0.50	$\lceil m(1+\hat{\rho})\rceil$	2 T1, 4 T2
Z	0.83	0.11	0.06	0.50	0.50	$\lceil m(1+\hat{\rho})\rceil$	2 T1, 4 T2

AP = Accompanying Person, CS = Civilian Servant
$\hat{\rho}$ randomly chosen in $[0, 1]$
T1: 1 staff seat, 6 patient seats, 1 wheelchair place
T2: 2 staff seats, 1 patient seat, 1 stretcher, 1 wheelchair place

were carried out on a 3.2 GHz Pentium D computer with a memory of 4 GB. Both solution procedures have been tested on three artificial data sets and on real world data. In the following, first, the characteristics of the different instances are described. Then, the results obtained are discussed.

6.7.1 Artificial instances

For each instance of data set "A" by Cordeau (2006), containing between 16 and 48 requests and between 2 and 4 vehicles, three instances with different degrees of heterogeneity have been generated. The characteristics of these instances are depicted in Table 6.1. Setting "X" is the most homogeneous one; 50% of the original users have been converted into seated passengers; 25% into patients on stretchers; and 25% into persons needing a wheelchair. The probability that an Accompanying Person (AP) is present has been set to zero. The number of civilian servants available has been randomly set to between 0.5 of and 1.5 times the number of drivers available (ceiled to the next integer). The probability for a civilian servant to be demanded by a patient has been set to 25%. For setting "Y", 25% of the original users have been transformed into seated patients, 25% into patients on stretchers and 50% into wheelchair patients. 10% of all patients are assumed to be accompanied by someone. The number of civilian servants has been randomly set to at least the number of drivers available and at most to twice the number of drivers. The probability for a civilian servant to be demanded by a patient has been set to 50%. In the third setting, denoted by "Z", 83% of the patients are assumed to be seated, 11% are assumed to be on a stretcher, and 6% are assumed to be in a wheelchair. This setting is based on the data provided by the ARC. Finally, the probability for a patient to be accompanied by someone has been set to 50%. All other settings are equal to those of data set "Y". In all instances the vehicle fleet consists of 2 T1 and 4 T2 vehicles and the number of available drivers has been set to the original number of vehicles. At the start depot a 60-minute time window has been set; and maximum route duration has been reduced by 60 minutes with respect to the original data. As before it is equal for all vehicles. The time window at the noon depot has been set to $e_{2n+2} = e_0 + T/2$ and $l_{2n+2} = e_{2n+2} + 15$.

6.7.2 Real world instances

Furthermore, we apply the different solution algorithms to 15 real world instances from the ARC. They dispose of the following characteristics. As in data set "Z" 83% of the passengers are seated patients, 11% have to be transported on a stretcher, and 6% in a wheelchair. 50% of all these passengers take an accompanying person with them and about 40% demand additional personnel (a civilian servant) on the vehicle. Three T1 vehicles and 31 T2 vehicles are available. Maximum route duration (driver working hours) are limited to 510 minutes. The lunch break has to be held between 11am and 2pm. It lasts 30 minutes. Every driver starts to work between 6:30 am and 8:30 am. These two points in time give the time window at the start depot. The time window at the noon depot lasts from 12:30 until 1:00. Users specify a time window for either the pickup or the drop off location. Time window length is equal to 30 minutes. Maximum user ride times have been set to $\bar{L}_i = t_{i,n+i} + 30$ for all $i \in P$. As mentioned above, ride time limits are not explicitly considered; depending which time window has been specified by the user, the time window for the corresponding location without time window is set relative to the existing time window; in case of a time window at the destination, it is set to $e_i = e_{n+i} - (\bar{L}_i + d_i)$ and $l_i = l_{n+i} - (t_{i,n+i} + d_i)$ at the origin; in case of a time window at the origin, it is set to $e_{n+i} = e_i + d_i + t_{i,n+i}$ and $l_{n+i} = l_i + d_i + \bar{L}_i$ at the corresponding destination.

6.7.3 Column generation results

In a first step two versions of the column generation framework are tested on the artificial data sets. The first version only uses pure column generation, while in the second version the collaborative scheme is employed. Table 6.2 gives the results for pure column generation; Table 6.3 the results for the collaborative scheme. The following information is provided. First, the time needed to compute the initial VNS solution and the total run time, excluding the initial VNS, of the respective program is given. A maximum run time limit of 432000 seconds was imposed for both procedures. Then, the lower bounds and the best integer solutions found throughout the search are given. In case the time limit was reached, no lower bound can be given. The best integer solution found throughout the search is either the optimal integer solution, in case the obtained lower bound is integer, or the solution obtained from solving SP on the set of generated columns within a run time limit of 10 minutes. Furthermore, the status of the obtained lower bound is given (integer (int.), fractional (frac.), or infeasible (inf.)). This is followed by the total number of columns generated, and the number of times the different pricing procedures found at least one new negative reduced cost column. Rows $\bar{X}$, $\bar{Y}$, and $\bar{Z}$ give the average values for the respective data set. Row $\overline{XYZ}$ gives the total average values across all data sets.

In both cases, pure column generation and the collaborative approach, the two heuristic pricing procedures which prove to be the most useful are *ConstrHeur* and *LimLabels*. Also

all other heuristic pricing procedures contribute a number of reduced cost columns and thus should not omitted.

When comparing the two tables, the following differences can be observed. In case of the smaller instances, pure column generation is faster than the collaborative approach. This is due to the fact that in case of the smaller instances, the initial VNS quite often already finds the optimal solution and thus, intermediate calls to the VNS do not improve this solution. They only increase computation times. This relation changes in case of the larger instances. Here, in some cases the collaboration approach is faster while in others, lower computation times are due to pure column generation. The intuition is that calls to the intermediate VNS are not always useful. In many cases they, however, are. When comparing the best integer solution found during the search, the collaborative approach outperforms pure column generation on average. When comparing the results of each instance individually, the collaborative scheme is better or equal in all but two cases. The most remarkable difference is certainly the fact, that with the collaborative scheme in place instance a4-48 of data set "X" can be solved within the time limit, while in case of pure column generation, this instance cannot be solved. From this fact, it can be derived that the integration of a collaborative local search derived method into the column generation framework has a positive impact on the performance of the whole method.

As the original real world derived instances are too large to be solved with either of the two exact procedures, results for these will only be provided by the heuristic method.

6.7.4 Heuristic results

In Table 6.4 the results obtained by means of *heurVNShetd* for the artificial data sets are presented. As stopping criterion a limit on the number of iterations (10^5) is used. For each data set the following information can be taken from the table; the name of the instance; average solution values over five random runs; best solution values out of these five runs; and the respective percentage deviations from the lower bounds (see above). The average percentage gap between the lower bound and the obtained average solution value is less than 2% for all three data sets. Computation times are not really low, but acceptable (less than 9 minutes on average). The rather long computation times with respect to the low total iteration limit are due to the complex evaluation procedure, including a possible repositioning of the noon depot and the appropriate choice of the lunch break location.

As mentioned above, all real world instances were too large to be solved by means of column generation. Therefore, in order to get an idea of how difficult real world derived instances are, we generated two additional real world based data sets. In these two data sets every instance has been reduced by the factor 5 and 3, regarding the original data, respectively; and the time window length has been decreased to 15 minutes. For those instances that could be solved by the column generation framework, the obtained lower

Table 6.2: Artificial instances - pure column generation

	CPU						number of times new cols found by					
	init. VNS	all	LB	best int.	stat.	cols	*Constr-Heur*	*LNS-Curr-Basis*	*Rand-Constr-Heur*	*LNS-Rand-Constr*	*Lim-Labels*	*Lim-Graph*
X												
a2-16	17.50	5.77	299.37	299.37	int.	662	6	1	6	3	5	0
a2-20	79.78	94.93	376.70	376.70	int.	2282	44	9	32	5	27	0
a2-24	133.54	3518.32	461.66	461.66	int.	4217	30	53	13	3	47	0
a3-18	17.80	14.83	291.68	291.68	int.	1232	10	5	9	5	3	0
a3-24	47.53	166.43	351.19	361.39	frac.	2786	51	9	24	5	32	1
a3-30	112.62	4668.83	510.79	510.79	int.	6336	87	39	50	24	93	12
a3-36	-	-	-	602.52	-	14800	123	216	5	6	50	47
a4-16	20.42	4.12	-	-	inf.	711	12	8	4	1	2	0
a4-24	34.63	41.92	388.95	393.57	frac.	2275	23	4	3	1	10	0
a4-32	101.56	557.36	504.79	509.23	frac.	5136	41	40	5	6	28	0
a4-40	-	-	-	610.15	-	12448	147	113	1	6	41	32
a4-48	-	-	-	670.06	-	18725	109	72	103	19	89	47
$\overline{X}$	62.82	1008.06	398.14	462.46		5968	57	47	21	7	36	
Y												
a2-16	54.42	5.79	-	-	inf.	1111	15	4	10	2	2	0
a2-20	77.59	27.98	371.62	371.62	int.	1968	52	10	33	9	21	0
a2-24	234.72	1742.20	-	-	inf.	3389	18	54	27	13	35	0
a3-18	25.30	4.63	295.39	295.39	int.	596	9	2	6	1	3	0
a3-24	92.05	69.11	353.13	359.75	frac.	2806	37	14	37	9	26	0
a3-30	110.69	3278.17	487.13	487.13	int.	7752	86	11	30	26	38	24
a3-36	-	-	-	620.27	-	13328	186	49	40	19	61	47
a4-16	20.46	2.18	-	-	inf.	544	8	2	4	0	4	0
a4-24	59.90	82.75	-	-	inf.	2019	28	24	16	4	14	0
a4-32	96.58	645.16	504.91	510.73	frac.	4751	46	35	3	1	21	4
a4-40	148.65	6287.23	587.56	590.21	frac.	8161	70	25	21	14	61	30
a4-48	198.22	276470.00	717.95	717.95	int.	22932	140	52	107	37	227	122
$\overline{Y}$	101.69	26237.75	473.96	494.13		5780	58	24	28	11	43	19
Z												
a2-16	41.77	7.98	308.30	308.30	int.	1091	24	2	9	3	6	0
a2-20	97.43	449.47	398.65	398.65	int.	2877	39	15	52	12	40	0
a2-24	125.06	1293.53	423.05	423.05	int.	3965	95	31	34	11	47	0
a3-18	26.15	4.40	297.24	297.24	int.	783	13	6	2	1	2	0
a3-24	54.97	104.19	354.19	355.15	frac.	2159	34	10	28	7	26	0
a3-30	122.81	863.53	495.70	498.43	frac.	5322	91	42	27	5	33	1
a3-36	174.07	100628.00	569.36	570.60	frac.	12123	103	18	46	30	93	95
a4-16	25.77	3.30	-	-	inf.	697	11	6	10	0	2	0
a4-24	39.24	47.68	388.69	388.69	int.	1903	24	8	13	6	11	0
a4-32	121.32	769.89	498.34	513.98	frac.	4040	67	47	13	8	22	0
a4-40	119.20	58136.70	581.86	604.28	frac.	9632	72	52	45	12	20	33
a4-48	248.59	57015.70	679.21	679.21	int.	12740	100	39	93	30	163	55
$\overline{Z}$	99.70	18277.03	454.05	457.96		4778	56	23	31	10	39	15
$\overline{XYZ}$	90.01	16156.63	456.34	469.26		5508	57	31	27	10	39	15

cols = columns, frac. = fractional, inf. = infeasible, init. = initial, int. = integer, LB = Lower Bound, stat. = status.

bound will serve to assess the solution quality of *heurVNShetd*.

Table 6.5 contains the results obtained for 8 instances, that have been reduced by the factor 5 with respect to the original data. 7 out of these 8 instances could be solved by means of column generation. The following information is provided; the name of the instance; the size of the instance in terms of the total number of requests n; the number of drivers m_d; and

