universität wien

DISSERTATION

Titel der Dissertation

Vehicle routing problems with service consistency considerations

verfasst von

Mag. Attila A. Kovacs

angestrebter akademischer Grad

Doctor of Philosophy (PhD)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 094 151 403
Dissertationsgebiet lt. Studienblatt:	Betriebswirtschaft (Logistics and Operations Management)
Betreut von:	o. Univ.-Prof. Dipl.-Ing. Dr. Richard F. Hartl

I dedicate this thesis to Viktoria

Acknowledgements

Throughout my time as a PhD student I have been supported by many great individuals. I am especially grateful to four of them:

I have been more than lucky to have Richard Hartl as my advisor. Prof. Hartl endeavors to create an atmosphere of support and creativity and his door is always open. In my research, he let me be guided by my curiosity and paved the way for finishing my PhD successfully. Prof. Hartl always knew how to bring the best out in our research papers.

Sophie Parragh is smart and hardworking, and she has played an important role in completing this thesis. Sophie is probably the reason I started doing research in the first place. She always trusted my abilities and supported me however she could.

I met Bruce Golden at a conference in Spain where he invited me to visit him at the University of Maryland. He treated me as one of his own PhD students, making time for me on a daily basis, and helping me find my way around. I had a great time in Maryland and learned a lot about the essence of research. Prof. Golden helped me to improve this thesis significantly.

I owe many thanks to Zsuzsanna Scheidl. She celebrated with me in good times (e.g., when a paper was accepted for publication) and stood by my side in bad times (e.g., when reviews were challenging). She helped me organize my thoughts each time I missed the forest for the trees. Not least, she proof-read this book several times. It would have been much harder to make it this far without Zsuzsanna.

Preface

This dissertation has been prepared over several years and represents the result of my own work. Many contents of this book are already published in whole or in part in following scientific journals: Chapter 2 has been accepted for publication in form of a survey paper in Kovacs et al. [104]; the papers Kovacs et al. [107] and Kovacs et al. [103] are based on Chapters 3 and 4, respectively; a technical report submitted for publication (Kovacs et al. [106]) is based on Chapter 5.

Attila A. Kovacs

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

An increasing number of companies focus on customer satisfaction in order to increase the lifetime value of each customer. Satisfied customers are loyal, remain with the company for a long time, and generate recurring revenues. Customer-first business strategies have been acknowledged in the vehicle routing literature early on. However, customer satisfaction has been quantified explicitly only in recent papers. In vehicle routing, customer satisfaction is often a result of consistent service. Customers appreciate service at regular times of the day provided by the same driver each time. Additionally, drivers become more familiar with their tasks if they visit the same customers and service regions repeatedly.

In Chapter 2, we survey literature that addresses service consistency in vehicle routing. We present early solution approaches, starting from the 1970's, that focus on reducing the operational complexity resulting from planning and executing new routes each day. One side benefit of these approaches is service consistency; therefore, many recent solution approaches devised for improving customer satisfaction are based on previous achievements. We classify the literature according to three consistency features: arrival time consistency, person-oriented consistency, and delivery consistency. For each feature, we survey different modeling concepts and measurements, demonstrate solution approaches, and examine the increase in cost of improving service consistency.

The consistent vehicle routing problem (ConVRP) combines traditional vehicle routing constraints with the requirements for service consistency. Consistency is achieved by assigning one driver to a customer and by bounding the variation in the arrival times over a given planning horizon. In Chapter 3, we present a fast solution method called template-based adaptive large neighborhood search for the ConVRP. Compared to state-of-the-art heuristics, the developed algorithm is highly competitive on the available benchmark instances. By varying different model parameters in new test instances, we can observe inter-

esting effects on the trade-off between cost and service consistency. Additionally, a relaxed variant of the original ConVRP is presented. In this variant, we adjust the departure times from the depot in order to improve arrival time consistency. The routing cost is considerably reduced by flexible departure times when consistency requirements are tight.

The requirements in the ConVRP may be too restrictive in some applications. In the generalized ConVRP (GenConVRP), presented in Chapter 4, each customer is visited by a limited number of drivers and the variation in the arrival times is penalized in the objective function. The vehicle departure times may be adjusted to obtain stable arrival times. Additionally, customers are associated with AM/PM time windows. In contrast to previous work on the ConVRP, we do not use the template concept to generate routing plans. Our approach is based on a flexible large neighborhood search (LNS) that is applied to the entire solution. Several destroy and repair heuristics have been designed to remove customers from the routes and to reinsert them at better positions. Arrival time consistency is improved by a simple 2-opt operator that reverses parts of particular routes. A computational study is performed on ConVRP benchmark instances and on new instances generated for the generalized problem. The proposed algorithm performs well on different variants of the ConVRP. It outperforms template-based approaches in terms of travel cost and time consistency. For the GenConVRP, we experiment with different input parameters and examine the trade-off between travel cost and customer satisfaction.

Typically, we have to accept an increase in routing cost in order to provide a higher level of service consistency, i.e., routing cost and service consistency are conflicting objectives. In Chapter 5, we extend the GenConVRP by considering several objective functions: improving driver consistency and arrival time consistency, and minimizing routing cost are independent objectives of the problem. We refer to the problem as the multi-objective generalized consistent vehicle routing problem (MOGenConVRP). A multi-objective optimization approach enables a thorough trade-off analysis between conflicting objective functions. The results of this chapter should help companies in finding adequate consistency goals to aim for. Results are generated for several test instances by two exact solution approaches and one heuristic. The exact approaches are based on the ε-constraint framework and are used to solve small test instances to optimality. Large instances are solved by multi directional large neighborhood search (MDLNS) that combines the multi directional local search framework and the LNS for the GenConVRP. The solution quality of the heuristic is evaluated by computing five multi-objective quality indicators. We find that MDLNS is an eligible solution approach for performing a meaningful trade-off analysis.

Chapter 2

Vehicle routing problems in which consistency considerations are important

2.1 Introduction

The ultimate objective of private companies is profitability. Only profitable companies can prevail against competition, develop further, and offer their products and services to customers successfully. Companies that are unable to cover their expenditures in the long run, will disappear from the market.

Several strategies exist for how companies can achieve their goals in competitive markets. These strategies differ by putting the primary emphasis either on the interests of the company or on the needs of the customers. One of the oldest strategies, still common today, is based on the idea that customers prefer products and services that are available easily and cheaply. Companies applying this strategy, concentrate on operating efficiently, decreasing cost, and achieving economies of scale. Low-priced, standard products or services are offered to a broad range of customers. The emphasis under this strategy is clearly on the company's interest.

In the last couple of decades, this strategy has dominated the field of vehicle routing. The primary objective function in most routing models is to minimize variable cost (e.g., travel cost), fixed cost (e.g., number of required vehicles), or both; the constraints place limits on operational decisions.

However, company-first strategies have proven to be inefficient in developed markets. With increasing (global) competition and increasingly demanding customers, companies are more likely to apply a strategy that is more "customer-first" (Keith [96], Kotler [102]). Companies applying this strategy do not strive for profit per se. Instead, they stimulate profit as a

consequence of satisfying customer needs better than their competitors (Kohli and Jaworski [100]). Here, innovative measurements of customer satisfaction are used in addition to traditional business performance measurements such as return on investment (Fornell [65]). The positive effect of customer-first business strategies on profit is described in Homburg et al. [89], Kohli and Jaworski [100], and Narver and Slater [124].

The relationships among profitability, customer satisfaction, employee satisfaction, and productivity are addressed in Heskett et al. [87] and Storbacka et al. [165]: First, profit is mostly driven by customer loyalty. Loyal customers generate recurring revenues. So, the longer a customer stays with a company, the more profit he generates. Reducing attrition by 5% increases profit between 25% and 85%, depending on the industry (Reichheld and Sasser [141]). This increase is composed of profits from referrals, profits from reduced operating cost, and profits from price premiums that loyal customers are willing to pay. Second, loyalty is a function of customer satisfaction. Customers are highly satisfied if the product or service received exceeds their expectations. At low and medium satisfaction levels, customers are prone to switch to competitors with a better offer. Highly satisfied customers form a bond with the company and stay with it over time. Storbacka et al. [165] note that legal, social, and economic bonds between the customer and the company prolong the relationship in addition to customer satisfaction. Third, satisfaction is a result of the service value perceived by the customers. The service value is the ratio between the benefit a customer would receive from a product or service (e.g., functional benefits plus emotional benefits) and the cost (e.g., cost in time and money). Accordingly, companies can increase the value of their service by increasing service quality or by decreasing cost to the customer. Fourth, service value is created by satisfied and productive employees. In the service industry, employees meet customers on a regular basis. They acquire useful knowledge about customer preferences and report their findings back to management. Typically, the turnover of dissatisfied employees with years of customer relationship experience results in significant loss of productivity and decreased customer satisfaction. Finally, employee satisfaction increases with the employee's ability and authority to meet the needs of customers. Kohli and Jaworski [100] find that employees are proud of belonging to a company whose focus is on serving customers. Frey et al. [69] and Reichheld and Sasser [141] report that low employee turnover is linked closely to high customer satisfaction.

The chain from employee satisfaction to customer satisfaction to profitability, referred to as the service-profit chain, is illustrated in Figure 2.1.

Many companies focus on customers and employees as a key part of their management philosophy. These companies have realized that competing on price alone is not effective

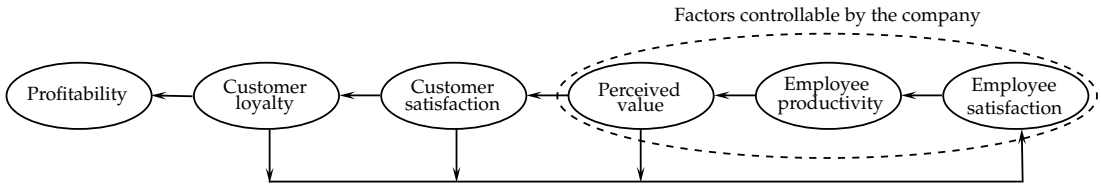


Figure 2.1 Service-profit chain: relationships between profitability, customer satisfaction, employee satisfaction, and productivity (adapted from Heskett et al. [87]).

(Fornell [65]). As a consequence, they spend a lot of effort on improving service quality and customer satisfaction. Highly satisfied customers stay longer with the company, pay less attention to competitors, cost less to serve, are less sensitive to price changes, and have positive things to say about the company. Yet, before investing in customers and employees, companies have to quantify the cost and benefit of this customer-first strategy.