Table 6.3: Artificial instances - collaborative scheme

	CPU[a]						number of times new cols found by					
	init. VNS	all	LB	best int.	stat.	cols	*Constr-Heur*	*LNS-Curr-Basis*	*Rand-Constr-Heur*	*LNS-Rand-Constr*	*Lim-Labels*	*Lim-Graph*
X												
a2-16	17.50	17.65	299.37	299.37	int.	660	7	3	6	0	7	0
a2-20	79.51	149.48	376.70	376.70	int.	1919	24	7	23	3	18	0
a2-24	132.04	4892.11	461.66	461.66	int.	4453	29	93	28	7	50	0
a3-18	17.70	15.43	291.68	291.68	int.	1143	9	3	6	0	3	0
a3-24	47.82	322.04	351.19	358.29	frac.	2886	52	11	23	16	30	2
a3-30	112.75	1617.41	510.79	510.79	int.	5897	103	21	58	17	77	11
a3-36	-	-	-	600.83	-	16178	129	171	15	10	75	80
a4-16	20.47	5.94	-	-	inf.	665	9	4	7	1	2	0
a4-24	34.33	33.94	388.95	392.37	frac.	2157	17	0	6	0	8	0
a4-32	100.33	675.10	504.79	509.23	frac.	4829	38	15	10	5	30	1
a4-40	-	-	-	608.55	-	14533	181	44	57	32	83	46
a4-48	159.00	192018.00	665.11	665.74	frac.	19313	114	43	63	42	110	86
$\overline{X}$	72.15	19974.71	459.97	461.38		6211	59	35	25	11	41	19
Y												
a2-16	54.31	24.04	-	-	inf.	1132	13	3	4	5	5	0
a2-20	77.34	107.12	371.62	371.62	int.	1649	28	7	40	4	11	0
a2-24	234.83	709.20	-	-	inf.	3415	19	48	29	4	32	0
a3-18	24.96	6.29	295.39	295.39	int.	716	9	1	3	0	4	0
a3-24	91.86	223.08	353.13	355.47	frac.	2863	39	21	46	15	25	0
a3-30	107.60	2540.69	487.13	487.13	int.	7400	61	8	30	5	46	30
a3-36	-	-	-	598.62	-	13803	175	40	37	9	79	58
a4-16	20.40	3.06	-	-	inf.	557	10	1	2	1	3	0
a4-24	59.76	110.69	-	-	inf.	2113	26	25	17	4	10	0
a4-32	95.43	689.41	504.91	507.34	frac.	4990	47	44	3	2	21	5
a4-40	145.27	8329.59	587.56	590.21	frac.	8080	76	24	8	4	62	28
a4-48	197.55	218190.00	717.95	717.95	int.	21355	206	47	113	50	254	99
$\overline{Y}$	100.85	20993.92	489.45	490.46		5673	59	22	28	9	46	18
Z												
a2-16	41.80	32.49	308.30	308.30	int.	972	24	4	14	2	3	0
a2-20	97.05	247.69	398.65	398.65	int.	2415	57	22	48	12	19	0
a2-24	124.48	696.34	423.05	423.05	int.	4178	81	32	35	12	38	0
a3-18	26.11	22.23	297.24	297.24	int.	842	13	8	4	6	2	0
a3-24	54.03	217.16	354.19	355.15	frac.	2275	40	21	46	11	22	0
a3-30	121.22	1370.22	495.70	501.60	frac.	5716	73	40	53	21	54	1
a3-36	169.32	191169.00	569.36	579.61	frac.	12599	99	8	65	54	99	96
a4-16	25.67	8.07	-	-	inf.	630	11	11	2	1	3	0
a4-24	38.77	61.82	388.69	388.69	int.	1837	19	6	7	5	10	0
a4-32	121.41	680.31	498.34	504.26	frac.	4027	50	35	10	3	20	1
a4-40	118.14	29156.20	581.86	603.86	frac.	9511	76	38	23	23	29	26
a4-48	246.36	55650.80	679.21	679.21	int.	12386	81	37	68	37	145	60
$\overline{Z}$	98.70	23276.03	454.05	458.15		4782	52	22	31	16	37	15
$\overline{XYZ}$	91.37	21514.93	465.66	467.95		5555	57	26	28	12	41	18

cols = columns, frac. = fractional, inf. = infeasible, init. = initial, int. = integer, LB = lower bound, stat. = status.
[a] run times in seconds

the number of civilian servants m_c available; the lower bound, where known; and for each iteration limit, the average and the best solution value out of 5 random runs, the according deviation from the lower bound, and the total average run time in seconds. All lower bound solutions are fractional, with one exception, in case of instance aug1108a the obtained lower bound is integer. On average *heurVNShetd* yields solutions within 3.17 percent from the

Table 6.4: Artificial instances - *heurVNShetd* (10^5 iterations, 5 runs)

	data set X					data set Y					data set Z				
	avg.	%	best	%	CPU[a]	avg.	%	best	%	CPU[a]	avg.	%	best	%	CPU[a]
a2-16	299.37	0.00	299.37	0.00	201.34	-	-	-	-	-	308.30	0.00	308.30	0.00	270.89
a2-20	377.14	0.12	376.70	0.00	345.76	371.62	0.00	371.62	0.00	680.22	398.65	0.00	398.65	0.00	436.34
a2-24	461.66	0.00	461.66	0.00	685.8	-	-	-	-	-	426.08	0.72	425.30	0.53	633.21
a3-18	291.68	0.00	291.68	0.00	89.72	295.39	0.00	295.39	0.00	132.68	297.24	0.00	297.24	0.00	128.09
a3-24	359.72	2.43	356.30	1.45	247.14	361.66	2.42	360.78	2.17	314.13	361.00	1.92	355.15	0.27	280.76
a3-30	512.34	0.30	510.79	0.00	516.5	496.02	1.82	487.13	0.00	527.24	506.05	2.09	498.29	0.52	555.64
a3-36	614.31	-	604.35	-	987.97	617.55	-	596.61	-	1108.15	580.01	1.87	572.55	0.56	942.36
a4-16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
a4-24	402.44	3.47	394.34	1.38	169.8	-	-	-	-	-	396.92	2.12	393.13	1.14	180.79
a4-32	516.21	2.26	513.10	1.64	367.19	508.30	0.67	507.29	0.47	340.34	506.36	1.61	504.44	1.22	353.16
a4-40	623.57	-	613.33	-	753.95	595.90	1.42	592.55	0.85	644.27	600.02	3.12	595.72	2.38	617.07
a4-48	689.15	3.61	680.27	2.28	862.78	734.42	2.29	728.74	1.50	1122.68	697.64	2.71	684.56	0.79	1164.09
Avg.	467.96	1.35	463.81	0.75	475.27	497.61	1.23	492.51	0.71	608.71	461.66	1.47	457.58	0.67	505.67

avg. = average
[a] run times in seconds

Table 6.5: Real world based instances (5 times smaller) - *heurVNShetd* (5 runs)

					heurVNShetd 10^5 iterations					*heurVNShetd* 2x10^5 iterations				
	n	m_d	m_c	LB	avg.	%	best	%	CPU[a]	avg.	%	best	%	CPU[a]
aug0508a	29	4	6	377.32	384.96	2.02	383.10	1.53	274.54	383.64	1.68	378.89	0.42	519.25
aug1108a	30	4	6	449.74	458.75	2.00	451.03	0.29	229.47	459.30	2.13	453.15	0.76	445.85
aug1208a	34	4	6	426.01	431.95	1.39	428.99	0.70	313.55	436.23	2.40	428.99	0.70	671.22
aug1308a	33	4	6	474.38	480.80	1.35	479.66	1.11	316.00	483.59	1.94	479.66	1.11	651.54
mai0605a	41	5	7	471.47	488.11	3.53	485.94	3.07	287.16	483.61	2.58	473.38	0.41	590.89
mai0705a	42	5	7	-	664.30	-	654.11	-	471.91	640.08	-	627.32	-	769.51
mai1805a	54	6	9	582.38	623.57	7.07	603.78	3.67	495.44	622.99	6.97	615.81	5.74	937.00
mai2105a	28	4	5	390.65	409.49	4.82	391.62	0.25	190.51	399.35	2.23	391.10	0.12	375.15
Avg.				453.14	492.74	3.17	484.78	1.52	322.32	488.60	2.49	481.04	1.16	620.05

avg. = average
[a] run times in seconds

lower bound, using 10^5 iterations. With an increased limit of 2x10^5 iterations, the average percentage gap is reduced to 2.49. In case of instance mai1805a, the obtained lower bound is highly fractional. This explains why the obtained gap between the heuristic upper bound and the column generation lower bound is so large. It can be assumed that it is at least partly due to a larger integrality gap with respect to the other instances. Total run times are below 16 minutes for all instance.

Table 6.6 provides similar information as Table 6.5. However, for this medium-sized data set, which is again based on the available real world data, considering only every third request, and reducing the number of available drivers and civilian servants accordingly, lower bounds cannot be computed. Thus, we only provide average and best solution values for *heurVNShetd* within a limit of 10^5 and 2x10^5 iterations. The percentage deviations presented in Table 6.6 give the deviations from the best solutions encountered with the two iteration limits. As expected, more iterations lead, on average, to better solution values.

Table 6.7, finally, provides the results for those data sets that are based on ARC data for 15 days in the city of Graz. As mentioned above, all of them are too large to be solved by

Table 6.6: Real world based instances (3 times smaller) - *heurVNShetd* (5 runs)

					heurVNShetd 10^5 iterations					*heurVNShetd* $2x10^5$ iterations				
	n	m_d	m_c	all best	avg.	%	best	%	CPU[a]	avg.	%	best	%	CPU[a]
aug0508b	49	6	9	667.90	676.06	1.22	667.90	0.00	461.64	676.76	1.33	670.34	0.36	833.46
aug1108b	51	6	9	693.31	712.22	2.73	693.31	0.00	428.37	706.65	1.92	696.80	0.50	802.34
aug1208b	58	7	10	718.02	733.79	2.20	718.02	0.00	404.47	735.34	2.41	730.46	1.73	788.95
aug1308b	55	7	10	643.30	653.10	1.52	643.30	0.00	355.91	650.33	1.09	644.15	0.13	731.29
feb0402b	60	7	11	684.51	703.89	2.83	695.09	1.55	444.74	697.84	1.95	684.51	0.00	847.73
mai0605b	69	8	11	764.79	785.08	2.65	764.79	0.00	498.42	777.71	1.69	769.87	0.66	969.59
mai1805b	91	10	15	1107.04	1167.19	5.43	1140.29	3.00	743.60	1129.74	2.05	1107.04	0.00	1362.91
nov0411b	79	9	13	867.97	902.28	3.95	878.21	1.18	598.93	882.35	1.66	867.97	0.00	1173.83
Avg.				768.36	791.70	2.82	775.11	0.72	492.01	782.09	1.76	771.39	0.42	938.76

avg. = average
[a] run times in seconds

Table 6.7: Real world instances - *heurVNShetd* (5 runs)

					heurVNShetd $5x10^5$ iterations					*heurVNShetd* 10^6 iterations				
	n	m_d	m_c	all best	avg.	%	best	%	CPU[a]	avg.	%	best	%	CPU[a]
aug0508	147	17	26	1374.33	1406.03	2.31	1379.42	0.37	6797.44	1392.12	1.29	1374.33	0.00	12241.06
aug1108	154	18	27	1415.55	1447.50	2.26	1422.90	0.52	5121.00	1429.40	0.98	1415.55	0.00	10179.36
aug1208	174	19	29	1632.16	1688.72	3.47	1673.86	2.55	7854.12	1666.13	2.08	1632.16	0.00	15394.96
aug1308	166	19	29	1552.40	1600.94	3.13	1576.00	1.52	6729.45	1565.90	0.87	1552.40	0.00	12795.56
feb0202	233	20	30	2087.65	2144.81	2.74	2103.59	0.76	20555.08	2113.63	1.24	2087.65	0.00	37997.24
feb0402	182	21	32	1677.70	1722.32	2.66	1701.25	1.40	7484.74	1696.75	1.14	1677.70	0.00	14200.60
feb1002	186	18	27	1804.26	1854.35	2.78	1829.44	1.40	12483.68	1839.04	1.93	1804.26	0.00	24718.90
mai0605	208	22	33	1760.82	1851.03	5.12	1833.69	4.14	10917.44	1779.48	1.06	1760.82	0.00	20268.88
mai0705	210	23	35	1964.68	2077.48	5.74	2046.29	4.15	10119.45	1997.39	1.66	1964.68	0.00	18350.90
mai1805	273	30	45	2570.25	2677.39	4.17	2629.02	2.29	11725.66	2601.40	1.21	2570.25	0.00	22719.08
mai2105	140	16	24	1457.36	1494.56	2.55	1476.78	1.33	5745.56	1470.55	0.91	1457.36	0.00	10813.82
nov0411	239	25	38	2155.53	2225.80	3.26	2186.96	1.46	11984.40	2171.58	0.74	2155.53	0.00	23501.62
nov0911	247	24	36	2207.46	2303.18	4.34	2269.98	2.83	15720.66	2248.11	1.84	2207.46	0.00	29666.94
nov1211	192	21	32	1711.99	1805.97	5.49	1784.38	4.23	9152.52	1749.12	2.17	1711.99	0.00	16999.46
nov1611	219	21	32	2097.98	2236.66	6.61	2212.37	5.45	15687.24	2158.05	2.86	2097.98	0.00	30054.52
Avg.				1831.34	1902.45	3.77	1875.06	2.29	10538.56	1858.58	1.47	1831.34	0.00	19993.53

[a] run times in seconds
avg. = average

means of column generation. Therefore, only *heurVNShetd* is applied. Because of the large size of these instances, two different iteration limits have been used. First, *heurVNShetd* is run for $5x10^5$ iterations. This configuration results in solution values that are on average 3.77% worse than the best solution found during both experimental settings. On average, a bit less than 3 hours of run time is needed. Then, the limit is increased to 10^6 iterations. In less than twice the time (on average 5.5 hours), the average gap from the best known solution is reduced to 1.47%.

While for the smallest instances, containing up to 50 requests, 10^5 iterations will lead to solutions of high quality, when compared to the lower bound; for medium-sized real world derived instances, more iterations are necessary to yield solutions of acceptable quality. In case of the largest real world instances, a limit of $5x10^5$ or even 10^6 iterations, if time allows, should be chosen.

6.8 Summary

In this chapter the DARP with driver related constraints has been introduced and a 3-index as well as a set partitioning type formulation have been proposed. Lower bounds by means of column generation have been computed for three sets of artificial instances with differing heterogeneous characteristics. Also a VNS heuristic has been developed and applied to artificial as well as real world (derived) instances. When compared to the lower bounds, in case of the artificial instances, high quality solutions are obtained within rather short run times. When applied to the much larger real world (derived) data sets, more time needs to be given to the procedure in order to obtain good results. Finally, also a collaborative scheme, integrating the proposed variable neighborhood search heuristic into the column generation framework, has been developed. Comparison with "pure" column generation shows that the collaborative scheme improves the efficiency of the original method.

7 Conclusion

In this book, ambulance routing problems with different degrees of heterogeneity have been modeled as static DARP. In Chapter 3, a heuristic solution method for a rather standard DARP version has been developed. Total routing costs are minimized while respecting time windows, maximum user ride time, and a route duration limit. The proposed method combines VNS with ideas from simulated annealing. When compared to the best known solutions from the literature, the developed heuristic yields competitive results. Furthermore, it is flexible enough to accommodate a completely different objective function. Using this modified objective function, the obtained results are compared to those of a genetic algorithm from the literature. In this case, the proposed method clearly outperforms the existing one.

In Chapter 4, the multi-objective DARP has been defined. It is obtained by adding a second objective to the problem dealt with in Chapter 3. Besides routing costs, user inconvenience, in terms of average user ride time, is minimized. Based on the findings of Chapter 3, a tailor-made heuristic algorithm has been developed. It combines a weighted sum based VNS with path relinking in a two-phase scheme. In order to test its performance, an existing branch and cut algorithm has been integrated into the ϵ-constraint framework, yielding optimal Pareto frontiers for small to medium-sized instances. Comparison indicates that the proposed method is able to compute high quality approximations of the true Pareto frontier. The central new aspect of our approach is that a set of trade-off solutions is computed. This allows the decision maker to visualize how much he/she would have to pay for lower user inconvenience or a higher quality of service level. Previous solution algorithms only considered multiple objectives in form of a weighted sum objective function. The main disadvantage of the weighted sum approach refers to the fact that the decision maker has to define weights for the different objectives before starting the optimization procedure. In our case, the decision maker chooses the appropriate solution from a bundle of trade-off solutions, generated in one single run of the optimization procedure.

In Chapter 5, available information from the ARC led to the integration of heterogeneous passenger types and vehicles into the standard formulation. Furthermore, a penalization option for waiting time with passengers aboard a vehicle has been added. A 2-index and a 3-index program for the defined problem have been devised. Each formulation serves as the basis for a branch and cut algorithm. Comparison shows that the 2-index based method outperforms the 3-index based one. The VNS heuristic of Chapter 3 has also been adapted to

this problem version. Several smaller modifications were necessary in order to accommodate the additional requirements. When compared to the exact solutions of small to medium-sized instances, the adapted method yields high quality solutions within short computation times. The aspect of vehicle waiting time with passengers aboard has also been subject to investigation. The results obtained for three data sets indicate that user waiting time can be avoided without considerable cost increase (at most 6%). The ambulance routing problem considered in this chapter represented a first step towards reality.

In Chapter 6, the real world problem, as, e.g., encountered by the ARC, has been solved. In addition to heterogeneous vehicles and patients, staff deployment decisions have been integrated into the problem formulation. At the ARC there are usually more vehicles than drivers available. Thus, at the beginning of each day the appropriate fleet combination has to be determined. Furthermore, some patients need an additional staff member (besides the driver) on the vehicle. The additional staff members are civilian servants (employees who serve their alternative civilian service at the Red Cross). They only work half of the day on a vehicle. Therefore, additional stops at the depot have to be scheduled. Based on recent developments in the literature, a column generation algorithm has been chosen to compute lower bounds. These bounds were used to assess the solution quality of the heuristic method. The central new aspect in the devised column generation algorithm refers to the way the maximum route duration limit is considered in the pricing subproblem, solved by means of dynamic programming. To solve this complex problem situation heuristically, another VNS based solution method, integrating the findings of the previous chapters, has been implemented. Further modifications were necessary in order to accommodate all additional real world constraints. The obtained procedure was first tested on small to medium-sized artificial instances and then on real world data. For the former set of instances lower bounds were computed by means of column generation. Comparison indicates that the modified heuristic method is able to compute high quality solutions within reasonable time. When applied to the much larger real world instances, longer run times are needed in order to obtain good results.

Summarizing, in this book four ambulance routing problems have been considered and solved by a heuristic framework that applies the VNS idea. New and existing neighborhood operators have been employed and several real world conditions have been successfully accommodated. For three out of these four problem versions exact solution procedures based on state-of-the-art solution methods have been devised. The results obtained show that for real world sized problems, exact solution procedure are still not suitable. They are, however, very useful to validate the results of heuristic algorithms. We also investigated the impact of user related objectives. In a Pareto multi-objective framework, the trade-off between costs and mean user ride time has been illustrated. User inconvenience has also been considered in terms of user waiting time on board a vehicle. Instead of the different trade-off solutions, only two extreme solutions have been computed: on the one hand, the

minimum cost solution, on the other hand, a solution without user waiting time on board a vehicle. In this case, the conclusion is that avoiding user waiting time does not yield a significant cost increase.

Future research should involve the investigation of the two-objective problem with additional real world constraints by combining the findings of Chapters 4 to 6. This will lead to additional insights regarding the trade-off between cost and user-oriented objectives.

Our hope is that one day in the near future, the findings of this thesis will serve as the basis for the development of a computer aided routing tool that will support ambulance dispatchers (at the ARC) in their day-to-day work.

A Notation

A.1 Problem formulations

A.1.1 Indices

i, j	user or vertex
k	vehicle
s	transportation mode or resource
r	route
t	vehicle type
0	start depot
$2n+1$	end depot
$2n+2$	noon depot

A.1.2 Parameters

n	number of requests
m	number of vehicles
m_d	number of drivers
m_c	number of civilian servants
m_t	number of vehicles of type t
c_{ij}^k	travel cost for vehicle k on arc (i, j)
c_{ij}	travel cost for arc (i, j)
t_{ij}^k	travel time for vehicle k on arc (i, j)
t_{ij}	travel time for arc (i, j)
e_i	earlier time window at vertex i
l_i	later time window at vertex i
d_i	service time at vertex i
q_i	demand/supply of vertex i
q_i^s	demand/supply of resource s at vertex i
$\hat{q}_i^s$	cumulative demand/supply of resource s at vertex i
a_i	civilian servant demanded at vertex i

$\bar{L}_i$	maximum ride time of user i
$\bar{L}$	maximum user ride time
T^k	maximum route duration of route/vehicle k
T	maximum route duration
$\hat{H}$	planning horizon
C	vehicle capacity
$C^{s,k}$	capacity of resource s on vehicle k
$\hat{C}^{s,k}$	cumulative capacity (including up-grading) for resource s on vehicle k
$C^{s,t}$	capacity of resource s on vehicle type t
$\mathcal{W}_{ij}^{s,k}$	linearization term for load propagation constraints
H	lunch duration
e_H	earlier lunch time window
l_H	later lunch time window
ρ	penalization term for vehicle waiting time with passengers aboard the vehicle
c_r	routing costs of route/column r
$b_{ir} \in \{0,1\}$	whether a request i is covered by route r
g_r	number of civilian servants needed on route r
$\bar{c}_r$	reduced cost of route/column r
$\pi_i, \sigma_i, \lambda, \phi$	dual variable values

A.1.3 Sets and sequences

V	set of all vertices
$V(.)$	set of vertices in .
A	set of all arcs
$A(.)$	set of arcs in .
$G = (V, A)$	a graph
$P = \{1, \ldots, n\}$	set of pickup vertices
$D = \{n+1, \ldots, 2n\}$	set of delivery vertices
D_o	set of origin depots
D_d	set of destination depots
$R = \{0, 1, 2, 3\}$	set of resources (transportation modes)

$\{i, n+i\}$	pickup and delivery vertex pair (a transportation request)
$\mathcal{S}, \mathcal{U}$	sets of vertex subsets with certain characteristics
$S, \hat{S}, \hat{T}, U \subseteq V$	vertex subsets
$\mathcal{F}$	set of infeasible paths
$\mathcal{H}$	set of infeasible paths regarding load starting at a depot
$\pi(S)$	set of predecessors of S
$\sigma(S)$	set of successors of S
$x(S)$	sum over all x_{ij} in S
$x(\hat{S} : \hat{T})$	sum over all x_{ij}, $i \in \hat{S}$, $j \in \hat{T}$
$\hat{R}(S)$	minimum number of vehicles needed to serve all vertices in S
$q(S)$	sum over all q_i, $i \in S$
$\Delta(S)$	sum over all arcs entering and leaving S
$\Delta^+(S)$	sum over all arcs leaving S
$\Delta^-(S)$	sum over all arcs entering S
$A^+_{\hat{T}}$	reachable arc set of $\hat{T}$
$A^-_{\hat{T}}$	reaching arc set of $\hat{T}$
$F = (j_1, \ldots, j_h)$	a sequence (path)
Ω	all feasible routes (columns)
Ω_t	feasible routes (columns) of type t
Ω'	partial set of feasible routes (columns)
$\mathcal{T}$	set of vehicle types

A.1.4 Variables

$x^k_{ij} \in \{0, 1\}$	whether arc (i, j) is traversed by vehicle k
$x_{ij} \in \{0, 1\}$	whether arc (i, j) is traversed by some vehicle
$y^k_i \in \{0, 1\}$	whether vehicle k arrives empty at vertex i
$y_i \in \{0, 1\}$	whether the vehicle arrives empty at vertex i
$v^k_i \in \{0, 1\}$	whether the lunch break of vehicle k is held at vertex i
$z^k_0 \in \{0, 1\}$	whether a civilian servant is on the vehicle in the morning
$z^k_1 \in \{0, 1\}$	whether a civilian servant is on the vehicle in the afternoon
$u_r \in \{0, 1\}$	whether column (route) r is used in the solution
w	number of drivers used as civilian servants
A^k_i	arrival time at vertex i with vehicle k
A_i	arrival time at vertex i
B^k_i	beginning of service at vertex i of vehicle k

B_i	beginning of service at vertex i
L_i^k	ride time of user i on vehicle k
L_i	ride time of user i
$Q_i^{s,k}$	load of resource s when leaving vertex i on vehicle k
Q_i^k	load when leaving vertex i on vehicle k
Q_i	load when leaving vertex i
$\hat{W}_i^k$	waiting time of vehicle k with passengers at vertex i
$\hat{W}_i$	waiting time with passengers aboard a vehicle at vertex i
W_H^k	waiting time of vehicle k until start of lunch break time window

A.2 Variable neighborhood search

$\hat{k}$	index of a neighborhood
$\hat{k}_{max}$	total number of neighborhoods
$\hat{N}_{\hat{k}}$	neighborhood $\hat{k}$
$\int$	a solution
$\int_{init}$	an initial solution
$\int_{best}$	the best feasible solution
$\hat{f}(\int)$, $\hat{f}'(\int)$, $\hat{f}''(\int)$	evaluation function values of a solution $\int$
$\hat{c}(\int)$	routing costs associated with solution $\int$
$\hat{v}(\int)$	total vehicle waiting time with passengers aboard in a solution $\int$
$\hat{r}(\int)$	total excess ride times with respect to direct ride times in solution $\int$
$\hat{l}(\int)$	total waiting time weighted by the passengers aboard the vehicle while waiting in solution $\int$
$\hat{g}(\int)$	sum over all individual route durations in a solution $\int$
$\hat{e}(\int)$	sum over all waiting times between arrival and the beginning of the time window in a solution $\int$ (early arrivals)
$\hat{z}_1(\int)$	normalized routing costs of a solution $\int$
$\hat{z}_2(\int)$	normalized mean user ride time of a solution $\int$
$\bar{K}$	normalization term for minimum cost objective
$\bar{L}$	maximum user ride time and normalization term for minimum mean user ride time objective

$w_1 \dots w_7$	weights
ω	weight of minimum cost objective
ρ	penalization term for $\hat{v}(\int)$
$\hat{q}(\int)$	load violation of a solution $\int$
$\hat{q}_s(\int)$	load violation regarding resource s of a solution $\int$
$\hat{d}(\int)$	duration violation of a solution $\int$
$\hat{w}(\int)$	time window violation of a solution $\int$
$\hat{t}(\int)$	ride time violation of a solution $\int$
$\hat{a}(\int)$	civilian servant violation of a solution $\int$
$\hat{\alpha}$	penalty term for load violation
$\hat{\alpha}_s$	penalty term for resource violation
$\hat{\beta}$	penalty term for duration violation
$\hat{\gamma}$	penalty term for time window violation
$\hat{\tau}$	penalty term for ride time violation
$\hat{\zeta}$	penalty term for civilian servant violation
$\hat{\delta}$	penalty adjustment term
$\overline{\delta}$	upper bound for $\hat{\delta}$
$\underline{\delta}$	lower bound for $\hat{\delta}$
χ	correction term in move neighborhood
$\bar{d}$	total distance of a request from a given route
$\tilde{n}$	number of forbidden arcs in $\bar{d}$
$\bar{t}$	temperature in simulated annealing
p_{rand}	a random number $0 \leq p_{rand} < 1$
l_{LS}	local search frequency
F_i	forward time slack at vertex i
$\hat{F}_i$	modified forward time slack at vertex i
$\mathcal{B}_j$	time buffer at vertex j
$\hat{\mathcal{B}}_j$	modified time buffer at vertex j
W_i	vehicle waiting time at vertex i
A_i	arrival time at vertex i
B_i	beginning of service at vertex i
D_i	departure time from vertex i

L_i	ride time of user i
P_j	ride time of user $i = j - n$ at its destination vertex $(n + 1 \leq j \leq 2n)$
Q_i	load at vertex i
C	vehicle capacity
$\bar{L}$	maximum user ride time
T	maximum route duration
$\hat{H}$	end of planning horizon

A.3 Path relinking

$\mathcal{P}$	the current Pareto frontier
$\int_I$	the initial solution
$\int_G$	the guiding solution
$\hat{M}$	requests that are on different routes in $\int_I$ and $\int_G$
$\hat{F}$	vertices that are on the wrong position with respect to $\int_G$
S_{path}	solutions part of a path from $\int_I$ to $\int_G$
P^k	every k-th solution of $\mathcal{P}$ (ordered according to one objective)
N^k	set of Nadir points based on P^k
n_{seed}	number of seeding solutions