Compared to other research areas, e.g., marketing, the transition from company-first to customer-first management strategies has received little attention in vehicle routing. In Section 2.2, we survey application areas of routing models in which customer satisfaction has been found to be important. In these applications, service consistency is the key characteristic of high quality service and, therefore, of high customer satisfaction. The main reason for inconsistency in routing plans is fluctuating input parameters such as changing customer demand and random customer orders (see Section 2.3). Solution approaches for coping with varying input parameters have been presented in early articles (e.g., Beasley [4], Christofides [29], Golden and Stewart [74], Hardy [79], Jaillet [92], Wong and Beasley [184]). However, the focus has been on reducing the effort of adjusting routing plans on a daily basis rather than on increasing customer or employee satisfaction. In Section 2.4, we give an overview of these approaches because they serve as a basis for many recent solution techniques that improve consistency explicitly. Service consistency can be achieved in three dimensions: consistency in the timing of the service, consistency in the person who provides a service (customer point of view) or consistency in the person who receives a service (employee point of view), and consistency in the quantity delivered or in each customer's inventory level (e.g., in inventory routing problems). In Section 2.5, we survey recent papers adopting at least one of these consistency features and examine how consistency is implemented. In addition, we describe how a variety of solution approaches are applied to solve vehicle routing problems in which consistency is considered. We also examine the effect of improved consistency on the resulting routing cost. There is a close relationship between routing problems in which consistency is important and problems in which consistency is

undesirable (e.g., money transporters are exposed to attacks if vehicle routes are entirely predictable). Therefore, in Section 2.6, we also review the dimensions of inconsistency and the relevant implementation aspects.

2.2 Routing for customer satisfaction

Many companies in different routing-related industries seek to provide high quality service in order to improve customer satisfaction. Here, high quality means being consistent over time: regular customers (i.e., customers that are visited frequently) want to be served by the same familiar person at roughly the same time; if a delivery consists of storable goods, customers want small variations in their inventory levels. Additionally, consistency reduces the requirements imposed on drivers to learn about new routes and new customers, i.e., consistency improves employee satisfaction and productivity.

We remark that customer convenience can also be achieved in other ways than by providing service consistency. For example, in the Vehicle Routing Problem with Time Windows (VRPTW), customers may be visited only if they are available (Cordeau et al. [38]). In the Dial-a-Ride Problem (DARP), each passenger is associated with a pickup and a delivery location. Maximum ride-time restrictions are considered in order to avoid customer inconvenience (Cordeau and Laporte [39], Parragh et al. [132]). In the Periodic Vehicle Routing Problem (PVRP), customers require a given number of visits during the planning horizon and specify a minimum and maximum spacing between two visits (Francis et al. [68], Campbell and Wilson [24]). Delivery timing, delivery quantity, and routing are determined simultaneously in the Inventory Routing Problem (IRP) (Bertazzi et al. [10], Campbell et al. [19], Coelho et al. [33]). The challenge is to find a replenishment plan with minimal cost that prevents customer stockouts.

In this thesis, we focus on service consistency and start with describing some application areas to illustrate the relevance of consistency in vehicle routing.

2.2.1 Consistency in small package shipping

Delivering parcels, packages, and letters has become an essential part of the transportation industry. Pharmaceuticals, medical supplies, electronic products, repair parts, legal documents, magazines, and packages from private households and organizations are transported from one location to another on a daily basis. The revenue generated by small package carriers exceeds the revenues of all major freight transportation modes (e.g., air, railroad, and

water way transportation) except trucking (Morlok et al. [119]). In 1997, 10% of the value of all goods and services produced in the U.S. (i.e., the gross domestic product) were delivered by package carriers. Since then, the small package shipping industry's importance to the economy has grown steadily. The process of shipping packages is the following: Drivers collect packages from (sending) customers, the packages are aggregated at an origin hub and sent to a destination hub via long-haul transportation or air, the packages are disaggregated and delivered to (receiving) customers. Collecting packages from and delivering them to customers is the most critical part of the supply chain. First, less-than-truckload transportation is more expensive than shipping full truckloads. Second, usually, this is the only opportunity for face-to-face contact between the company, represented by its drivers, and the customers. The competition in small package shipping is intense (The Wall Street Journal [171]); private (e.g., DHL, FedEx, UPS) and public companies (e.g., national postal services) have to operate at maximum efficiency to remain in the market. In order to differentiate themselves, companies offer a variety of transportation-related services. Additionally, many companies improve service quality by providing service consistency. Customers can choose between several delivery speeds, track the current location of their packages, and rely on punctual delivery. Additionally, many companies improve service quality by providing service consistency.

Wong [185] investigates the difference between the routing problem of package carriers and the classical vehicle routing problem. A major distinguishing characteristic of small package shipping is the requirement of service consistency. Regular customers are those who are visited routinely and they prefer to be visited at roughly the same time on each day. Driver consistency improves the driver-customer relationship and enables a customized service. After being awarded with the 2003 Best Retention Practices Award and being named to the list of Canada's 50 Best Managed Companies in 2002, Evan MacKinnon, CEO of MacKinnon Transport, explained his company's success as follows (Canadian Transportation & Logistics [25]): *One of the biggest advantages our driver retention provides our shipper clients is consistency of service - familiar faces showing up at their facilities. Our drivers get to know the shippers and their receivers and their company policies and safety requirements very well. And also because of their longevity with our company our drivers are better trained to handle a company's specific product requirements and the needs of their customers.* Typically, a larger routing cost due to a focus on consistency is more than offset by higher customer satisfaction and increased driver familiarity with the region and the customers.

Vehicle routing problems that consider the requirements of the small package shipping

industry are investigated, e.g., in Campbell and Thomas [23], Francis et al. [67], Groër et al. [75], Haughton [84], Smilowitz et al. [158], Sungur et al. [167], and Zhong et al. [190].

2.2.2 Consistency in health care

Improving health care is a major challenge for modern societies (Schneider et al. [153]). The number of care-dependent people (e.g., chronically ill, physically disabled, and the elderly) is increasing steadily. At the same time, financial constraints are getting tighter due to a shrinking working population.

One of the most important achievements in health care management has been the shift from stationary forms of health care (e.g., nursing homes) to home care. Home care service enables the patients to live at home with the greatest possible degree of autonomy while they are visited and supported by skilled staff. Typically, the service includes housekeeping, meals-on-wheels, and medical care. Additionally, significant public cost-savings are realized by shifting from stationary forms of health care to home care (Coyte and McKeever [40], Hollander et al. [88], Schneider et al. [153]): First, investments in care facilities decrease; second, lower investment requirements enable private home care providers to enter the market; third, the share of informal care provided by family and friends increases.

Because of increasing competition, budget cuts, and, more importantly, social service obligations, home care providers have to carefully balance between cost efficiency and service quality; this is true for private and public organizations. The significance of service consistency in home care is examined in Woodward et al. [186]. The results are based on interviews with managers, nurses, home workers, clients, and physicians. In order to provide high quality service, the workers need knowledge about the clients and their homes, and have to be trained for the required service. Furthermore, workers should be familiar with the likes and dislikes, the abilities and limitations, and routines of the clients. Because of the physical intimacy, the clients must feel comfortable with the service provider. Personnel consistency significantly improves service quality; it simplifies the communication between the persons involved and it reduces the time to review a client's history (Bennett Milburn [7], Bertels and Fahle [11], Borsani et al. [16], Eveborn et al. [58], Nowak et al. [129]).

With the expansion of home care services, there is also an increasing demand for customized door-to-door transportation of care-dependent people. Paquette et al. [131] investigate appropriate quality measurements for dial-a-ride services in paratransit. Based on an extensive customer survey, the authors identify driver consistency as one of the most effec-

tive criteria to improve service quality. Feillet et al. [59] focus on consistent time schedules, as clients of paratransit services are particularly sensitive to changes in their daily routines.

2.2.3 Consistency in other application areas

In vendor-managed inventory systems, the decision-making about the timing and the quantity of an order is transferred from the customers to the supplier. The supplier is more flexible in optimizing the replenishment process, but he also has a larger responsibility towards the customers. Day et al. [44] investigate the inventory replenishment problem of a company that delivers CO_2 gas for carbonated soft drinks to restaurants, movie theaters, hospitals, and many other businesses. Because of increased competition, the company wants to distinguish itself by superior customer service. This is achieved by increasing delivery regularity and by improving driver-customer familiarity. Coelho et al. [32] and Coelho et al. [33] confirm that focusing on cost in replenishment problems results in inconvenient solutions for suppliers and customers. The authors introduce consistency requirements into the inventory routing problem in order to decrease variations in the delivery amount, the delivery interval, and the specific drivers assigned to a customer.

The routing problem in Erera et al. [57] is motivated by a beer, wine, and spirits distributor. The company wants to increase service quality by visiting regular customers by not more than two different drivers per weekday over a long horizon.

An aircraft fleet routing and scheduling problem, faced by many airlines, is solved by Ioachim et al. [91]. For marketing purposes, regular flights have to be scheduled at the same time every day.

2.2.4 Consistency apart from customer satisfaction

Consistency in routing plans is also investigated for reasons other than customer and employee satisfaction. Francis et al. [66] examine the problem of transporting loan items (e.g., books) between spatially distributed libraries. The authors propose a periodic vehicle routing formulation in which each library is visited by the same driver in the planning period. The strict consistency requirement reduces the number of integer decision variables and makes the problem tractable with the devised mathematical programming approach. The same observation is made in Chapter 4 (Kovacs et al. [103]). Dayarian et al. [45] and Dayarian et al. [46] investigate a multi-period routing problem motivated by the dairy industry. In this problem, milk is transported from the producers to a dairy. An association of milk producers is responsible for negotiating long-term contracts with carriers. The negotiations

and payments are based on a single routing plan that will be executed on each day for the next, say, six months. The challenge in generating the basic routing plan is the daily and seasonally varying milk supply.

In this example, consistency is a matter of cost assessment rather than a matter of customer satisfaction. Similarly, in a priori routing, routes are planned before full demand information becomes available (Bertsimas [12], Bertsimas et al. [13], Golden and Stewart [74], Jaillet [92]). These routes define the visit sequence of potential customers for a long period of time.

2.3 Problem description

The problem description is similar in all vehicle routing problems that consider consistency: A set of drivers are visiting a set of spatially distributed customers over multiple periods, e.g., several days. The planning period can be bounded, such as planning week after week, or unbounded, e.g., planning on a rolling horizon basis. The operations are constrained by limited workforce (or fleet), limited working time (or travel time), and limited vehicle capacity. The daily input parameters are fluctuating, e.g., customers require service only on some days, demand is varying from day to day, travel times depend on the traffic, and service times depend on the delivery quantity. In many cases, the service times and the travel times also depend on the drivers' familiarity with the assigned region and customers (e.g., Kunkel and Schwind [110], Smilowitz et al. [158], Zhong et al. [190]). Input parameters are usually deterministic in bounded planning period problems and either stochastic or unknown (i.e., information is revealed day by day) in unbounded planning horizon problems. The objective always involves minimizing travel cost.

An obvious approach to deal with varying input parameters is to optimize routes day after day. This approach generates minimal cost routing plans by taking into account the actual input parameters; however, it has significant drawbacks since customer data has to be acquired, new routes have to be computed in a short amount of time, and the routes have to be forwarded to the drivers on a daily basis. Additionally, the daily routes are different and impose a high learning burden on the drivers and reduce the service quality for the customers. Efficient solution approaches for the PVRP and the IRP incorporate several periods simultaneously. Nevertheless, holistic approaches also produce inconsistent routing and replenishment plans when the focus is on cost (Coelho et al. [32], Coelho et al. [33]).