A.4 Quality indicators

$\mathcal{A}$	approximation set
$\mathcal{R}$	reference set
I_ϵ	unary epsilon indicator
I_H	hypervolume indicator
I_{R3}	R3 indicator
$\int_i$	a solution
$\hat{z}_j(\int_i)$	normalized value for objective j of solution $\int_i$
z^*	ideal point
ϵ	minimum factor such that, when applied to $\mathcal{R}$, $\mathcal{A}$ weakly dominates $\mathcal{R}$
λ	a random weight vector
Λ	set of random weight vectors
$u_\lambda(\int)$	utility of a solution $\int$ for some weight vector λ
$u^*(\lambda, \mathcal{A})$	best utility across all $\int \in \mathcal{A}$ for weight vector λ

A.5 Labeling algorithms

κ	a label
η	name of the vertex of the label
δ	departure time at η
Q^s_{cum}	the cumulative load of resource s when leaving η
c_{cum}	the accumulated cost until η
$b \in \{0,1\}$	whether a lunch break has already been made
$\alpha \in \{0,1\}$	whether a civilian servant is aboard the vehicle or not
$o \in \{0,1\}$	whether the noon depot has already been visited or not
$\mathcal{V} \subset \{0,\ldots,2n+5\}$	the set of vertices visited along the path
$\mathcal{O} \subset \{1,\ldots,n\}$	the set of open requests
f	the forward time slack
w_{cum}	the accumulated waiting time
$\mathcal{L}_i$	set of labels at vertex i
Γ	list of unprocessed labels
F^0_j	final time slack
$\dot{q}(\kappa), \dot{z}(\kappa)$	resources coupled by a max term to compute
	minimum route duration
$\tilde{e}_j$	earliest possible arrival time at j
$\tilde{l}_j$	latest possible arrival time at j
s_i	sum over travel and service times along a path ending at i
$\tilde{q}_i$	difference between s_i and $\tilde{l}_i$
$\tilde{z}_i$	maximum of s_i and the earliest possible arrival time reduced by $\tilde{q}_i$
$\tilde{f}^0_i$	forward time slack from origin depot 0 until end of path i
$\tilde{w}_i$	accumulated waiting time until node i

B Additional results

Table B.1: Results for *heurVNS* with modified move neighborhood compared to TS (version 1). Instead of a fixed correction term a varying correction term χ is used. It is randomly chosen in $[4\frac{1.5n}{m}, 4\frac{2.5n}{m})$.

		heurVNS (5 runs)				
	TS[a]	avg.	(%)	best	(%)	CPU[b]
R1a	190.02	**190.02**	0.00	**190.02**	0.00	8.34
R2a	302.08	**301.72**	-0.12	**301.34**	-0.25	25.43
R3a	532.08	534.89	0.53	533.05	0.18	31.10
R4a	572.78	578.43	0.99	574.02	0.22	45.95
R5a	636.97	638.06	0.17	**632.33**	-0.73	164.81
R6a	801.40	804.32	0.36	**794.88**	-0.81	203.88
R7a	291.71	294.12	0.83	**291.71**	0.00	9.60
R8a	494.89	495.84	0.19	**493.36**	-0.31	54.72
R9a	672.44	**664.75**	-1.14	**659.60**	-1.91	186.88
R10a	878.76	884.72	0.68	**874.42**	-0.49	304.70
Avg.	537.31	538.69	0.25	534.47	-0.41	103.54
R1b	164.46	**164.46**	0.00	**164.46**	0.00	11.00
R2b	296.06	299.59	1.19	297.31	0.42	27.52
R3b	493.30	**490.52**	-0.56	**487.20**	-1.24	57.30
R4b	535.90	538.73	0.53	**535.04**	-0.16	99.89
R5b	589.74	590.27	0.09	**583.99**	-0.98	254.68
R6b	743.60	755.48	1.60	749.09	0.74	383.59
R7b	248.21	**248.21**	0.00	**248.21**	0.00	13.87
R8b	462.69	470.37	1.66	467.11	0.95	58.97
R9b	601.96	606.33	0.73	602.32	0.06	203.89
R10b	798.63	818.78	2.52	805.84	0.90	538.16
Avg.	493.46	498.27	0.78	494.06	0.07	164.89
Total avg.	515.38	518.48	0.51	514.27	-0.17	134.21

[a] best known solutions computed by Cordeau and Laporte (2003b)
[b] average run times in minutes on a Pentium D computer with 3.2 GHz

Table B.2: Results for *heurVNS* with modified move neighborhood compared to TS (version 2). Instead of a fixed correction term a varying correction term χ is used. It is randomly set to the number of vertices on a currently existing route excluding the two depots and multiplied by 4.

		heurVNS (5 runs)				
	TS[a]	avg.	(%)	best	(%)	CPU[b]
R1a	190.02	**190.02**	0.00	**190.02**	0.00	8.26
R2a	302.08	303.09	0.34	**301.34**	-0.25	16.96
R3a	532.08	536.15	0.76	532.42	0.06	29.86
R4a	572.78	575.14	0.41	**572.48**	-0.05	62.10
R5a	636.97	642.67	0.90	638.45	0.23	155.11
R6a	801.40	809.93	1.06	805.34	0.49	198.68
R7a	291.71	294.26	0.87	291.71	0.00	9.71
R8a	494.89	497.28	0.48	**491.84**	-0.62	42.17
R9a	672.44	**670.79**	-0.25	**661.47**	-1.63	146.24
R10a	878.76	879.41	0.07	**872.97**	-0.66	311.87
Avg.	537.31	539.87	0.47	535.80	-0.24	98.10
R1b	164.46	164.63	0.10	**164.46**	0.00	10.18
R2b	296.06	298.48	0.82	296.36	0.10	31.64
R3b	493.30	**490.99**	-0.47	**487.97**	-1.08	58.49
R4b	535.90	542.47	1.23	536.54	0.12	131.58
R5b	589.74	592.89	0.53	**583.23**	-1.10	311.71
R6b	743.60	749.57	0.80	745.70	0.28	374.70
R7b	248.21	248.88	0.27	**248.21**	0.00	13.23
R8b	462.69	468.04	1.16	464.09	0.30	74.59
R9b	601.96	608.03	1.01	602.67	0.12	187.50
R10b	798.63	807.22	1.08	**797.00**	-0.20	632.50
Avg.	493.46	497.12	0.65	492.62	-0.15	182.61
Total avg.	515.38	518.50	0.56	514.21	-0.19	140.35

[a] best known solutions computed by Cordeau and Laporte (2003b)
[b] average run times in minutes on a Pentium D computer with 3.2 GHz

List of Abbreviations

AP	Accompanying Person
ARC	Austrian Red Cross
C	Chain neighborhood
CS	Civilian Servant
DARP	Dial-A-Ride Problem
dHDARP	Heterogeneous Dial A Ride Problem with driver related constraints
GA	Genetic Algorithm
HDARP	Heterogeneous Dial A Ride Problem
IP	Integer Program
LNS	Large Neighborhood Search
LP	Linear Program
LS	Local Search
LSP	Linear relaxation of the Set Partitioning problem
M	Move neighborhood
MIP	Mixed Integer Program
MOSA	Multi-Objective Simulated Annealing
MOTS	Multi Objective Tabu Search
P-ACO	Pareto Ant Colony Optimization
P-SA	Pareto Simulated Annealing
PDP	Pickup and Delivery Problem
PDPTW	Pickup and Delivery Problem with Time Windows
PDVRP	Pickup and Delivery Vehicle Routing Problems
S	Swap neighborhood
SDARP	Single vehicle Dial-A-Ride Problem
SP	Set Partitioning problem
SPDP	Single vehicle Pickup and Delivery Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problems with Backhauls
VRPCB	Vehicle Routing Problem with Clustered Backhauls

VRPDDP ..... Vehicle Routing Problem with Divisible Deliveries and Pickups
VRPMB ..... Vehicle Routing Problem with Mixed Backhauls
VRPPD ..... Vehicle Routing Problems with Pickups and Deliveries
VRPSDP ..... Vehicle Routing Problem with Simultaneous Deliveries and Pickups
Z ..... Zero split neighborhood

Bibliography

Aarts, E. and Lenstra, J. (1997). *Local Search in Combinatorial Optimization*. Wiley, Chichester.

Aiex, R. M., Binato, S., and Resende, M. G. C. (2003). Parallel GRASP with path relinking for job shop scheduling. *Parallel Comput*, 29:393–430.

Aiex, R. M., Resende, M. G. C., Pardalos, P. M., and Toraldo, G. (2005). GRASP with path relinking for the three-index assignment problem. *INFORMS J Comput*, 17:224–247.

Aldaihani, M. and Dessouky, M. M. (2003). Hybrid scheduling methods for paratransit operations. *Comput Ind Eng*, 45:75–96.

Alfa, A. S. (1986). Scheduling of vehicles for transportation of elderly. *Transport Plan Tech*, 11:203–212.

Alshamrani, A., Mathur, K., and Ballou, R. H. (2007). Reverse logistics: Simultaneous design of delivery routes and returns strategies. *Comput Oper Res*, 34:595–619.

Ambrosini, M., Caruso, T., Foresti, S., and Righini, G. (2004). A GRASP for the pickup and delivery problem with rear loading. Technical Report Note del Polo - Ricerca n.65, DTI, University of Milan.

Angelelli, E. and Mansini, R. (2002). The vehicle routing problem with time windows and simultaneous pick-up and delivery. In Klose, A., Speranza, M. G., and Van Wassenhove, L. N., editors, *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, pages 249–267, Berlin-Heidelberg. Springer.

Anily, S. (1996). The vehicle-routing problem with delivery and back-haul options. *Naval Res Logist*, 43:415–434.

Anily, S. and Bramel, J. (1999). Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Res Logist*, 46:654–670.

Anily, S. and Mosheiov, G. (1994). The traveling salesman problem with delivery and backhauls. *Oper Res Lett*, 16:11–18.

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000a). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36:69–79.

Ascheuer, N., Krumke, S. O., and Rambau, J. (2000b). Online dial-a-ride problems: Minimizing the completion time. In *STACS 2000: 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000. Proceedings*, LNCS, pages 639–650, Berlin-Heidelberg. Springer.

Assad, A. A. (1988). Modeling and implementation issues in vehicle routing. In *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in management science and systems*, pages 7–45, Amsterdam. North-Holland.

Attanasio, A., Cordeau, J.-F., Ghiani, G., and Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Comput*, 30:377–387.

Balas, E., Fischetti, M., and Pulleyblank, W. R. (1995). The precedence-constraint asymmetric traveling salesman polytope. *Math Programming*, 68:241–265.

Baldacci, R., Battara, M., and Vigo, D. (2007). Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs. Technical report, DEIS, University Bologna, Italy.

Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A. (2003). An exact algorithm for the traveling salesman problem with deliveries and collections. *Networks*, 42:26–41.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Oper Res*, 46:316–329.

Baugh, J. W., Krishna, G., Kakivaya, R., and Stone, J. R. (1998). Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Eng Optim*, 30:91–123.

Beaudry, A., Laporte, G., Melo, T., and Nickel, S. (2008). Dynamic transportation of patients to hospitals. *OR Spectrum*. to appear (available online).

Belisle, J.-P., Desrosiers, J., Dumas, Y., Rousseau, J.-M., Roy, S., and Soumis, F. (1986). The impact on vehicle routing of various operational rules of a tansportation system for handicapped. In *Third International Conference on Mobility and Transportation of Handicapped Persons, Office of the Secretary of Transportation, Washington, DC 20590, 6.47–6.50*.

Bent, R. and van Hentenryck, P. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Comput Oper Res*, 33:875–893.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15:1–31.

Bergvinsdottir, K. B. (2004). The genetic algorithm for solving the dial-a-ride problem. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark.

Bianchessi, N. and Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Comput Oper Res*, 34:578–594.

Bodin, L. and Sexton, T. (1986). The multi-vehicle subscriber dial-a-ride problem. *TIMS Stud Manag Sci*, 22:73–86.

Borndörfer, R., Grötschel, M., Klostermeier, F., and Küttner, C. (1997). Telebus Berlin: Vehicle scheduling in a dial-a-ride system. Technical Report SC 97-23, Konrad-Zuse-Zentrum für Informationstechnik Berlin.

Brandão, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *Eur J Oper Res*, 173:540–555.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS J Comput*, 15:347–368.

Bräysy, O. and Gendreau, M. (2005). Vehicle routing with time windows. Part II: Metaheuristics. *Transport Sci*, 39:119–139.

Caramia, M., Italiano, G. F., Oriolo, G., Pacifici, A., and Perugia, A. (2002). Routing a fleet of vehicles for dynamic combined pickup and deliveries services. In Chamoni, P., Leisten, R., Martin, A., Minnemann, J., and Stadtler, H., editors, *Operations Research Proceedings 2001*, pages 3–8, Berlin, Germany. Springer.

Caricato, P., Ghiani, G., Grieco, A., and Guerriero, E. (2003). Parallel tabu search for a pickup and delivery problem under track contention. *Parallel Comput*, 29:631–639.

Carlson, R. C. (1976). Anatomy of a systems failure: Dial-a-ride in Santa Clara County, California. *Transportation*, 5:3–16.

Carlton, W. B. (1995). *A tabu search approach to the general vehicle routing problem.* PhD thesis, University of Texas at Austin.

Carrabs, F., Cordeau, J.-F., and Laporte, G. (2007). Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS J Comput*, 19:618–632.

Casco, D., Golden, B., and Wasil, E. (1988). Vehicle routing with backhauls: Models, algorithms, and case studies. In Golden, B. and Assad, A., editors, *Vehicle Routing: Methods and Studies*, pages 127–147, Amsterdam. North-Holland.

Castelli, L., Coslovich, L., Pesenti, R., and Ukovich, W. (2002). Improving techniques for an on-line dial-a-ride problem with time windows and capacity constraints. In *Proceedings of the 13th Mini-EURO Conference and 9th Meeting of the EURO Working Group on Transportation, Bari, Italy - June 10-13 2002*.

Chalasani, P. and Motwani, R. (1999). Approximating capacitated routing and delivery problems. *SIAM J Comput*, 28:2133–2149.

Chen, J.-F. and Wu, T.-H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *J Oper Res Soc*, 57:579–587.

Chisman, J. A. (1975). The clustered traveling salesman problem. *Comput Oper Res*, 2:115–119.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *Oper Res Quart*, 20:309–318.

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 315–338, Chichester. Wiley.

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

Coja-Oghlan, A., Krumke, S. O., and Nierhoff, T. (2005). A hard dial-a-ride problem that is easy on average. *J Sched*, 8:197–210.

Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., and Trubian, M. (1996). Heuristics from nature for hard combinatorial optimization problems. *Int Trans Oper Res*, 3:1–21.

Colorni, A. and Righini, G. (2001). Modeling and optimizing dynamic dial-a-ride problems. *Int Trans Oper Res*, 8:155–166.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Oper Res*, 54:573–586.

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

Cordeau, J.-F., Iori, M., Laporte, G., and Salazar-González, J. J. (2006). A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Technical Report OR-06-01, DEIS, University of Bologna.

Cordeau, J.-F. and Laporte, G. (2003a). The dial-a-ride problem (DARP): Variants modeling issues and algorithms. *4OR*, 1:89–101.

Cordeau, J.-F. and Laporte, G. (2003b). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transport Res B-Meth*, 37:579–594.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Ann Oper Res*, 153:29–46.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., and Savelsbergh, M. W. P. (2004). Transportation on demand. In *Handbooks in Operations Research and Management Science*. Elsevier, North-Holland, Amsterdam. (to appear).

Cordeau, J.-F., Laporte, G., and Ropke, S. (2007). Recent models and algorithms for one-to-one pickup and delivery problems. In Golden, B., Raghavan, R., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*. (to appear).

Coslovich, L., Pesenti, R., and Ukovich, W. (2005). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *Eur J Oper Res*. (to appear, available online).

Creput, J.-C., Koukam, A., Kozlak, J., and Lukasik, J. (2004). An evolutionary approach to pickup and delivery problem with time windows. In Bubak, M., van Albada, G. D., Sloot, P. M. A., and Dongarra, J. J., editors, *Computational Science - ICCS 2004: 4th International Conference, Krakow, Poland, June 6-9, 2004, Proceedings*, pages 1102–1108. Springer.

Crispim, J. and Brandão, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *J Oper Res Soc*, 56:1296–1302.

Crispim, J. and Brandão, J. (2001). Reactive tabu search and variable neighbourhood descent applied to the vehicle routing problem with backhauls. In *MIC 2001 4th Metaheuristic International Conference, Porto, Portugal, July 16-20, 2001.*

Cullen, F., Jarvis, J., and Ratliff, D. (1981). Set partitioning based heuristics for interactive routing. *Networks*, 11:125–143.

Czyzak, P. and Jaszkiewicz, A. (1996). A multiobjective metaheuristic approach to the localization of a chain of petrol stations by the capital budgeting model. *Control Cybern*, 25:177–187.

Czyzak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *J Multi-Crit Dec Anal*, 7:34–47.

Daganzo, C. F. (1978). An approximate analytic model of many-to-many demand responsive transportation systems. *Transport Res*, 12:325–333.

Daganzo, C. F. (1984). Checkpoint dial-a-ride systems. *Transport Res B-Meth*, 18:315–327.

Danna, E. and Lepape, C. (2005). Branch-and-price heuristics: a case study on the vehicle routing problem with time windows. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, pages 99–129.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Oper Res*, 8:101–111.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T Evolut Comput*, 6:182–197.

Deif, I. and Bodin, L. D. (1984). Extensions of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In *Proceedings of the Babson College Conference of Software Uses in Transportation and Logistics Management, Babson Park, MA*, pages 75–96.

Dell'Amico, M., Righini, G., and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transport Sci*, 40:235–247.

Derigs, U. and Döhmer, T. (2008). Indirect search for the vehicle routing problem with pickup and delivery and time windows. *OR Spectrum*, 30:149–165.

Derigs, U. and Metz, A. (1992). A matching-based approach for solving a delivery/pick-up VRP with time constraints. *OR-Spektrum*, 14:91–106.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., and Soumis, F. (2002). VRP with pickup and delivery. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, pages 225–242, Philadelphia, PA. SIAM.

Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors (2005). *Column Generation*. Number 5 in GERAD 25th Anniversary. Springer.

Desaulniers, G. and Villeneuve, D. (2000). The shortest path problem with time windows and linear waiting costs. *Transport Sci*, 34(3):312–319.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354.

Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Oper Res Lett*, 10:27–36.

Desrochers, M., Lenstra, J. K., Savelsbergh, M. W. P., and Soumis, F. (1988). Vehicle routing with time windows: Optimization and approximation. In Golden, B. L. and Assad, A. A., editors, *Vehicle Routing: Methods and Studies.*, pages 65–84, Amsterdam. Elsevier (North-Holland).

Desrosiers, J. and Dumas, Y. (1988). The shortest path for the construction of vehicle routes with pick-up, delivery and time constraints. In Eiselt, H. and Pederzoli, G., editors, *Advances in Optimization and Control*, Lecture Notes in Economics and Mathematical Systems, pages 144–157, Heidelberg Berlin. Springer.

Desrosiers, J., Dumas, Y., and Soumis, F. (1986). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *Am J Math Manag Sci*, 6:301–325.

Desrosiers, J., Dumas, Y., and Soumis, F. (1988). The multiple vehicle dial-a-ride-problem. In Daduna, J. and Wren, A., editors, *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems*, volume 308, pages 15–27, Berlin. Springer.

Desrosiers, J., Dumas, Y., Soumis, F., Taillefer, S., and Villeneuve, D. (1991). An algorithm for mini-clustering in handicapped transport. Technical Report G-91-02, HEC, Montréal, Canada.

Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23:79–96.

Dethloff, J. (2002). Relation between vehicle routing problems: An insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *J Oper Res Soc*, 53:115–118.

Dial, R. (1995). Autonomous dial-a-ride transit introductory overview. *Transport Res C-Emer*, 3:261–275.

Diana, M. (2004). Innovative systems for the transportation disadvantaged: towards more efficient and operationally usable planning tools. *Transport Plan Tech*, 27:315–331.

Diana, M. (2006). The importance of information flows temporal attributes for the efficient scheduling of dynamic demand responsive transport services. *J Adv Transport*, 40:23–46.

Diana, M., Dessouky, M., and Xia, N. (2006). A model for the fleet sizing of demand responsive transportation services with time windows. *Transport Res B-Meth*, 40:651–666.

Diana, M. and Dessouky, M. M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transport Res B-Meth*, 38:539–557.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer Math*, 1:269–271.

Doerner, K. F., Gutjahr, W. J., Hartl, R. F., Strauss, C., and Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Ann Oper Res*, 131:79–99.

Dror, M., Fortin, D., and Roucairol, C. (1998). Redistribution of self-service electric cars: A case of pickup and delivery. Technical Report W.P. 3543, INRIA-Rocquencourt, Rocquencourt, France.

Duhamel, C., Potvin, J.-Y., and Rousseau, J.-M. (1997). A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transport Sci*, 31:49–59.

Dumas, Y., Desrosiers, J., and Soumis, F. (1989). Large scale multi-vehicle dial-a-ride problems. Technical Report G-89-30, HEC, Montréal, Canada.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *Eur J Oper Res*, 54:7–22.

Ehrgott, M. and Gandibleux, X. (2004). Approximative solution methods for multiobjective combinatorial optimization. *TOP*, 12:1–89.

Ehrgott, M. and Tenfelde-Podehl, D. (2003). Computation of ideal and Nadir values and implications for their use in MCDM methods. *Eur J Oper Res*, 151:119–139.

Elmberg, C. M. (1978). Dial-a-ride with customer operated dispatching. *Transportation*, 7:35–43.

Eurostat (2004). *Energy, transport and environment indicators – Data 1991–2001.* Luxembourg: Office for Official Publications of the European Communities. Online: http://europa.eu.int/comm/eurostat/.

Eurostat (2006). *Key figures on Europe – Statistical Pocketbook 2006 – Data 1995–2005.* Luxembourg: Office for Official Publications of the European Communities. Online: http://europa.eu.int/comm/eurostat/.

Fabri, A. and Recht, P. (2006). On dynamic pickup and delivery vehicle routing with several time windows and waiting times. *Transport Res B-Meth*, 40:335–350.

Feuerstein, E. and Stougie, L. (2001). On-line single-server dial-a-ride problems. *Theor Comput Sci*, 268:91–105.

Fischetti, M. and Toth, P. (1989). An additive bounding procedure for combinatorial optimization problems. *Oper Res*, 37:319–328.

Fischetti, M., Toth, P., and Vigo, D. (1994). A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Oper Res*, 42:846–859.

Fu, L. (2002a). Scheduling dial-a-ride paratransit under time varying, stochastic congestion. *Transport Res B-Meth*, 36:485–506.

Fu, L. (2002b). A simulation model for evaluating advanced dial-a-ride paratransit systems. *Transport Res A-Pol*, 36:291–307.

Funke, B., Grünert, T., and Irnich, S. (2005). Local search for vehicle routing and scheduling problems: Review and conceptual integration. *J Heuristics*, 11:267–306.

Gandibleux, X., Mezdaoui, N., and Fréville, A. (1997). A tabu search procedure to solve multiobjective combinatorial optimization problems. In Caballero, R., Ruiz, F., and Steuer, R., editors, *Advances in Multiple Objective and Goal Programming*, Lecture Notes in Economics and Mathematical Systems, pages 291–300, Heidelberg-Berlin. Springer.

Gandibleux, X., Morita, H., and Katoh, N. (2004). A population-based metaheuristic for solving assignment problems with two objectives. *J Math Model Algorithm.* accepted.

Ganesh, K. and Narendran, T. T. (2007). CLOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. *Eur J Oper Res*, 178:699–717.

Gélinas, S., Desrochers, M., Desrosiers, J., and Solomon, M. M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Ann Oper Res*, 61:91–109.

Gendreau, M., Guertin, F., Potvin, J., and Seguin, R. . (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transport Res C-Emer*, 14:157–174.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Oper Res*, 40(6):1086–1094.

Gendreau, M., Hertz, A., and Laporte, G. (1996a). The traveling salesman problem with backhauls. *Comput Oper Res*, 23:501–508.

Gendreau, M., Laporte, G., and Potvin, J.-Y. (1996b). Heuristics for the clustered traveling salesman problem. *Combin Optim*, 1:41–56.

Gendreau, M., Laporte, G., and Potvin, J.-Y. (2002). Metaheuristics for the capacitated vrp. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem.*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, pages 129–154, Philadelphia. SIAM.

Gendreau, M., Laporte, G., and Vigo, D. (1999). Heuristics for the traveling salesman problem with pickup and delivery. *Comput Oper Res*, 26:699–714.

Gendreau, M. and Potvin, J.-Y. (1998). Dynamic vehicle routing and dispatching. In Crainic, T. and Laporte, G., editors, *Fleet Management and Logistics*, pages 115–126, New York. Kluwer.

Ghamlouche, I., Crainic, T. G., and Gendreau, M. (2004). Path relinking, cycle-based neighborhoods and capacitated multicommodity network design. *Ann Oper Res*, 131:109–133.

Ghaziri, H. and Osman, I. H. (2003). A neural network algorithm for the traveling salesman problem with backhauls. *Comput Ind Eng*, 44:267–281.

Ghaziri, H. and Osman, I. H. (2006). Self-organizing feature maps for the vehicle routing problem with backhauls. *J Sched*, 9:97–114.

Ghiani, G., Guerriero, F., Laporte, G., and Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *Eur J Oper Res*, 151:1–11.

Gillett, B. E. and Johnson, J. G. (1976). Multi-terminal vehicle dispatch algorithm. *Omega*, 4:639–641.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting stock problem. *Oper Res*, 9:849–859.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput Oper Res*, 13:533–549.

Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Appl Math*, 65:223–253.

Glover, F. and Laguna, M. (1997). *Tabu search*. Kluwer, Norwell.

Goetschalckx, M. and Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *Eur J Oper Res*, 42:39–51.

Golden, B., Baker, E., Alfaro, J., and Schaffer, J. (1985). The vehicle routing problem with backhauling: Two approaches. In Hammesfahr, R. D., editor, *Proceedings of the Twenty-First Annual Meeting of S. E. TIMS*, pages 90–92, Myrtle Beach, SC.

Gribkovskaia, I., Halskau, Ø., Laporte, G., and Vlček, M. (2007). General solutions to the single vehicle routing problem with pickups and deliveries. *Eur J Oper Res*, 180:568–584.

Grötschel, M. and Padberg, M. W. (1985). Polyhedral theory. In *The traveling salesman problem*, pages 251–305, New York. Wiley.

Gutenschwager, K., Niklaus, C., and Voss, S. (2004). Dispatching of an electronic monorail system: Applying metaheuristics to an online pickup and delivery problem. *Transport Sci*, 38:434–446.