A sufficient condition for inconsistent routing plans is fluctuating input parameters together with either cost minimization or tight resource constraints. Let C, F, R, and I be

Boolean variables that are true if cost is minimized (C), demand is significantly fluctuating (F), resource constraints are tight (R), and the routing plans are inconsistent (I). Then, expressed in Boolean algebra, the condition $F \wedge (C \vee R)$ is sufficient for I, i.e., $F \wedge (C \vee R) \Rightarrow I$. Each customer can be served on an individual route if cost is not a concern and resources are unlimited. Without variations in the input parameters, there is at least one minimum cost solution that provides perfect consistency, e.g., when the same set of optimal routes is executed each day. A real-world example of periodic routing with constant demand is school bus routing; students are picked up in the same sequence every morning. Similarly, the same routing plan can be repeated periodically if customer demand follows a regular pattern, e.g., when one vehicle serves two customers *a* and *b* with *a* requiring service every other day and *b* requiring service every third day, the routing plan can be repeated in a six day loop (six is the least common multiple of two and three).

Condition $F \wedge (C \vee R)$ is also necessary for inconsistency, i.e., $F \wedge (C \vee R) \Leftrightarrow I$, if there is only a single optimal solution to the problem. For example, consider two customers, *a* and *b*, with a demand of one unit each on two consecutive days. The vehicle capacity is two units. The solution with route 0-*a*-*b*-0 on day 1 and route 0-*b*-*a*-0 on day 2 is feasible and has minimal cost (we assume a symmetric distance matrix). Yet, the routing plan is inconsistent despite constant input parameters.

Figure 2.2 illustrates a routing example with a planning period of three days. The square denotes the depot and the circles denote customers. The customer demand, given next to each circle, is variable. The vehicle has a capacity of five units and the objective is to minimize cost. The daily routing plans are different because some vehicles are fully utilized and the focus is on cost. Visiting customers 4, 5, and 6 on the same route on day one and day two would exceed the vehicle capacity on the second day. Executing the day one routing plan on day three would be feasible and it would result in higher consistency. However, serving customer 3 on the same route with customer 1 and 2 is cheaper.

In Figure 2.3, the customers requesting a service are changing from one day to the next. Generally, varying customers generate the largest differences in the daily routes.

Well-known problems that conform to the given problem definition and are, therefore, prone to inconsistency are the PVRP and the IRP. The PVRP was introduced in Beltrami and Bodin [6] and further examined, e.g., in Christofides and Beasley [30] and Russell and Igo [147]. Customers require a given number of visits (one or several) within a bounded time interval. The delivery days, however, are flexible, i.e., the supplier can select the days on which he wants to visit a customer from a set of feasible day combinations. Visit combinations can be fixed, e.g., days 1, 3, and 5, or days 2, 4, and 6 if a customer requires three

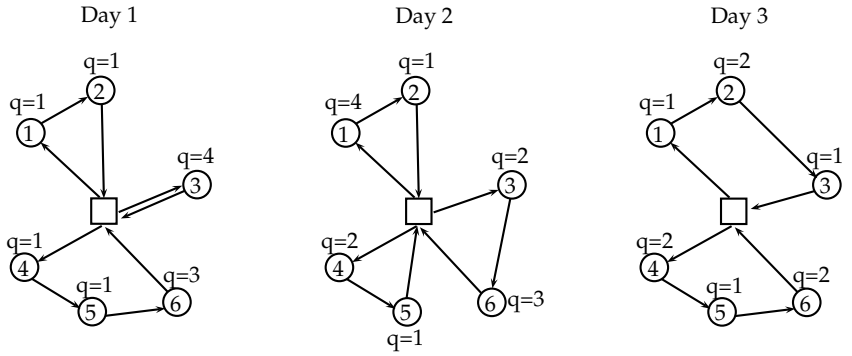


Figure 2.2 Inconsistency because of varying demand. Vehicle capacity is five. Demand (q) is given for each customer.

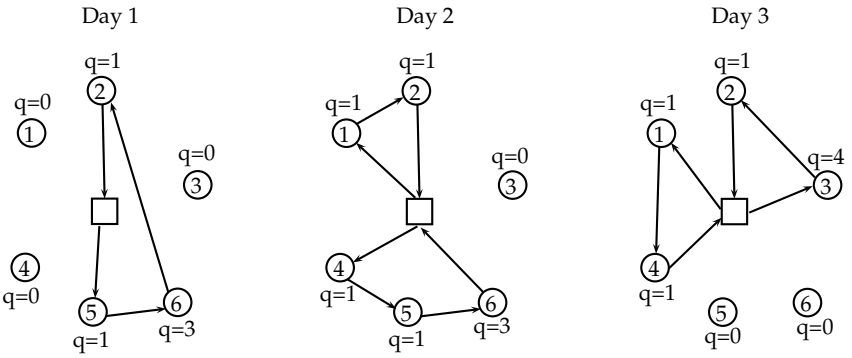


Figure 2.3 Inconsistency because of changing customer requests. Vehicle capacity is five. Demand (q) is given for each customer.

visits (Beltrami and Bodin [6], Christofides and Beasley [30]), they can be defined as fixed visit intervals, e.g., visit a customer every other day (Chao et al. [28]), or be specified by a minimum and maximum spacing between two visits (Gaudioso and Paletta [70]). In Francis et al. [66, 67], customers may be visited any number of times provided that the minimum number of visits is met; this problem is referred to as the PVRP with service choice (PVRP-SC). Survey papers on the PVRP are presented by Campbell and Wilson [24] and Francis et al. [68].

In the IRP, the supplier decides about the timing and the quantity of a delivery in addition to the routing plan (Bell et al. [5]). The basic IRP is deterministic and the planning horizon is bounded (exceptions to both limitations have been presented in the literature). The optimal replenishment plan avoids stockouts at customer locations at minimal routing and holding costs. Common replenishment policies are the maximum level policy (i.e., delivery amount is flexible, but the storage capacity at the customer has to be respected) and

Time Period	Acknowledged Side Benefit	Solution Approach
1970's	**Christofides [29]:** improved driver familiarity with customers.	A priori routing and districting
	A working paper by Stewart cited in Golden and Stewart [74]: improved driver familiarity with routes and customers.	A priori routing
1980's	**Beasley [4]:** improved driver familiarity with routes and customers.	A priori routing
	Waters [182]: improved driver familiarity with routes and customers and regular arrival times at customers.	A priori routing
	Wong and Beasley [184]: improved driver familiarity with customers.	Districting
1990's	**Bertsimas [12]:** personalization of service.	A priori routing
	Savelsbergh and Goetschalckx [149]: personalization and predictability of service and improved driver efficiency.	A priori routing

Table 2.1 Early papers that acknowledge the benefit of the applied solution approach on service consistency.

the order-up-to-level policy (i.e., customer inventory is filled up at each visit). For further information on the IRP and its variants, we refer to Bertazzi et al. [10], Campbell et al. [19], and Coelho et al. [33].

2.4 Early approaches to cope with fluctuating demand

Planning new routes each morning to cope with fluctuating demand results in cheap routing plans, but it entails many operational difficulties (e.g., data has to be acquired regularly, re-optimization is time consuming, communicating new routes to drivers every day is costly). Therefore, alternative approaches have been proposed in the literature. These approaches can be divided into three groups: approaches that generate a priori routes and perform daily adaptations if necessary, approaches that assign each driver to a fixed service region, and approaches that simplify the problem by smoothing fluctuations. All strategies promote service consistency but consistency was rarely addressed explicitly in early publications. In Table 2.1, we list early papers in chronological order that acknowledge the side benefit of the applied solution approach on service consistency.

2.4.1 A priori routing

Many papers have been devoted to the problem of serving customers with stochastic demand over a long period (e.g., several weeks). A common strategy to overcome the difficulties of daily re-optimization is to design a set of routes in advance, i.e., based on stochastic information, and use the pre-planned routes as a basis for the daily routes. We refer to these pre-planned routes as a priori routes. Depending on the context, the a priori routes can be updated in different ways to better match the actual demand realization. For example, customers without demand are skipped if the demand becomes known before the vehicles start their tours. This updating strategy is investigated, e.g., in Bertsimas [12], Bertsimas et al. [13], Gendreau et al. [73], and Jaillet [92]. This strategy is inapplicable if a driver learns the actual demand only when he arrives for delivery. This situation is investigated, e.g., in Bertsimas [12], Golden and Stewart [74], Savelsbergh and Goetschalckx [149], and Tillman [172]. In some cases, the actual demand on an a priori route is larger than the vehicle capacity; then, the updating mechanism inserts a detour to the depot for reloading. In Campbell and Thomas [23], customers requiring a service are random but each order has to be delivered before a specific deadline (e.g., FedEx's next-business-day delivery service with guaranteed delivery by 10 a.m.). Two updating strategies are examined: First, customers are visited even if deadlines are violated; second, customers are skipped if the deadline cannot be met (skipped customers are then visited by a separate vehicle). In the vehicle rescheduling problem presented by Spliet et al. [162], routes are updated by minimizing travel cost and a penalty cost that is incurred for deviating from the a priori routes.

Figure 2.4 shows the difference between the a priori strategy with different updating mechanisms and the re-optimization strategy. The first figure illustrates the a priori routes for visiting six potential customers. The second figure shows the route that is executed when demand information becomes available only upon arrival; all customers are visited in the same order as in the a priori routes. In the third figure, customers without demand are skipped. The last figure illustrates the result of re-optimization by taking into account the actual demand realization.

Demand fluctuations can be ignored if vehicle capacity is unlimited. Challenges are then posed by stochastic customer locations. This problem is called the Noisy Traveling Salesman Problem (NTSP) and it is examined by Braun and Buhmann [17] and Li et al. [115]. Customer locations are unknown but there are areas with higher probability of customers placing an order, e.g., business districts in small package shipping. Delivery is performed by a single vehicle; the capacity is unbounded. The specific visit locations are changing from day to day but the shape of the optimal traveling salesman routes will be similar. The

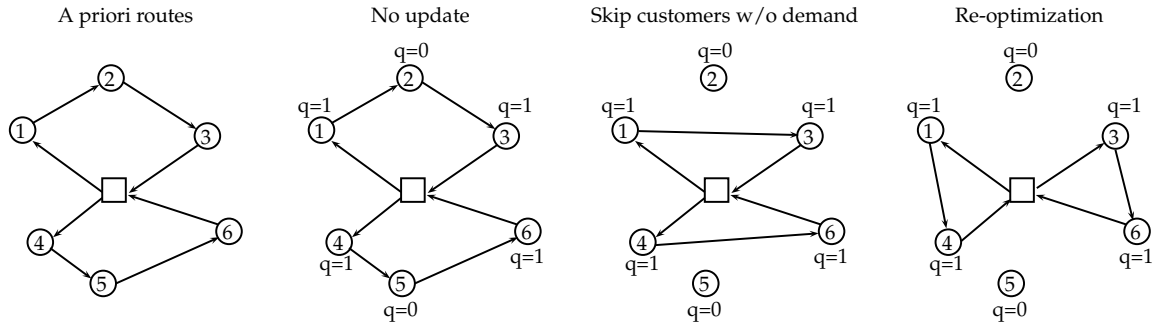


Figure 2.4 A priori routing with two updating mechanisms compared to re-optimization strategy. Actual demand (q) is given for each customer. Vehicle capacity is three.

goal in the NTSP is to sketch a route in advance that can be easily adapted to the daily requirements. Figure 2.5 illustrates the quad tree approach of Li et al. [115] to generate the a priori route. Based on a random sample, the algorithm starts by subdividing the visit area into smaller regions. The maximum number of customers per sub-region, called a quadrant, is given. Two customers per quadrant are allowed in our example. If a quadrant contains more than two customers, it is subdivided into four smaller quadrants. The next step is to compute the centroid of each quadrant (this is only necessary for quadrants with more than one customer). Finally, the centroids are linked to form the a priori route. On each day, the a priori route is updated to match the daily requirements.

Given an updating mechanism, the main goal in a priori optimization is robustness with regard to cost, i.e., finding a single set of routes that minimizes the expected travel cost (including the expected extra cost of reloading at the depot if vehicle capacity is reached) for a large number of demand and location realizations. Empirical results indicate that the routing cost of a priori optimization is very close to the cost of re-optimization: Christofides [29] and Golden and Stewart [74] report a difference of 2.7% and 8%, respectively; Savelsbergh and Goetschalckx [149] give empirical evidence that in some cases a priori solutions even outperform daily routes generated by heuristics; Hemmelmayr et al. [86] compare the a priori strategy to a more flexible routing strategy for an inventory routing problem – the differences between the two approaches are small; Bertsimas [12] and Bertsimas et al. [13] show that the cost of a priori optimization is asymptotically very close to the cost of re-optimization for both updating strategies discussed above (following the a priori route strictly and skipping customers without demand). A side benefit of a priori optimization is route consistency; each daily routing plan is derived from the same set of a priori routes. The actual consistency, however, depends on the chosen updating strategy.

Many recent publications that address service consistency explicitly use the concept

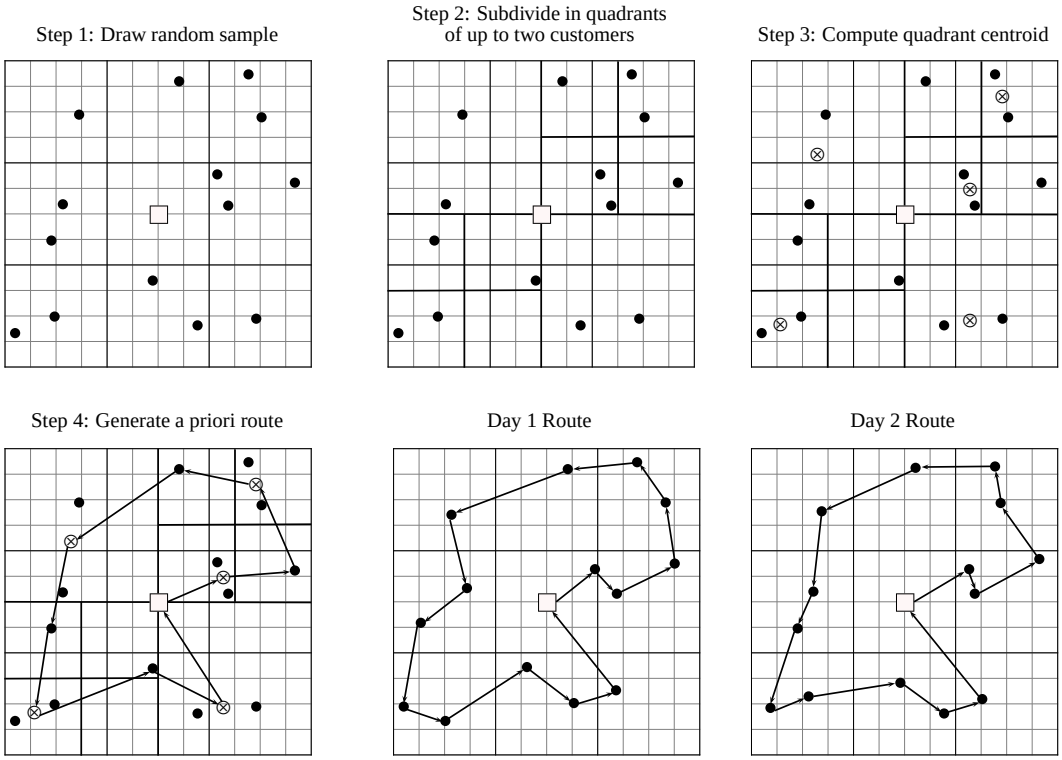


Figure 2.5 Quad tree appraoch of Li et al. [115] for the noisy traveling salesman problem.

of deriving daily routes from routes that have been fixed in advance (e.g., Campbell and Thomas [23], Day et al. [44], Groër et al. [75], Sungur et al. [167]). Accordingly, many terms have been used to refer to the same concept. Table 2.2 lists synonyms of a priori routing, the context in which the term is applied, and some papers using that term.

A survey on routing problems with stochastic input parameters is presented by Campbell and Thomas [20] and Gendreau et al. [72].

2.4.2 Districting

The complexity of daily re-optimization is often reduced by subdividing a service region into smaller districts. All customer requests within a district are served by one driver. Repetitive deliveries within the same region improve the drivers' familiarity with their service territory (Wong [185]). Customer service is also improved since regular customers are visited by the same driver each time they require a service. Additionally, districts are planned to balance the workload among drivers.

Districting and a priori routing are similar concepts but there is one fundamental difference: A priori routing specifies the customers belonging to a route and also the sequence of

Term	Context	Used in (e.g.)
A priori routes	Random demand or customers	Bertsimas [12], Jaillet [92]
Average trajectory routes	Random locations	Braun and Buhmann [17], Li et al. [115]
Backbone routes	Deterministic but dynamic demand	Day et al. [44]
Base plan routes	Random customers	Eveborn et al. [58]
Fixed routes	Deterministic but dynamic or random demand	Beasley [4], Christofides [29] Savelsbergh and Goetschalckx [149]
Master plan routes	Random service time and customers	Sungur et al. [167]
Master schedule	Random demand	Spliet et al. [162]
Modified-fixed routes	Random customers (skipping is allowed)	Benton and Rossetti [8]
Permanent routes	Random customers	Benton and Rossetti [8]
Semi-fixed routes	Random customers (skipping is allowed)	Haughton [80], Waters [182]
Skeleton routes	Random locations	Li et al. [115]
Standard routes	Random customers	Bianchi et al. [15], Haughton [82]
Template routes	Deterministic but dynamic demand	Groër et al. [75], Kovacs et al. [107]

Table 2.2 Synonyms for the concept of deriving daily routes from routes that have been fixed in advance.

the stops. Accordingly, a priori routes become ineffective if there is a substantial change in the regular customers or in the demand pattern (Benton and Rossetti [8], Erera et al. [57]). Districting decisions are made on a strategic or tactical level, i.e., drivers remain with their district for several years. Potential customers are grouped into districts and, therefore, assigned to routes well before they place an order; routing decisions, however, are made on a daily basis when actual demand is known. As a consequence, districting is a generalization of the a priori routing approach.

Solving the problem in two phases, districting first - routing second, reduces the flexibility in generating optimal routes when demand is fluctuating. For example, a large demand in a district might force the assigned driver to perform a second trip while the driver in another district is underutilized because of low demand. However, districting significantly reduces operational difficulties. Figure 2.6, illustrates the difference between daily re-optimization and districting. Four drivers are available with a capacity of four customers a day. Customers placing an order are random. The three figures at the top show the solution for three days with daily re-optimization. There is no incentive to dispatch all drivers because three drivers can cover the demand at lower cost. So, the workload is divided among three drivers while one driver is idle each day. The three figures at the bottom show the solution with districting. The region is subdivided into four service territories by taking into account the expected workload. Each driver operates only within his service territory and visits three customers each day.

Earlier articles on districting (in the context of vehicle routing) assume deterministic in-

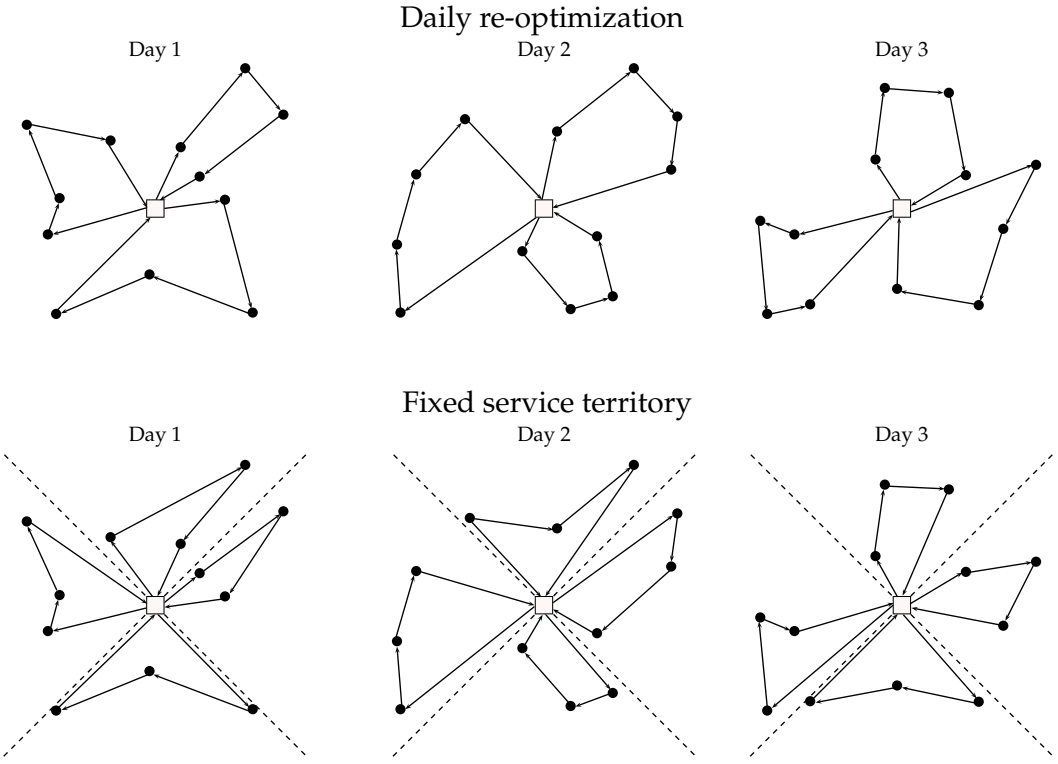


Figure 2.6 Solution with daily re-optimization (top) and fixed service territory (bottom). Each customer has a demand of one unit; vehicle capacity is four.

put parameters (e.g., demand) that might be sampled from historical data (e.g., Christofides [29], Hardy [79], Wong and Beasley [184]). Recent papers also account for the stochastic nature of the problem (e.g., Carlsson [26], Carlsson and Delage [27], Haughton [83], Haughton [84], Haugland et al. [85], Lei et al. [113], Zhong et al. [190]). Daganzo [42] and Daganzo [43] present simple strategies for estimating the transportation cost when districts have irregular shapes. The increase in travel distance if customers are pre-assigned to districts compared to daily re-optimization is 3.8% in Christofides [29] and 7.1% in Hardy [79].