Halse, K. (1992). *Modeling and solving complex vehicle routing problems.* PhD thesis, Institute of Mathematical Statistics and Operations Research (IMSOR), Technical University of Denmark.

Halskau, Ø., Gribkovskaia, I., and Myklebost, K. N. B. (2001). Models for pick-up and deliveries from depots with lasso solutions. In *Proceedings of the 13th Annual Conference on Logistics Research – NOFOMA 2001 Collaboration in logistics: Connecting Islands using Information Technology, Reykjavik, Iceland, 2001-06-14 – 2001-06-15, Chalmers University of Technology, Göteborg, Sweden*, pages 279–293.

Hanne, T., Melo, T., and Nickel, S. (2007). Bringing robustness to patient flow management through optimized patient transports in hospitals. *Interfaces.* to appear.

Hansen, M. P. and Jaszkiewicz, A. (1998). Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical Unversity of Denmark.

Hasama, T., Kokubugata, H., and Kawashima, H. (1998). A heuristic approach based on the string model to solve vehicle routing problem with backhauls. In *Proceedings of the 5th World Congress on Intelligent Transport Systems (ITS), Seoul, 1998.*

Hauptmeier, D., Krumke, S. O., and Rambau, J. (2000). The online dial-a-ride problem under reasonable load. In *Algorithms and Complexity: 4th Italian Conference, CIAC 2000, Rome, Italy, March 2000. Proceedings.*, LNCS, pages 125–136. Springer.

Healy, P. and Moll, R. (1995). A new extension of local search applied to the dial-a-ride problem. *Eur J Oper Res*, 83:83–104.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *Eur J Oper Res*, 195:791–802.

Hernández-Pérez, H. and Salazar-González, J. J. (2003). The one-commodity pickup-and-delivery travelling salesman problem. In Jünger, M., Reinelt, G., and Rinaldi, G., editors, *Combinatorial Optimization - Eureka, You Shrink!*, volume 2570 of *LNCS*, pages 89–104. Springer.

Hernández-Pérez, H. and Salazar-González, J. J. (2004a). A branch-and cut algorithm for the traveling salesman problem with pickup and delivery. *Discrete Appl Math*, 145:126–139.

Hernández-Pérez, H. and Salazar-González, J. J. (2004b). Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transport Sci*, 38:245–255.

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

Ho, S. C. and Haugland, D. (2004). Local search heuristics for the probabilistic dial-a-ride problem. Technical Report 286, University of Bergen.

Hoff, A. and Løkketangen, A. (2006). Creating lasso-solutions for the traveling salesman problem with pickup and delivery by tabu search. *Cent Eur J Oper Res*, 14:125–140.

Hoos, H. and Stützle, T. (2005). *Stochatic Local Search Foundations and Applications*. Morgan Kaufmann Publishers, Elsevier, San Francisco, CA.

Horn, M. E. T. (2002a). Fleet scheduling and dispatching for demand-responsive passenger services. *Transport Res C-Emer*, 10:35–63.

Horn, M. E. T. (2002b). Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transport Res A-Pol*, 36:167–188.

Hunsaker, B. and Savelsbergh, M. W. P. (2002). Efficient feasibility testing for dial-a-ride problems. *Oper Res Lett*, 30:169–173.

Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M. M., and Villeneuve, D. (1995). A request clustering algorithm for door-to-door handicapped transportation. *Transport Sci*, 29:63–78.

Irnich, S. (2007). Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*. to appear, available online.

Jaillet, P. and Stafford, M. (2001). Online searching. *Oper Res*, 49:501–515.

Jaw, J., Odoni, A. R., Psaraftis, H. N., and Wilson, N. H. M. (1986). A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. *Transport Res B-Meth*, 20:243–257.

Jih, W.-R. and Hsu, Y.-J. (1999). Dynamic vehicle routing using hybrid genetic algorithms. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 453–458, Los Alamitos, CA. IEEE Computer Society.

Jongens, K. and Volgenant, T. (1985). The symmetric clustered traveling salesman problem. *Eur J Oper Res*, 19:68–75.

Jørgensen, R. M., Larsen, J., and Bergvinsdottir, K. B. (2007). Solving the dial-a-ride problem using genetic algorithms. *J Oper Res Soc*, 58:1321–1331.

Jozefowiez, N., Semet, F., and Talbi, E.-G. (2007a). The bi-objective covering tour problem. *Comput Oper Res*, 34:1929–1942.

Jozefowiez, N., Semet, F., and Talbi, E.-G. (2007b). Target aiming Pareto search and its applications to the vehicle routing problem with route balancing. *J Heuristics*, 13:455–469.

Jozefowiez, N., Semet, F., and Talbi, E.-G. (2008). Multi-objective vehicle routing problems. *Eur J Oper Res*, 189:293–309.

Jung, S. and Haghani, A. (2000). A genetic algorithm for a pick-up and delivery problem with time windows. *Transport Res Rec*, 1733:1–7.

Kalantari, B., Hill, A. V., and Arora, S. R. (1985). An algorithm for the traveling salesman problem with pickup and delivery customers. *Eur J Oper Res*, 22:377–386.

Kallehauge, B., Larsen, J., Madsen, O. B. G., and Solomon, M. M. (2005). Vehicle routing problems with time windows. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, Nen York. Springer.

Kikuchi, S. (1984). Scheduling of demand-responsive transit vehicles. *J Transp Eng*, 110:511–520.

Kikuchi, S. and Rhee, J. (1989). Scheduling algorithms for demand-responsive transportation system. *J Transp Eng*, 115:630–645.

Kirkpatrick, S., Gelatt Jr., C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report TIK-Report No. 214, Computer Engineering and Networks Laboratory, ETH Zurich.

Kontoravdis, G. and Bard, J. F. (1995). A GRASP for the vehicle routing problem with time windows. *ORSA J Comput*, 7(1):10–23.

Krumke, S. O., de Paepe, W. E., Poensgen, D., Lipmann, M., Marchetti-Spaccamela, A., and Stougie, L. (2005). On minimizing the maximum flow time in the online dial-a-ride problem. In *Approximation and Online Algorithms: Third International Workshop, WAOA 2005, Palma de Mallorca, Spain, October 6-7, 2005, Revised Selected Papers*, LNCS, pages 258–269. Springer.

Lacomme, P., Prins, C., and Sevaux, M. (2006). A genetic algorithm for a bi-objective capacitated arc routing problem. *Comput Oper Res*, 33:3473–3493.

Landrieu, A., Mati, Y., and Binder, Z. (2001). A tabu search heuristic for the single vehicle pickup and delivery problem with time windows. *J Intell Manuf*, 12:497–508.

Laporte, G., Potvin, J.-Y., and Quilleret, F. (1996). A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *J Heuristics*, 2:187–200.

Lau, H. C. and Liang, Z. (2001). Pickup and delivery with time windows : Algorithms and test case generation. In *IEEE Computer Society, eds, 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*, pages 333–340.

Lau, H. C. and Liang, Z. (2002). Pickup and delivery with time windows : Algorithms and test case generation. *Int J Artif Intell Tools*, 11:455–472.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon constraint method. *Eur J Oper Res*, 169:932–942.

Laysgaard, J. (2006). Reachability cuts for the vehicle routing problem with time windows. *Eur J Oper Res*, 175:210–223.

Li, H. and Lim, A. (2001). A metaheuristic for the pickup and delivery problem with time windows. In *13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*, pages 333–340, Los Alamitos, CA. IEEE Computer Society.

Li, H., Lim, A., and Huang, J. (2001). Local search with annealing-like restarts to solve the VRPTW. Technical report, Department of Computer Science, National University of Singapore.

Lim, A., Wang, F., and Xu, Z. (2005). The capacitated traveling salesman problem with pickups and deliveries on a tree. In Deng, X. and Du, D., editors, *Algorithms and Computation: 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005. Proceedings*, pages 1061–1070.

Lim, H., Lim, A., and Rodrigues, B. (2002). Solving the pickup and delivery problem with time windows using squeaky wheel optimization with local search. In *American Conference on Information Systems, AMICS 2002, Dallas, USA*.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *AT&T Tech J*, 44:2245–2269.

Lipmann, M., Lu, X., de Paepe, W. E., Sitters, R. A., and Stougie, L. (2004). On-line dial-a-ride problems under a restricted information model. *Algorithmica*, 40:319–329.

Little, J., Murty, K., Sweeney, D., and Karel, C. (1963). An algorithm for the traveling salesman problem. *Oper Res*, 11:972–989.

Lokin, F. C. J. (1978). Procedures for traveling salesman problems with additional constraints. *Eur J Oper Res*, 3:135–141.

Lu, Q. and Dessouky, M. M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transport Sci*, 38:503–514.

Lu, Q. and Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *Eur J Oper Res*, 175:672–687.

Madsen, O. B. G., Ravn, H. F., and Rygaard, J. M. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Ann Oper Res*, 60:193–208.

Mageean, J. and Nelson, J. D. (2003). The evaluation of demand responsive transport services in Europe. *J Transport Geogr*, 11:255–270.

Malca, F. and Semet, F. (2004). A tabu search algorithm for a dynamic pickup and delivery vehicle routing problem. In *Triennal Symposium on Transportation Analysis, Le Gosier, Guadeloupe, France, juin.*

Melachrinoudis, E., Ilhan, A. B., and Min, H. (2007). A dial-a-ride problem for client transportation in a health-care organization. *Comput Oper Res*, 34:742–759.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pickup points. *Transport Res A-Pol*, 23:377–386.

Min, H., Current, J., and Schilling, D. (1992). The multiple depot vehicle routing problem with backhauling. *J Bus Log*, 13:259–288.

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

Mitrović-Minić, S. (1998). Pickup and delivery problem with time windows: A survey. Technical Report SFU CMPT TR 1998-12, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

Mitrović-Minić, S., Krishnamurti, R., and Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transport Res B-Meth*, 38:669–685.

Mitrović-Minić, S. and Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transport Res B-Meth*, 38:635–655.

Mitrović-Minić, S. and Laporte, G. (2006). The pickup and delivery problem with time windows and transshipment. *INFOR*, 44:217–227.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Comput Oper Res*, 24:1097–1100.

Mosheiov, G. (1994). The traveling salesman problem with pickup and delivery. *Eur J Oper Res*, 79:299–310.

Mosheiov, G. (1998). Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. *Comput Ind Eng*, 34:669–684.

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.*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, pages 53–84, Philadelphia. SIAM.

Nagy, G. and Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *Eur J Oper Res*, 162(1):126–141.

Nanry, W. P. and Barnes, W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transport Res B-Meth*, 34:107–121.

Noda, I. (2005). Scalability of dial-a-ride systems: A case study to assess utilities of ubiquitous mass user support. In Ishida, T., Gasser, L., and Nakashima, H., editors, *Massively Multi-Agent Systems I: First International Workshop, MMAS 2004, Kyoto, Japan, December 10-11, 2004, Revised Selected and Invited Papers. Massively Multi-agent Systems in Public Space*, LNCS, pages 323–334. Springer.

Noda, I., Ohta, M., Shinoda, K., Kumad, Y., and Nakashima, H. (2003). Evaluation of usability of dial-a-ride systems by social simulation. In Hales, D., Edmonds, B., Norling, E., and Rouchier, J., editors, *Multi-Agent-Based Simulation III. MABS Techniques for Real World Modelling*, LNCS, pages 167–181. Springer.

Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking.* PhD thesis, Northwestern University, Evanston, IL.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Ann Oper Res*, 41:421–452.

Osman, I. H. and Wassan, N. A. (2002). A reactive tabu search metaheuristic for the vehicle routing problem with backhauls. *J Sched*, 5:263–285.

Österreichisches Rotes Kreuz (2006). Tätigkeitsbericht 2006. bilanz.roteskreuz.at.

Pacheco, J. and Martí, R. (2006). Tabu search for a multi-objective routing problem. *J Oper Res Soc*, 57:29–37.

Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev*, 33:60–100.

Palmer, K., Dessouky, M. M., and Abdelmaguid, T. (2004). Impacts of management practices and advanced technologies on demand responsive transit systems. *Transport Res A-Pol*, 38:495–509.

Pankratz, G. (2005). A grouping genetic algorithm for the pickup and delivery problem with time windows. *OR Spectrum*, 27:21–41.

Paquette, J., Cordeau, J.-F., and Laporte, G. (2007). Quality of service in dial-a-ride operations. *Comput & Ind Eng.* to appear.

Parragh, S. N., Cordeau, J.-F., Doerner, K. F., and Hartl, R. F. (2009a). Algorithms for the heterogeneous dial-a-ride problem with driver related constrainst. Technical report, University of Vienna.

Parragh, S. N., Doerner, K. F., Gandibleux, X., and Hartl, R. F. (2009b). A heuristic two-phase solution method for the multi-objective dial-a-ride problem. *Networks*. accepted.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008a). A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *J Betriebswirtschaft*, 58:21–51.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008b). A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *J Betriebswirtschaft*, 58:81–117.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2009c). Variable neighborhood search for the dial-a-ride problem. Technical report, University of Vienna.

Pasia, J. M., Doerner, K. F., Hartl, R. F., and Reimann, M. (2007a). A population-based local search for solving a bi-objective vehicle routing problem. In *Evolutionary Computation in Combinatorial Optimization, Proc. of EvoCOP 2007, Valencia, Spain, April 11-13, 2007*, LNCS, pages 166–175, Heidelberg-Berlin. Springer.

Pasia, J. M., Doerner, K. F., Hartl, R. F., and Reimann, M. (2007b). Solving a bi-objective vehicle routing problem by pareto-ant colony optimization. In Stützle, T., Birattari, M., and Hoos, H. H., editors, *Proc. of SLS 2007. Engineering Stochastic Local Search Algorithms. 6-8 September 2007, Brussels, Belgium*, LNCS, pages 187–191, Heidelberg-Berlin. Springer.