Many papers on districting (e.g., in sales and salt spreading) are devoted to the problem of partitioning a region into small areas with their own autonomous depot. Each depot is responsible for the operations performed within its district, i.e., drivers are never assigned to external districts (Van Oudheusden et al. [175]). Work in this field is presented, e.g., in Fleischmann and Paraschis [61], Marlin [117], Muyldermans et al. [121], and Muyldermans et al. [122].

Haughton [83] and Haughton [84] examine the concept of territory sharing, i.e., merg-

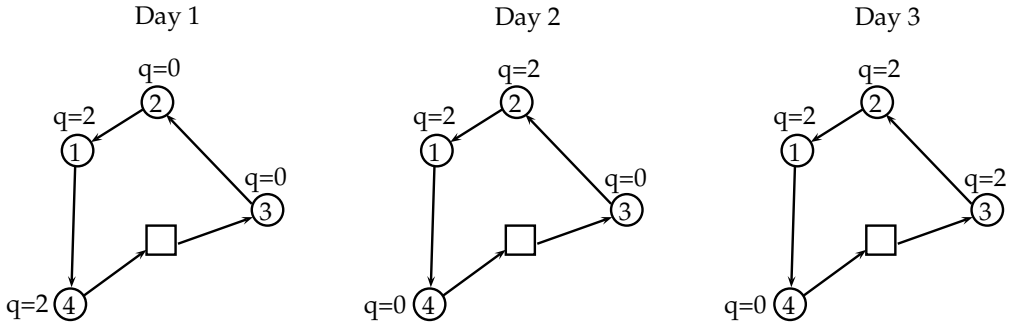


Figure 2.7 Smoothing demand: Demand follows a Bernoulli process; the actual demand (q) is given next to the customers. Each customer is delivered every day with its expected demand (one unit). Vehicle capacity is eight units.

ing districts and serving them by a group of drivers. A group size of one corresponds to the basic districting problem, the districts become larger with increasing group size, and the problem corresponds to daily re-optimization if all drivers are assigned to one group.

2.4.3 Demand stabilization

One approach to maintain route consistency is to eliminate demand fluctuations (e.g., Haughton [80], Haughton [81]). For example, if demand follows a Bernoulli process, i.e., a customer i demands either q_i units a day with probability p_i, or zero with probability $(1-p_i)$, a company might choose to deliver each customer with its expected demand ($p_i q_i$) every day instead of responding to actual demand realizations. For example, if a customer has a daily demand of either two units or zero with a probability of 0.5, then this customer is visited each day and its expected demand (one unit) is delivered. Figure 2.7 illustrates this scenario with four customers. The daily demand of each customer is either two units or zero with a probability of 0.5 each. The actual demand (q) on each day is given next to the customers. Customers are visited each day and their expected demand (one unit) is delivered. The vehicle capacity is eight units. The same set of vehicle routes can be executed day by day; drivers have no learning requirement and customers are visited on the same route at the same time with the same quantity delivered each day. However, in many cases, the delivered quantity deviates from the demand incurring inventory and shortage costs at the customers (e.g., on day one, customer 1 and 4 have a shortage of one unit while customers 2 and 3 have an inventory of one unit, respectively). Without inventory and shortage costs, this delivery strategy is superior to the a priori strategy (with skipping customers with zero demand) only when the visit probabilities (p_i) are large and when the number of routes is

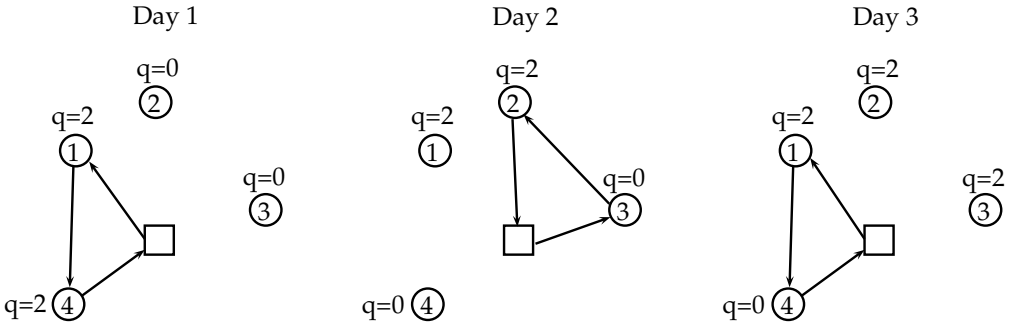


Figure 2.8 Smoothing demand: Demand (q) follows a Bernoulli process. Each customer is visited every second day and four units are delivered each time. Vehicle capacity is eight units.

neither very small nor very large (Haughton [80]). If the number of routes is very large, i.e., the number of customers per route is small, many routes can be eliminated with the a priori strategy; the lower the p_i's, the higher the savings. When the number of vehicles is very small, i.e., the number of customers per route is large, it is more difficult to reduce the number of routes through reduced delivery quantities ($p_i q_i$). With inventory and shortage costs, the daily delivery strategy is unlikely to be an appropriate alternative to daily re-optimization (Haughton [81]).

A more viable alternative is to deliver q_i/p_i units at fixed intervals of $1/p_i$ days to each customer. Using the example above, each customer receives four units every second day. This approach is illustrated in Figure 2.8. Route consistency is lower than before, but the same route is repeated periodically. Additionally, this strategy is never inferior to a priori routing with regard to routing cost and the risk of shortages quickly diminishes with time (Haughton [80]). However, this strategy tends to build large inventories. Haughton [81] suggests well-timed and controlled changes in the delivery routes in order to increase demand responsiveness, i.e., monitoring the inventory status of each customer and adapting the routes accordingly.

2.5 Consistency

Consistency in multi-period routing problems can be measured in three dimensions: arrival time consistency, person-oriented consistency, and delivery consistency.

Regular customers appreciate predictable service times. Arrival time consistency is achieved by visiting customers at similar times of the day over the long run.

Person-oriented consistency can be viewed from two perspectives: customer perspective and driver perspective. Visiting a customer with the same driver repeatedly improves service quality because of the improving customer-driver relationship and the customized service. Person-oriented consistency from the customer's point of view is referred to as driver consistency; it is achieved by reducing the number of different drivers that serve a customer. Consistency is also important for the driver's satisfaction and productivity. Irregular routing plans and unfamiliar tasks upon visiting new customers pose large learning requirements on drivers and make the execution of routes costly. Person-oriented consistency from the driver's point of view is achieved by assigning each driver to the same service region repeatedly; this notion is also called region consistency. If a service is very complex (e.g., as it is often the case in home health care), a driver should be assigned to the same customer frequently in order to make him more familiar with the assigned task; this is referred to as customer consistency.

Inventory routing problems require decisions about the timing and the size of an order. Service quality is increased by replenishing each customer's stock at regular intervals, with similar delivery quantities, or by keeping inventory levels stable. The attainable level of delivery consistency depends on the variability of demand (e.g., it is infeasible to replenish inventories at fixed intervals with fixed quantities if demand is highly fluctuating).

In this section, we survey papers on vehicle routing that explicitly address at least one dimension of consistency. In Section 2.3, we gave a general problem definition; here, we classify the different approaches according to the input parameters that are varying from day to day and the information that is available at the time of optimization. Table 2.3 gives an overview of recent articles. Models that work with full information typically have a bounded planning horizon (e.g., one week); models that use stochastic or historic information either generate routes day by day by considering historic routing plans, or they generate a priori routes that can be used for an unlimited amount of time. We examine how consistency considerations are modeled and discuss the devised solution approaches. Additionally, we summarize the findings of how the emphasis on consistency affects routing cost.

2.5.1 Arrival time consistency

In this section, we give an overview of concepts for modeling arrival time consistency. We describe efficient solution approaches and summarize results that analyze the impact of reducing variations in arrival times on transportation cost.

	Consistency			Varying Parameters				Available Information	
Paper	Delivery	Arrival Time	Person-oriented	Demand	Service Time	Travel Time	Customers	Full Information	Stoch./Hist. Information
Coelho et al. [32]	✓		✓	✓			✓	✓	
Coelho and Laporte [34]	✓		✓	✓			✓	✓	
Coelho and Laporte [35]	✓		✓	✓			✓	✓	
Day et al. [44]	✓	✓	✓	✓			✓	✓	
Erera et al. [57]			✓	✓			✓		✓
Eveborn et al. [58]			✓				✓		✓
Feillet et al. [59]		✓	✓				✓	✓	
Francis et al. [66]	✓		✓	✓			✓	✓	
Francis et al. [67]	✓	✓	✓				✓	✓	
Groër et al. [75]		✓	✓	✓	✓		✓	✓	✓
Haughton [82]		✓	✓				✓		✓
Haughton [83]			✓	✓					✓
Kovacs et al. [103]		✓	✓	✓	✓		✓	✓	
Kovacs et al. [106]		✓	✓	✓	✓		✓	✓	
Kovacs et al. [107]		✓	✓	✓	✓		✓	✓	✓
Kunkel and Schwind [110]			✓	✓	✓		✓		✓
Nowak et al. [129]		✓	✓				✓	✓	✓
Schneider et al. [151]			✓	✓	✓	✓	✓		✓
Schneider et al. [152]			✓				✓		✓
Smilowitz et al. [158]			✓	✓			✓	✓	
Spliet [159]		✓	✓	✓					✓
Spliet and Desaulniers [160]		✓		✓					✓
Spliet and Gabor [161]		✓		✓					✓
Sungur et al. [167]			✓		✓		✓		✓
Tarantilis et al. [170]		✓	✓	✓	✓		✓	✓	
Zhong et al. [190]			✓		✓	✓	✓		✓

Table 2.3 Recent papers that consider routing plan consistency and their modeling details.

Modeling arrival time consistency

Arrival time consistency is either modeled by hard constraints, i.e., variations are bounded by a given value, or soft constraints, i.e., variations are penalized in the objective function.

Groër et al. [75] introduce the Consistent Vehicle Routing Problem (ConVRP). In the ConVRP, the maximum arrival time variation is bounded for each customer to a given value L. The maximum arrival time difference l_{max} is the largest difference between the latest and the earliest arrival time per customer among all customers. Figure 2.9 illustrates l_{max} for a routing plan with a planning period of five days. The black squares denote the arrival times at the customer with the largest variation; the white squares denote the arrival times at the other customers. Arrival time consistency is achieved by enforcing $l_{max} \leq L$. The same modeling approach is applied in Chapter 3 (Kovacs et al. [107]) and Tarantilis et al. [170]. An arrival time difference of $l_{max} = 0$ is considered in Ioachim et al. [91]. The problem of visiting a customer at exactly the same time on several days is similar to routing problems with temporal vehicle synchronization: In a multi-period setting in which consistency is important, customers are visited at the same time each time they are visited. In contrast, in a single-period routing problem with synchronization constraints, several drivers have

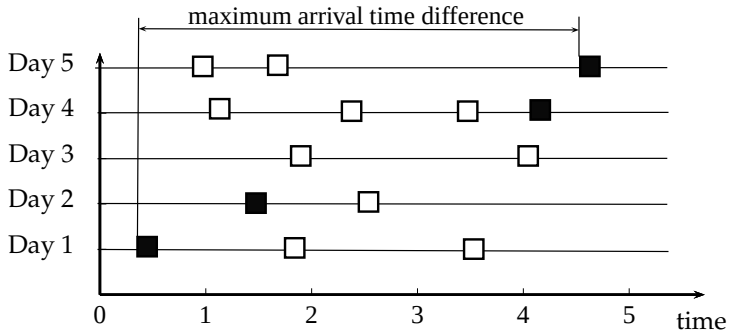


Figure 2.9 Maximum arrival time difference. Black squares denote the arrival times of a specific customer; white squares are arrival times of arbitrary customers.

to meet simultaneously at a customer. For example, synchronization is necessary in home health care if two workers are needed for bathing a heavy patient (Bredström and Rönnqvist [18]). In ground handling operations at airports (e.g., baggage handling and cleaning) teams have to work simultaneously on large airplanes in order to finish the task within a short time window (Dohn et al. [50]). Figure 2.10 illustrates the similarities between arrival time consistency and synchronization. The three figures on the top show a three-day routing problem; customers 1 and 3 need to be visited at the same time every day. The figure at the bottom shows a single period routing problem; the tasks at customers 1 and 3 require three drivers simultaneously. The solution of the synchronization problem is equivalent to the union of all routes in the consistency problem. A survey on synchronization in vehicle routing is given in Drexl [51].

Soft constraints for arrival time consistency are examined in Chapter 4 (Kovacs et al. [103]). The maximum arrival time difference is penalized in the objective function because it is difficult to predict reasonable bounds if demand is fluctuating: loose arrival time constraints decrease service quality while excessively tight constraints lead to a disproportional increase in travel cost. In Chapter 5, we present a multi-objective approach where routing cost and the maximum arrival time difference are independent objectives of the problem (Kovacs et al. [106]). Feillet et al. [59] propose an alternative to the maximum arrival time difference constraint. For example, visiting a customer on five days at (8:00, 8:05, 8:02, 8:58, 9:00) or at (8:00, 9:00, 8:20, 8:40, 8:30) yields the same maximum arrival time difference of one hour. Yet, the first schedule shows more regularity. Rather than bounding the maximum arrival time difference, Feillet et al. [59] minimize the maximum number of different time classes each customer is visited in. Two visits at customer i on day α and β are assigned to the same time class if the arrival times $a_{i\alpha}$

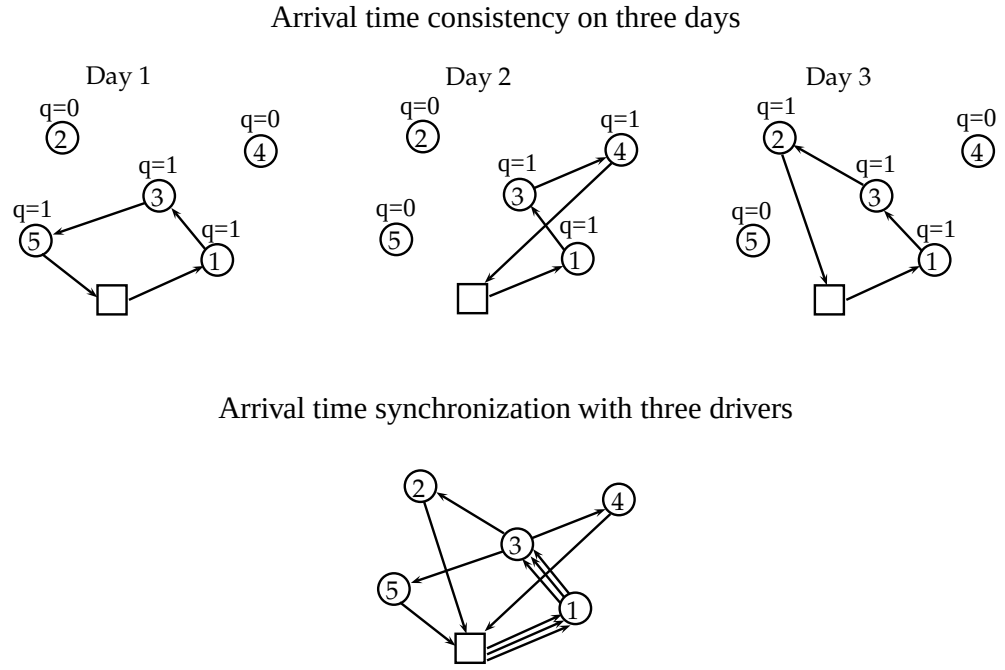


Figure 2.10 Arrival time consistency on three days versus synchronization of three drivers. Arrival time consistency and synchronization is required at customers 1 and 3, respectively. Vehicle capacity is three.

and $a_{i\beta}$ fulfill $|a_{i\alpha} - a_{i\beta}| \leq P$. So, for $P = 5$ the first visiting schedule is partitioned into two time classes {{8:00, 8:02, 8:05},{9:58, 9:00}} and the second schedule into five time classes {{8:00},{8:20},{8:30},{8:40},{9:00}}. Therefore, the first schedule has higher consistency.

Spliet [159], Spliet and Desaulniers [160], and Spliet and Gabor [161] investigate the problem of assigning a single time window to each customer before the actual demand is known; once fixed, customers have to be visited within the respective time window each day. The tighter the time window, the better the arrival time consistency. The width of the time windows serves the same purpose as the bound on the maximum arrival time consistency, L, in the ConVRP.

The decision to model arrival time consistency either with hard or with soft constraints depends on the availability of information. Using soft constraints is feasible in both deterministic and stochastic environments. Using hard constraints, however, may lead to infeasible or unreasonable solutions when demand fluctuations cannot be anticipated. Figure 2.11 gives an example when customer occurrence is random and the maximum arrival time difference is bounded by zero. There is one vehicle to visit customers. The figure at the top

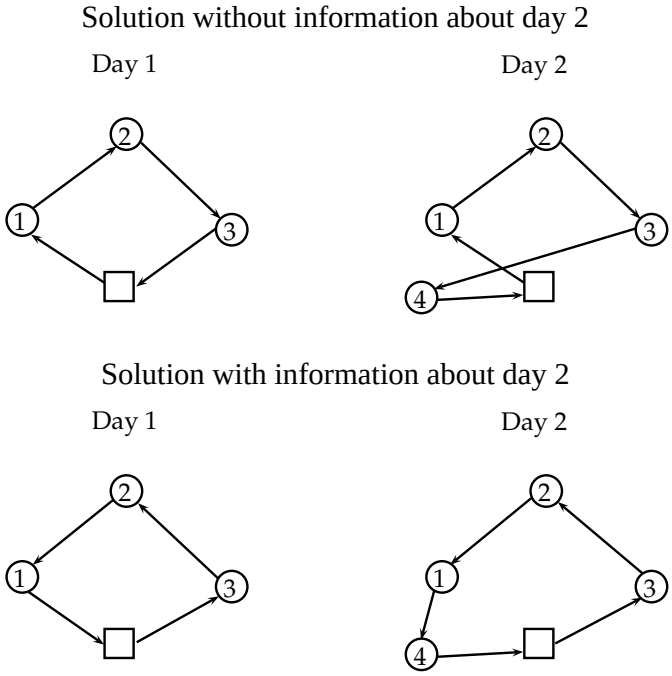


Figure 2.11 Effect of bounding the variation in the arrival times when future information is unavailable.

shows a solution when the route for day 1 is planned without information about day 2. The only feasible insertion position for customer 4 on day 2 is the last visit before the vehicle returns to the depot. Full information is available in the solution at the bottom. Customer 4 is again inserted into the last position; the routes, however, are more efficient.

Arrival time consistency is not modeled explicitly in Day et al. [44], Haughton [82], and Nowak et al. [129]. However, the applied a priori solution approaches promote stable arrival times.

Solving problems with arrival time consistency considerations

Many heuristics that consider arrival time consistency are based on the idea of a priori routing (e.g., Groër et al. [75], Kovacs et al. [107], Repoussis et al. [142], Tarantilis et al. [170]). An essential feature of the a priori approach is that it fixes the sequence in which customers are visited: if customer *a* is visited before customer *b* on the same a priori route, then customer *a* is visited before customer *b* on all days on which they both require service (Groër et al. [75]). Visiting customers in the same sequence every day promotes arrival time consistency. In the ConVRP, Groër et al. [75] consider a bounded planning horizon and deterministic demand. The authors include all customers into the a priori routes (referred

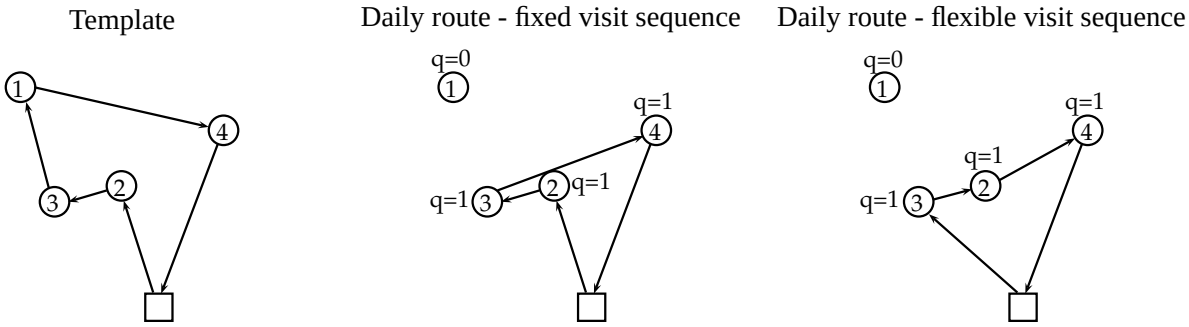


Figure 2.12 Two updating strategies: preserving visit sequence versus changing visit sequence.

to as template routes by the authors) that are concerned with consistency, i.e., all customers that require at least two visits during the planning period. On each day, customers without demand are skipped and customers that require only one visit are inserted at their cheapest position. In Chapter 3, we extend this updating strategy by allowing deviations from the visit sequence given by the template (Kovacs et al. [107]). Figure 2.12 illustrates the intuition behind this strategy. The first figure shows the template with four customers; the second figure shows the solution derived from the template when customer 1 is skipped; the third figure presents a more flexible updating strategy that allows a change in the visit sequence. Repoussis et al. [142] present a hybrid solution strategy: Template routes are generated by tabu search. Based on the resulting template, the customer-to-driver assignment is fixed for a part of the customer set and the partially fixed ConVRP is solved by branch-and-cut.