Pasia, J. M., Gandibleux, X., Doerner, K. F., and Hartl, R. F. (2007c). Local search guided by path relinking and heuristic bounds. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Evolutionary Multi-Criterion Optimization, Proc. of the 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007*, LNCS, pages 501–515, Heidelberg-Berlin. Springer.

Pasia, J. M., Hartl, R. F., and Doerner, K. F. (2006). Solving a bi-objective flowshop scheduling problem by using pareto ant colony optimization. In Dorigo, M., Gambardella, L. M., Birattari, M., Martinoli, A., Poli, R., and Stützle, T., editors, *Ant Colony Optimization*

and Swarm Intelligence 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006. Proc., LNCS, pages 294–305, Heidelberg-Berlin. Springer.

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

Popken, D. A. (2006). Controlling order circuity in pickup and delivery problems. *Transport Res E-Log*, 42:431–443.

Potvin, J.-Y., Duhamel, C., and Guertin, F. (1996). A genetic algorithm for vehicle routing with backhauling. *Appl Intell*, 6:345–355.

Potvin, J. Y. and Guertin, F. (1996). The clustered traveling salesman problem: A genetic approach. In Osman, I. H. and Kelly, J. P., editors, *Meta-heuristics theory and applications*, pages 619–631, Boston. Kluwer.

Potvin, J.-Y. and Rousseau, J.-M. (1992). Constrained-directed search for the advance request dial-a-ride problem with service quality constraints. In Balci, O., Shrada, R., and Zenios, Z. A., editors, *Computer Science and Operations Research: New Developments in their Interfaces*, pages 457–574, Oxford. Pergamon Press.

Potvin, J.-Y. and Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *J Oper Res Soc*, 46:1433–1446.

Potvin, J.-Y., Shen, Y., Dufour, G., and Rousseau, J.-M. (1995). Learning techniques for an expert vehicle dispatching system. *Expert Systems with Applications*, 8:101–109.

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

Przybylski, A., Gandibleux, X., and Ehrgott, M. (2008). Two phase algorithms for the bi-objective assignment problem. *Eur J Oper Res*, 185:509–533.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transport Sci*, 14:130–154.

Psaraftis, H. N. (1983a). Analysis of an O$|n^2|$ heuristic for the single vehicle many-to-many Euclidean dial-a-ride problem. *Transport Res B-Meth*, 17:133–145.

Psaraftis, H. N. (1983b). An exact algorithm for the single vehicle many to many dial-a-ride problem with time windows. *Transport Sci*, 17:351–357.

Psaraftis, H. N. (1983c). k-interchange procedures for local search in a precedence-constrained routing problem. *Eur J Oper Res*, 13:391–402.

Psaraftis, H. N. (1986). Scheduling large-scale advance-request dial-a-ride systems. *Am J Math Manag Sci*, 6:327–367.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. In Golden, B. L. and Assad, A. A., editors, *Vehicle Routing: Methods and Studies*, pages 223–248, Amsterdam. Elsevier (North-Holland).

Rebibo, K. K. (1974). A computer controlled dial-a-ride system. traffic control and transportation systems. In *Proceedings of 2nd IFAC/IFIP/IFORS Symposium Monte Carlo, September 1974*, Amsterdam. North-Holland.

Reimann, M., Doerner, K., and Hartl, R. F. (2002). Insertion based ants for vehicle routing problems with backhauls and time windows. In Dorigo, M., Di Caro, G., and Sampels, M., editors, *Ant Algorithms : Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002*, pages 135–148, Heidelberg-Berlin. Springer.

Reimann, M. and Ulrich, H. (2006). Comparing backhauling strategies in vehicle routing using ant colony optimization. *Cent Eur J Oper Res*. (to appear).

Reinelt, G. (1991). TSPLIB-A traveling salesman problem library. *ORSA J Comput*, 3:376–384.

Rekiek, B., Delchambre, A., and Saleh, H. A. (2006). Handicapped person transportation: An application of the grouping genetic algorithm. *Eng Appl Artif Intel*, 19:511–520.

Renaud, J., Boctor, F. F., and Laporte, G. (1996). A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS J Comput*, 8:134–143.

Renaud, J., Boctor, F. F., and Laporte, G. (2002). Perturbation heuristics for the pickup and delivery traveling salesman problem. *Comput Oper Res*, 29(9):1129–1141.

Renaud, J., Boctor, F. F., and Ouenniche, J. (2000). A heuristic for the pickup and delivery traveling salesman problem. *Comput Oper Res*, 27(9):905–916.

Resende, M. G. C. and Ribeiro, C. C. (2003). A GRASP with path relinking for private virtual circuit routing. *Networks*, 41:104–114.

Ropke, S. (2005). *Heuristic and exact algorithms for vehicle routing problems*. PhD thesis, Computer science department at the University of Copenhagen (DIKU).

Ropke, S. and Cordeau, J.-F. (2008). Branch-and-cut-and-price for the pickup and delivery problem with time windows. *Transport Sci*. forthcoming.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49:258–272.

Ropke, S. and Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transport Sci*, 40:455–472.

Ropke, S. and Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *Eur J Oper Res*, 171:750–775.

Roy, S., Rousseau, J., Lapalme, G., and Ferland, J. (1985a). Routing and scheduling for the transportation of disabled persons : The algorithm. Technical Report TP 5596E, CRT, Montréal, Canada.

Roy, S., Rousseau, J. M., Lapalme, G., and Ferland, J. (1985b). Routing and scheduling for the transportation of disabled persons : The tests. Technical Report TP 5598E, CRT, Montréal, Canada.

Ruland, K. S. and Rodin, E. Y. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Comput Math Appl*, 33:1–13.

Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *J Oper Res Soc*, 50:1034–1042.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA J Comput*, 4:146–154.

Savelsbergh, M. W. P. and Sol, M. (1995). The general pickup and delivery problem. *Transport Sci*, 29:17–29.

Savelsbergh, M. W. P. and Sol, M. (1998). DRIVE: Dynamic routing of independent vehicles. *Oper Res*, 46:474–490.

Schaffer, J. D. (1988). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications, 1985, Pittsburgh, PA*, pages 93–100, Hillsdale/New Jersey. Lawrence Erlbaum Associates.

Schönberger, J., Kopfer, H., and Mattfeld, D. C. (2003). A combined approach to solve the pickup and delivery selection problem. In Leopold-Wildburger, U., Rendl, F., and Wäscher, G., editors, *Operations Research Proceedings 2002*, pages 150–155, Berlin-Heidelberg-New York. Springer.

Semet, F. and Laporte, G. (2002). Classical heuristics for the capacitated vrp. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem.*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, pages 109–128, Philadelphia. SIAM.

Sexton, T. and Bodin, L. D. (1985a). Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. *Transport Sci*, 19:378–410.

Sexton, T. and Bodin, L. D. (1985b). Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing. *Transport Sci*, 19:411–435.

Sexton, T. R. and Choi, Y.-M. (1986). Pickup and delivery of partial loads with "soft" time windows. *Am J Math Manag Sci*, 6:369–398.

Shang, J. S. and Cuff, C. K. (1996). Multicriteria pickup and delivery problem with transfer opportunity. *Comput Ind Eng*, 30:631–645.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*.

Shen, Y., Potvin, J.-Y., Rousseau, J.-M., and Roy, S. (1995). A computer assistant for vehicle dispatching with learning capabilities. *Ann Oper Res*, 61:189–211.

Shinoda, K., Noda, I., Ohta, M., Kumada, Y., and Nakashima, H. (2003). Is dial-a-ride bus reasonable in large scale towns? Evaluation of usability of dial-a-ride systems by simulation. In Kurumatani, K., Chen, S.-H., and Ohuchi, A., editors, *Multiagent for Mass User Support: First International Workshop, MAMUS-03 Acapulco, Mexico, August 2003*, LNCS, pages 105–119. Springer.

Sigurd, M., Pisinger, D., and Sig, M. (2004). Scheduling transportation of live animals to avoid the spread of diseases. *Transport Sci*, 38:197–209.

Solomon, M. (1987). Algorithms for the vehicle routing problem with time windows. *Oper Res*, 35:254–265.

Sörensen, K. (2007). Distance measures based on the edit distance for permutation type representations. *J Heuristics*, 13:35–47.

Stein, D. M. (1978a). An asymptotic probabilistic analysis of a routing problem. *Math Oper Res*, 3:89–101.

Stein, D. M. (1978b). Scheduling dial-a-ride transportation systems. *Transport Sci*, 12:232–249.

Stummer, C. (1998). *Projektauswahl im betrieblichen F&E-Management*. Gabler, Deutscher Universitäts-Verlag, Wiesbaden.

Süral, H. and Bookbinder, J. H. (2003). The single-vehicle routing problem with unrestricted backhauls. *Networks*, 41:127–136.

Swihart, M. R. and Papstavrou, J. D. (1999). A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *Eur J Oper Res*, 114:447–464.

Tang Montané, F. A. and Galvão, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Comput Oper Res*, 33:595–619.

Teixeira, D. B. and Karash, K. H. (1975). An evaluation of councils on aging dial-a-ride systems in Massachusetts. *Transportation*, 4:105–121.

Teodorovic, D. and Radivojevic, G. (2000). A fuzzy logic approach to dynamic dial-a-ride problem. *Fuzzy Set Sys*, 116:23–33.

Thangiah, S. R. and Awan, A. (2006). Real-time split-delivery pickup and delivery time window problems with transfer. Technical Report SRT90-2006, Artificial Intelligence Robotics Lab, Slippery Rock University, PA.

Thangiah, S. R., Potvin, J.-Y., and Sun, T. (1996). Heuristic approaches to vehicle routing with backhauls and time windows. *Comput Oper Res*, 23:1043–1057.

Toth, P. and Vigo, D. (1996a). Fast local search algorithms for the handicapped persons transportation problem. In Osman, I. H. and Kelly, J. P., editors, *Metaheuristics: Theory and Applications*, pages 677–690, Boston, MA. Kluwer.

Toth, P. and Vigo, D. (1996b). A heuristic algorithm for the vehicle routing problem with backhauls. In Bianco, L. and Thot, P., editors, *Advanced Methods in Transportation Analysis*, pages 585–608, Berlin. Springer.

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

Toth, P. and Vigo, D. (1997b). Heuristic algorithms for the handicapped persons transportation problem. *Transport Sci*, 31:60–71.

Toth, P. and Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problem with backhauls. *Eur J Oper Res*, 113:528–543.

Toth, P. and Vigo, D. (2002). VRP with backhauls. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, pages 195–224, Philadelphia, PA. SIAM.

Tzoreff, T. E., Granot, D., Granot, F., and Sosic, G. (2002). The vehicle routing problem with pickups and deliveries on some special graphs. *Discrete Appl Math*, 116:193–229.

Černy, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *J Opt Theory Appl*, 45:41–51.

Uchimura, K., Saitoh, T., and Takahashi, H. (1999). The dial-a-ride problem in a public transit system. *Electron Commun Jpn*, 82:30–38.

Ulungu, E. L. (1993). *Optimisation combinatoire multicritère: Détermination de l'ensemble des solutions efficaces et méthodes interactives*. PhD thesis, Faculté des Sciences, Université de Mons-Hainaut, Mons (Belgique).

Ulungu, E. L. and Teghem, J. (1992). Heuristic for multi-objective combinatorial optimization problems with simulated annealing. Presented at the EURO XII Conference, Helsinki.

Ulungu, E. L. and Teghem, J. (1995). The two phase method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20:149–165.

van der Bruggen, L. J. J., Lenstra, J. K., and Schuur, P. C. (1993). Variable-depth search for the single vehicle pickup and delivery problem with time windows. *Transport Sci*, 27:298–311.

Van Hentenryck, P. and Bent, R. (2006). *Online stochastic combinatorial optimization*. MIT Press, Cambridge, MA.

Vincke, P. (1992). *Multicriteria decision-aid*. Wiley, Chichester.

Wade, A. and Salhi, S. (2004). An ant system algorithm for the mixed vehicle routing problem with backhauls. In *Metaheuristics: computer decision-making*, pages 699–719, Norwell, MA, USA. Kluwer Academic Publishers.

Wade, A. C. and Salhi, S. (2002). An investigation into a new class of vehicle routing problem with backhauls. *Omega*, 30:479–487.

Wang, F., Lim, A., and Xu, Z. (2006). The one-commodity pickup and delivery travelling salesman problem on a path or a tree. *Networks*, 48:24–35.

Wilson, H. and Colvin, N. (1977). Computer control of the Rochester dial-a-ride system. Technical Report R-77-31, Department of Civil Engineering. MIT Cambridge, MA.

Wilson, H., Sussman, J., Wang, H., and Higonnet, B. (1971). Scheduling algorithms for dial-a-ride system. Technical Report USL TR-70-13, Urban Systms Laboratory, MIT, Cambridge, MA.

Wilson, H. and Weissberg, H. (1967). Advanced dial-a-ride algorithms research project: Final report. Technical Report R76-20, Department of Civil Engineering. MIT, Cambridge, MA.

Winston, W. L. (1994). *Operations research: applications and algorithms*. Duxbury Press, Belmont, CA, 3rd edition.

Wolfler Calvo, R. and Colorni, A. (2007). An effective and fast heuristic for the dial-a-ride problem. *4OR*, 5:61–73.

Wong, K. I. and Bell, M. G. H. (2006). Solution of the dial-a-ride problme with multi-dimensional capacity constraints. *Int Trans Oper Res*, 13:195–208.