In Chapter 4, we propose an alternative to the a priori solution approach (Kovacs et al. [103]). In each iteration, a routing plan is generated for each day in the planning horizon regardless of the resulting arrival time consistency. Then, the maximum arrival time difference is reduced by repeatedly reversing parts of selected routes. The reversal corresponds to a 2-opt move (Lin [116]) on the route that contains the customer with the largest maximum arrival time difference. This solution approach performs well on ConVRP instances; the stronger the fluctuations in the customer demand, the larger the advantage of the flexible approach compared to the a priori approach.

In Feillet et al. [59], Spliet [159], Spliet and Desaulniers [160], and Spliet and Gabor [161], arrival time consistency is achieved by assigning customers to time windows of given width. This idea is illustrated in Figure 2.13. Customers 1, 2, and 3 require several visits during the planning period; the variation in their arrival times is bounded by time windows. The blank squares denote arbitrary customers that require only one visit. With this approach,

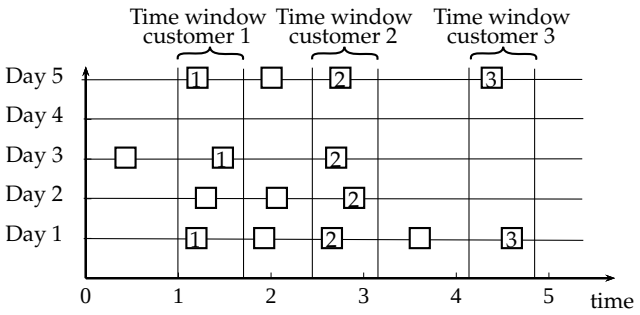


Figure 2.13 Achieving arrival time consistency by assigning time windows.

the multi-period routing problem decomposes into several single-period problems. In Spliet and Desaulniers [160] and Spliet and Gabor [161], each customer is assigned to a time window using a column generation algorithm. Feillet et al. [59] use an iterative solution approach. In each iteration, each customer is associated with one or more time windows. The resulting daily problem is an extension of the VRPTW referred to as Vehilce Routing Problem with multiple Time Windows (VRPmTW). Arrival time consistency is improved by gradually reducing the number of time windows in which a customer is visited.

Evaluating the cost of arrival time consistency

Groër et al. [75] examine the cost of consistency in the ConVRP. The authors report an average increase in travel cost between 6.6% and 15% (the cost of person-oriented consistency also contributes to this result). According to Feillet et al. [59], arrival time consistency increases travel cost between 1% and 5.9%. In both papers, the authors require the drivers to start their tour at time zero. Additionally, waiting is prohibited. The effect on consistency when departure times from the depot are fixed is illustrated in Figure 2.14. The planning horizon is three days. Customer 1 requires service on day 2 and day 3; customer 2 requires service on day 1 and day 3. The travel time between any two nodes is one time unit; service times can be neglected. The left figure shows the solution when departure times are flexible; both customers are visited at exactly the same time on both days they require service. The driver has to start his route at time zero in the right figure. Customer 2 is visited at time one on day 1 and at time 2 on day 3, resulting in an arrival time difference of one time unit. Figure 2.15 illustrates the effect of executing routes without idle time. All travel times are again one time unit and service times are neglected. One driver is visiting customers 1 and 3 on day 1 and customers 1, 2, and 3 on day 2. In the left figure, the driver waits one time unit before starting customer 3's service in order to achieve perfect arrival time consistency.

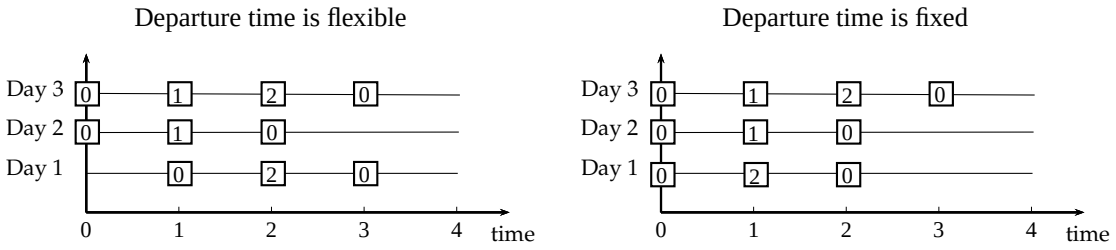


Figure 2.14 Effect of fixed departure times on arrival time consistency (adapted from Kovacs et al. [107]).

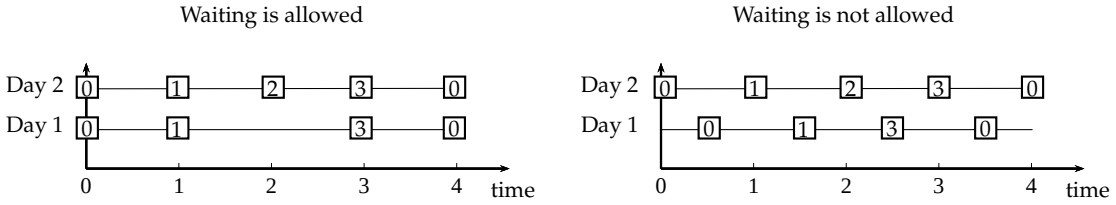


Figure 2.15 Effect of prohibiting waiting along the route.

The right figure shows a solution that has a minimal maximum arrival time difference when waiting is prohibited but later departure from the depot is allowed; by delaying the departure time on day 1 customers 1 and 3 are visited with an arrival time difference of 0.5 time units each. The maximum arrival time difference increases to one time unit at customer 3 if departure time is fixed at zero.

In Chapter 3, we demonstrate that fixing departure times and prohibiting waiting can cause the travel cost to soar if arrival time constraints are too tight (Kovacs et al. [107]). E.g., in the ConVRP, a 60% tighter constraint on the maximum arrival time difference increases travel cost by up to 186.16%. The reason for this result is the large number of routes needed to cope with the strict arrival time difference constraints. Allowing flexible departure times reduces the impact of tight arrival time consistency constraints on cost significantly, even without idle times along the route. Therefore, arrival time consistency can be improved without increasing working time (Kovacs et al. [103, 106]).

Francis et al. [67] examine the effect of operational flexibility on arrival time consistency in the PVRP-SC. Arrival time consistency deteriorates by up to 9.8%, on average, if the number of feasible visit schedules (i.e., combinations of days on which customers can be visited) increases from three to ten.

2.5.2 Person-oriented consistency

The person who benefits from person-oriented consistency is either the customer or the driver: driver consistency is beneficial for customers; region and customer consistency decreases the learning burden and is, therefore, beneficial for drivers. The three notions of person-oriented consistency are very similar; in the best case scenario, they are even equivalent. Optimal driver consistency means that each customer is visited by the same driver each time. Optimal customer consistency is achieved by assigning each driver to the same set of customers repeatedly. Region consistency peaks when each driver is assigned to only one region; since each customer is located within one region, each customer request is handled by the same driver. In either case, each customer encounters only one driver, each driver performs all routes in the same region, and each driver repeats service to a specific customer a maximum number of times, i.e., driver, region, and customer consistency are optimal simultaneously.

However, person-oriented consistency measurements are differing in general, i.e., optimizing one measurement does not necessarily optimize another. This is the case when consistency has a lower priority compared to cost or when resources (e.g., vehicle capacity, number of drivers) are tight. Consider, for example, two drivers who share the same service region in order to increase routing efficiency. Drivers experience perfect region consistency (i.e., there is one region per driver) but customers can be visited by two different drivers. Smilowitz et al. [158] and Zhong et al. [190] assume that the learning effect of visiting a customer repeatedly decreases with increasing visit frequency, i.e., there is a convex relationship between the learning burden and the frequency of visiting a customer. In Smilowitz et al. [157], the authors prove that minimizing the average number of different drivers per customer (i.e., optimizing driver consistency) is not equivalent to minimizing the drivers' average learning burden of visiting customers (i.e., optimizing customer consistency). This proof is illustrated in Figure 2.16. Three customers require service on four days; the daily demand is given next to each customer. There are two drivers and two vehicles with a capacity of two units each; the routes of the first driver are denoted with solid arrows and the routes of the second driver with dashed arrows. Let α_i denote a driver's learning burden per visit if he visits a customer i times; $\alpha = (1, 0.14, 0.05, 0.02)$. Given the fluctuating demand and the tight vehicle capacity, only one customer can be visited by the same driver at each visit (in this example, customer one). The other customers (customers two and three) are visited by both drivers. At the bottom of the figure, we show two feasible solutions (Option 1 and Option 2) for day 4. Both solutions yield optimal driver consistency with an average number of drivers per customer of $(1+2+2)/3 = 1.66$. Yet, only Option 1 provides op-

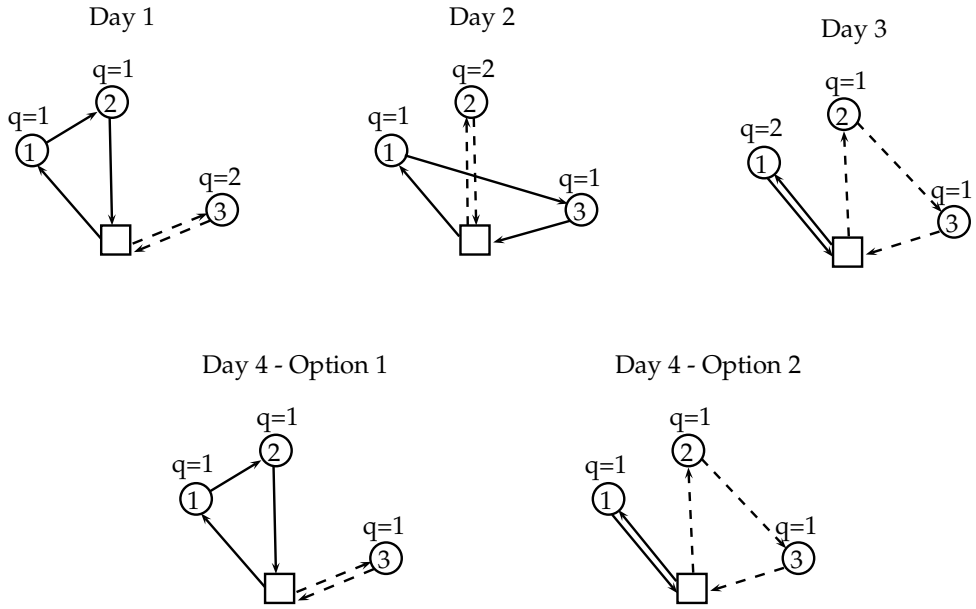


Figure 2.16 Minimizing the average number of different drivers per customer versus maximizing the average number of times the same driver is assigned to a customer (adapted from Smilowitz et al. [157]).

timal customer consistency. The two options differ only in the visit frequency at customer 2. With Option 2, customer 2 is visited once by driver one and three times by driver two; the learning burden at this customer is $1 * 1 + 3 * 0.05 = 1.15$. With Option 1, customer 2 is visited twice by both drivers; the leaning burden is $2 * 0.14 + 2 * 0.14 = 0.56$. Therefore, Option 1 is preferred to Option 2, if the goal is customer consistency.