Xiang, Z., Chu, C., and Chen, H. (2006). A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *Eur J Oper Res*, 174:1117–1139.

Xiang, Z., Chu, C., and Chen, H. (2007). The study of a dynamic dial-a-ride problem under time dependent stochastic environments. *Eur J Oper Res*. (to appear, available online).

Xu, H., Chen, Z.-L., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transport Sci*, 37:347–364.

Yano, C. A., Chan, T. J., Richter, L. K., Cutler, T., Murty, K. G., and McGettigan, D. (1987). Vehicle routing at quality stores. *Interfaces*, 17:52–63.

Yi, F. and Tian, L. (2005). On the online dial-a-ride problem with time-windows. In Megiddo, N., Xu, Y., and Zhu, B., editors, *Algorithmic Applications in Management: First International Conference, AAIM 2005, Xian, China, June 22-25, 2005. Proceedings*, LNCS, pages 85–94. Springer.

Zhong, Y. and Cole, M. H. (2005). A vehicle routing problem with backhauls and time windows: A guided local search solution. *Transport Res E-Log*, 41:131–144.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, K. C., Tsahalis, D. T., Périau, J., Papailiou, K. D., and Fogarty, T., editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece. International Center for Numerical Methods in Engineering (Cmine).

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE T Evolut Comput*, 3:257–271.

Zitzler, E., Thiele, L., Laumanns, C., Fonseca, M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE T Evolut Comput*, 7:117–132.

Abstract

Humanitarian not-for-profit ambulance dispatching organizations are committed to look at cost reduction potentials in order to decrease their expenses. While in the context of emergency transportation cost reduction cannot be achieved by means of combined passenger routes, this can be done when dealing with regular patients. This research work is motivated by the problem situation faced by ambulance dispatchers, such as the Austrian Red Cross, in the field of patient transportation.

Ambulance routing problems belong to the large class of vehicle routing problems involving pickups as well as deliveries. While standard pickup and delivery problems deal with the transportation of goods, in ambulance routing, people are subject to transportation. Problems of this kind are modeled as dial-a-ride problems. In the field of passenger or patient transportation, the provision of a certain quality of service is necessary; the term "user inconvenience" is used in this context. Low user inconvenience relates to punctual service and short individual ride times. A certain trade-off between user inconvenience and total operating costs can be observed. Lower user inconvenience usually entails higher operating costs and vice versa. User inconvenience can either be considered in terms of additional constraints or in terms of additional objectives. Both approaches are investigated in this book. The aim is to model and solve the real world problem based on available information from the Austrian Red Cross.

In a first step, a competitive solution method based on variable neighborhood search for a simplified problem version is developed. A vehicle fleet of fixed size, time windows, maximum user ride times, and a route duration limit are among the constraints considered.

In a second step, besides routing costs, a user-oriented objective, minimizing user inconvenience, in terms of mean user ride time, is introduced. An exact and a heuristic solution method are devised. The heuristic solution method integrates variable neighborhood search and path relinking in a two-phase scheme. The exact method iteratively solves single objective problems to optimality within a so-called ϵ-constraint framework. Both procedures provide the ambulance dispatcher with a number of efficient transportation plans in a Pareto multi-objective context. The term efficient implies here that neither is better than any other transportation plan in both objectives. The different plans are thus incomparable across each other and the dispatcher's choice will depend on whether more emphasis has to be put on costs or on user inconvenience.

In a third step, heterogeneous patient types and a heterogeneous vehicle fleet are intro-

duced into the standard dial-a-ride model. A 3-index and a 2-index mathematical problem formulation are proposed. Each of them serves as the basis for a branch and cut algorithm. Comparison shows that the 2-index formulation based algorithm clearly outperforms the 3-index based version. The previously developed variable neighborhood search method is also adapted. It proves to be flexible enough to accommodate the given real world constraints; transportation plans of high quality can be computed very quickly.

In a final step, besides heterogeneous patient types and vehicles, staff related conditions are integrated into a 3-index problem formulation. These refer to the assignment of drivers and other staff members to vehicles, and to the scheduling of lunch breaks and additional stops at the depot. A column generation algorithm is devised to compute lower bounds. These bounds serve to assess the solution quality of the proposed variable neighborhood search heuristic. In this case, further adaptations are necessary in order to accommodate the different additional real-world characteristics. The resulting method computes high quality solutions for adapted benchmark data sets; also much larger real world instances are solved.

All exact methods employed are not capable of solving instances of realistic size. This fact makes the development of according heuristic and metaheuristic solution methods necessary. In this book a rather generic variable neighborhood search framework is proposed. It is able to accommodate all single-objective problem versions and also proves to work well when applied to the bi-objective problem in combination with path relinking.

Zusammenfassung

Humanitäre non-profit Organisationen im Bereich des Patiententransports sehen sich dazu verpflichtet alle möglichen Einsparungs- und Optimierungspotentiale auszuloten um ihre Ausgaben zu reduzieren. Im Gegensatz zu Notfalleinsatzfahrten, bei denen ein Zusammenlegen mehrerer Transportaufträge normalerweise nicht möglich ist, besteht bei regulären Patiententransporten durchaus Einsparungspotential. Diese Tatsache gibt Anlass zur wissenschaftlichen Analyse jener Problemstellung, welche die täglich notwendige Planung regulärer Patiententransportaufträge umfasst.

Patiententransportprobleme gehören zur Klasse der *pickup and delivery* Probleme. Im Unterschied zu Standardproblemen dieser Klasse, werden in diesem Fall nicht Güter sondern Personen befördert. Solche Aufgabenstellungen werden als *dial-a-ride* Probleme modelliert. Im Bereich des Personen- und Patiententransports ist es notwendig den BenutzerInnen eine angemessene Service-Qualität zu bieten. Pünktliches Service und kurze Fahrtzeiten wirken sich günstig auf die Service-Qualität aus, während größere Unannehmlichkeiten in Bezug auf lange Warte- oder auch Wegzeiten negativ in die wahrgenommene Service-Qualität einfließen. Zwischen den verursachten systemweiten Kosten und den in Kauf zu nehmenden Unannehmlichkeiten herrscht eine direkte Wechselbeziehung. Eine Reduktion der Unannehmlichkeiten für die BenutzerInnen führt zu höheren Kosten des Dienstleisters und umgekehrt. Diese Unannehmlichkeiten können entweder durch Nebenbedingungen auf ein akzeptables Maß beschränkt oder in einer zusätzlichen Zielfunktion minimiert werden. Beide Herangehensweisen werden in diesem Buch untersucht. Das Ziel dieser Arbeit ist die Modellierung und Lösung des realen Problems, basierend auf entsprechenden Informationen des Österreichischen Roten Kreuzes.

In einem ersten Schritt wird eine kompetitive Lösungmethode, basierend auf dem *variable neighborhood search* Konzept für eine vereinfachte Problemstellung aus der Literatur entwickelt. Unter anderem werden eine Fahrzeugflotte gegebener Größe, Zeitfenster, maximale Fahrtzeiten für die BenutzerInnen sowie maximale Gesamtzeiten pro Tour berücksichtigt. Diese Standardproblemstellung wird in zwei Richtungen erweitert.

Zuerst wird, zusätzlich zur Minimierung der Gesamtkosten, eine zweite Zielfunktion eingeführt, welche Unannehmlichkeiten hinsichtlich der durchschnittlich im Fahrzeug verbrachten Zeit minimiert. Die resultierende Problemstellung wird sowohl exakt als auch heuristisch gelöst. Die heuristische Lösungsmethode integriert *variable neighborhood search* und *path relinking* in ein zweiphasiges Programm. Die exakte Methode löst das Einziel-

problem iterativ nach dem sogenannten ϵ-*constraint* Schema. Die entwickelten Methoden generieren eine Reihe von effizienten Transportplänen. Effizient bedeutet hier, dass keiner von ihnen in beiden Zielen besser ist als die anderen Transportpläne. Die einzelnen Transportpläne sind somit untereinander unvergleichbar und es obliegt dem Disponenten den jeweils passenden auszuwählen, in Abhängigkeit davon, ob die verursachten Kosten oder die Service-Qualität stärker im Vordergrund stehen soll.

Sodann werden eine heterogene Fahrzeugflotte und unterschiedliche Patiententypen in die Standardproblemstellung integriert. Sowohl eine 3-Index- als auch eine 2-Index-Formulierung werden eingeführt. Sie dienen als Grundlage für jeweils einen *branch and cut* Algorithmus. Vergleicht man die Ergebnisse, führt die 2-Index-Formulierung eindeutig zu den besseren Resultaten. Auch die entwickelte *variable neighborhood search* Heuristik wird an die neue Problemstellung angepasst. Sie erweist sich als ausreichend flexibel auch die nun gegebenen realitätsnahen Anforderungen zu berücksichtigen; die adaptierte Methode generiert Transportpläne von hoher Qualität innerhalb sehr kurzer Laufzeit.

In einem letzten Schritt werden schließlich auch Personaleinsatzaspekte in eine 3-Index-Problemformulierung integriert. Zusätzlich zu heterogenen Fahrzeugen und PatientInnen wird nun die Zuordnung von Fahrern und sonstigem Personal zu den unterschiedlichen Fahrzeugen sowie Mittagspausen und Aufenthalte am Depot berücksichtigt. Ein *column generation* Algorithmus wird zur Berechnung von unteren Schranken implementiert, welche der Qualitätsmessung für die entwickelte *variable neighborhood search* Methode dienen. In diesem Fall sind weitreichendere Adaptierungsmaßnahmen von Nöten um die gegebenen zusätzlichen realitätsnahen Aspekte zu berücksichtigen. Die resultierende Methode erstellt Transportpläne von hoher Qualität für adaptierte Standardinstanzen. Auch um vieles größere Echtweltinstanzen werden gelöst.

Die eingesetzten exakten Methoden können Instanzen von realistischer Größe nicht lösen. Dieser Umstand macht die Entwicklung von passenden heuristischen Verfahren nach wie vor unumgänglich. In der vorliegenden Arbeit wird ein relativ generisches System basierend auf der *variable neighborhood search* Idee entwickelt, das auf alle behandelten Einzielproblemversionen angewandt werden kann; auch für die bi-kriterielle Problemstellung, in Kombination mit *path relinking*, werden gute Ergebnisse erzielt.

Curriculum vitae

Personal data

name	Sophie Parragh
date of birth	April 8, 1981
citizenship	Austria

Education

01/2000–08/2000	**International Language School** *USIU* San Diego, California
10/2000–10/2005	**Study** International Business Administration *University of Vienna* Specialization in operations research and logistics
09/2002–06/2003	**Academic year abroad** *Universite Paul Sabatier* Toulouse, France
2005	**Master degree (Mag. rer. soc. oec.)** *University of Vienna* Master thesis title: Heuristic optimization for a stochastic bi-objective location problem
since 03/2006	**PhD program** PhD Management, *University of Vienna* Thesis title: Ambulance routing problems with rich constraints and multiple objectives

Professional activities

10/2004–02/2005	**Tutor** *University of Vienna* Discrete mathematics and graph theory Software applications in operations research
since 11/2005	**Research project assistant** *University of Vienna* Chair for Production and Operations Management
08/2008–10/2008	**Visiting student** *Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT)* Montreal, Canada

Publications

2009 S. N. Parragh, J.-F. Cordeau, K. F. Doerner, R. F. Hartl. *Algorithms for the heterogeneous dial-a-ride problem with driver related constraints*. Technical report, University of Vienna, 2009.

S. N. Parragh, K. F. Doerner, R. F. Hartl. *Variable neighborhood search for the the dial-a-ride problem*. Technical report, University of Vienna, 2009. submitted.

S. N. Parragh, K. F. Doerner, X. Gandibleux, and R. F. Hartl. *A heuristic two-phase solution method for the multi-objective dial-a-ride problem*. Networks, 2009. accepted.

2008 S. N. Parragh, K. F. Doerner, R. F. Hartl. *A survey on pickup and delivery problems. Part I: Transportation between customers and depot.* J Betriebswirtschaft, 58:21–51,2008.

S. N. Parragh, K. F. Doerner, R. F. Hartl. *A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations.* J Betriebswirtschaft, 58:81–117, 2008.

Presentations

2008 *Algorithms for the heterogeneous dial-a-ride problem.*
S. N. Parragh, J.-F. Cordeau, K. F. Doerner, R. F. Hartl.
CIRRELT, research seminar, Montreal, Canada, October 2008.

Algorithms for the heterogeneous dial-a-ride problem.
S. N. Parragh, J.-F. Cordeau, K. F. Doerner, R. F. Hartl.
Entscheidungsunterstützung in der Logistik, Salzburg, Austria, April 2008.

2007 *Metaheuristics for the multi-objective dial-a-ride problem.*
S. N. Parragh, K. F. Doerner, X. Gandibleux, R. F. Hartl.
MIC 2007, Montreal, Canada, June 2007.

Approaching the Pareto frontier of the multi-objective dial-a-ride problem.
S. N. Parragh, K. F. Doerner, X. Gandibleux, R. F. Hartl.
Route 2007, Jekyll Island, Georgia, May 2007.

Variable neighborhood search for the dial-a-ride problem.
S. N. Parragh, K. F. Doerner, R. F. Hartl.
LINA, research seminar, Nantes, France, February 2007.

Teaching

operations management — *University of Vienna*, summer term 2007
Content: basic principles, models and (heuristic) solution methods in graph theory, transportation network design, layout planing, assembly line balancing, forecasting, scheduling.

implementation of optimization methods — *University of Vienna*, since 2007
Content: introduction to C++, implementation of basic heuristic algorithms for transportation problems.