The focus in the next three sections is on modeling, solving, and evaluating problems that consider person-oriented consistency.

Modeling person-oriented consistency

A common approach to achieve person-oriented consistency is to visit each customer with the same driver every time he requires a service (Coelho et al. [32], Coelho and Laporte [35], Day et al. [44], Francis et al. [66], Francis et al. [67], Groër et al. [75], Haughton [82], Kovacs et al. [107], Nowak et al. [129], Tarantilis et al. [170]). This constraint is relaxed in Erera et al. [57], Haughton [83], Kovacs et al. [103], Schneider et al. [152], Spliet [159] and Zhong et al. [190]. In Erera et al. [57], each customer may be visited by two different drivers; in Haughton [83] and Chapter 4 (Kovacs et al. [103]), the number of different drivers per customer is a user-defined parameter. In Chapter 5, we propose a multi-objective

approach where routing cost and the maximum number of different drivers per customer are independent objectives of the problem (Kovacs et al. [106]). In Spliet [159], each customer is assigned to one driver in advance; on each day, at least a given fraction of customers has to be visited by the pre-assigned driver. In Schneider et al. [152] and Zhong et al. [190], only a fraction of customers is assigned to the same driver permanently; unassigned customers may be visited by any driver in order to increase routing flexibility.

Hard constraints on the number of different drivers per customer have mostly been used in deterministic environments. Fixing the driver-customer assignment in advance in stochastic environments might render routes infeasible when the actual demand of the customers is larger than the vehicle capacity. Another concern raised by using tight constraints on consistency is the loss of customer satisfaction and driver productivity if the usual driver has to be replaced by an unfamiliar colleague (e.g., because of workforce fluctuation) (Haughton [83]).

Some authors improve person-oriented consistency by penalizing undesired driver-customer assignments (or rewarding desired assignments) in the objective function (Coelho et al. [32], Coelho and Laporte [34], Eveborn et al. [58], Kunkel and Schwind [110], Schneider et al. [151], Smilowitz et al. [158], Zhong et al. [190]). Person-oriented consistency in Sungur et al. [167] is improved implicitly by maximizing the similarity of the daily routes to a so-called master plan. Similarity between two routes is measured by counting the number of customers in a daily route that are within a given distance to at least one customer in the master plan route. This idea is illustrated in Figure 2.17. The left figure shows the master plan with three customers 1, 2, and 3. The right figure shows the daily route visiting customers 1, 2, 4, and 5. Customers 1, 2, and 5 are within a specified distance of at least one customer in the master plan (denoted by a dashed circle); therefore, the similarity between the two routes is three. In Eveborn et al. [58], Kunkel and Schwind [110], Schneider et al. [151], and Zhong et al. [190], existing customer-driver familiarity is incorporated into the solution process by specifying an individual assignment cost for each customer-driver pair.

In Feillet et al. [59], Haughton [83], and Smilowitz et al. [158], person-oriented consistency is improved in a post-processing step: First, a solution is generated by ignoring person-oriented consistency; then, each route is assigned to the driver who is most familiar with the customers on the respective route or the region through which the route passes.

Solving problems with person-oriented consistency considerations

Providing person-oriented consistency is an inherent feature of a priori routing and districting. Both approaches guarantee the highest possible person-oriented consistency, i.e., each

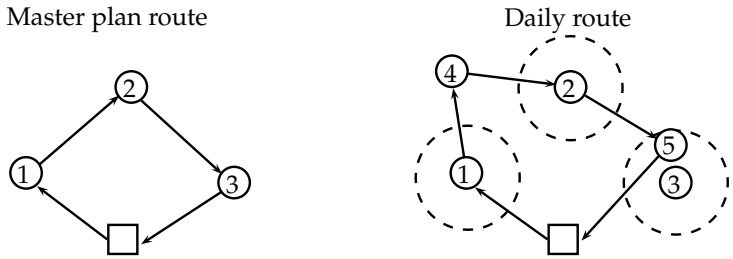


Figure 2.17 Similarity between daily route and master plan route according to Sungur et al. [167].

customer request is performed by the same driver. Therefore, many recent solution algorithms are based on either approach.

Districting approaches are proposed in Haughton [82], Haughton [83], Schneider et al. [152], and Zhong et al. [190]. Haughton [82] applies the traditional districting strategy, i.e., each customer is assigned to one region that is served by one driver. In Haughton [83], each region is served by a team of drivers. Schneider et al. [152] and Zhong et al. [190] associate each driver with a fixed service territory; however, service territories cover only a portion of the customers. Additionally, in Schneider et al. [152], assigned customers may be visited by a different driver if the number of vehicles can be reduced in this way.

A priori strategies are applied in Groër et al. [75], Kovacs et al. [107], Nowak et al. [129], Sungur et al. [167], and Tarantilis et al. [170]. The major challenge in generating a priori routes is to find the right balance between risky and safe routes. Figure 2.18 shows two extreme strategies. The planning horizon is three days and there is an unlimited number of vehicles with a capacity of four units each. Demand (q) is given for each customer on each day. The figure at the top shows a risky planning strategy; customer demand is underestimated when the a priori routes are planned and all customers are assigned to the same vehicle. The routing plan is efficient, but capacity is violated on day 3; the routes have to be rescheduled and at least one customer is assigned to a second driver. The figure at the bottom shows a conservative planning strategy; customer demand is overestimated and each customer is assigned to a separate a priori route. The resulting routing plan is feasible and highly consistent, but costly.

The proposed solution approaches mainly differ in the way they deal with the trade-off between risky, but low-cost routes and safe, but costly routes (i.e., how they estimate customer demand and how they allocate resources for planning the a priori routes).

The a priori approach in Groër et al. [75], Kovacs et al. [107], and Tarantilis et al. [170], for the ConVRP considers arrival time consistency and person-oriented consistency at the

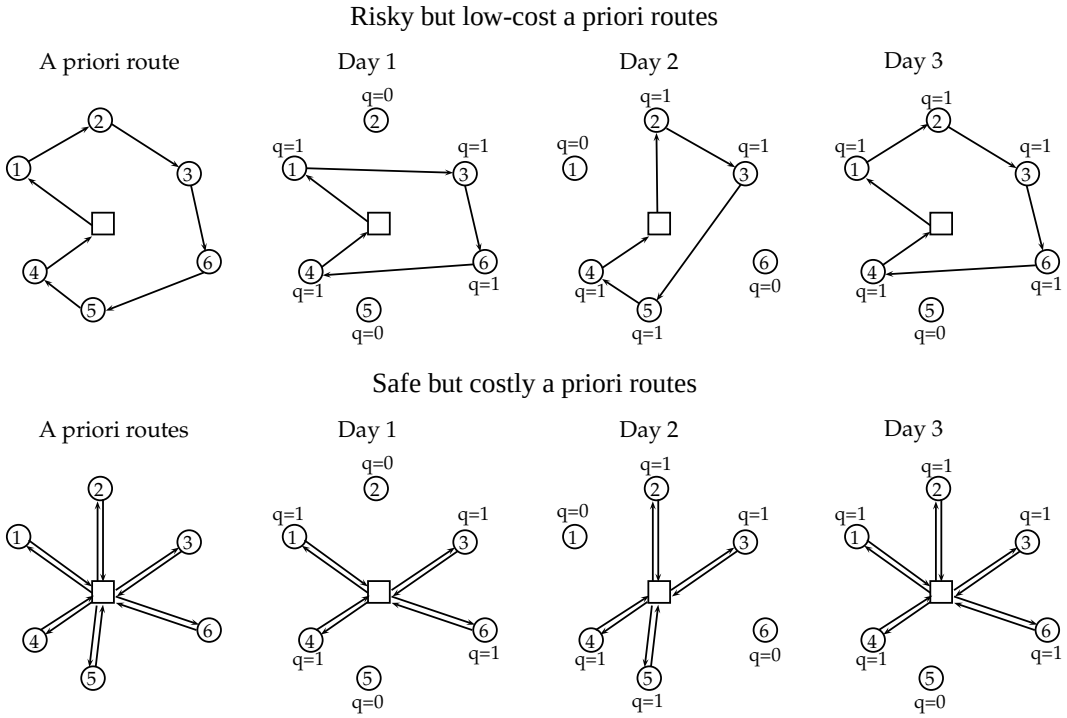


Figure 2.18 Two extreme strategies for planning a priori routes. Actual demand (q) is given for each customer. Vehicle capacity is four.

same time. The a priori routes (referred to as template routes) contain all customers that require service on two or more days during the planning period. Nowak et al. [129] consider a planning period of several months. Each customer requires a different visit frequency for periods of different length. All customers are included in the a priori routes; the service time of each customer used in the a priori routes is a function of the number of required visits per week, the period of the service (e.g., several weeks), the daily service time, and the length of the planning horizon. In Erera et al. [57], customers placing an order and their respective demands are random. Customers are partitioned into two groups: The first group includes customers that may be visited by any driver (e.g., customers with low probability of placing an order); the second group consists of customers that have to be served by no more than two different drivers (e.g., customers with high probability of placing an order). Each customer from group two is inserted into a primary a priori route and into a backup route. On each day, when the actual demand is revealed, the primary a priori routes are updated as follows: First, customers without demand are skipped; Second, routes are checked for capacity feasibility – if the demand on a route exceeds the vehicle capacity, some customers are moved to their backup route; finally, the partial routes are filled with customers from

group one. Customers requiring service are also random in Sungur et al. [167]. The a priori routes, referred to as master plan routes, contain customers with a high probability of placing an order. Master plan routes are updated by skipping customers without demand and by inserting unassigned customers into positions that improve similarity between the daily route and the underlying master plan route. Customers are left unassigned if their insertion would violate operational constraints. The updating strategies in Erera et al. [57] and Sungur et al. [167] involve a local search phase to improve the daily routes. So, in contrast to traditional a priori approaches, customers are not visited in the pre-planned order.

In Chapter 4, we propose an iterative solution approach that generates routes day-after-day. Infeasible driver-customer assignments are allowed during the search process, but they are penalized (Kovacs et al. [103]).

In Francis et al. [66], customers are visited by the same driver each time in the PVRP-SC. Additionally, the number of different routes each driver has to perform is limited by picking visit schedules that have a special property. Let S denote the set of feasible visit schedules and $|S|$ the number of elements in S; if S contains $|S|-1$ schedules without any common day and a schedule that is the union of these schedules, then the maximum number of different routes each driver has to perform is $|S|-1$. Consider the example illustrated in Figure 2.19. The planning period is five days and there are six customers visited by two drivers; customers can be assigned to one out of three visit schedules: two disjoint schedules, {1,3,5} and {2,4}, and their union, {1,2,3,4,5}. Customers 1 and 2 are visited three times (according to the first visit schedule), customers 3 and 4 twice (according to the second visit schedule), and customers 5 and 6 five times (according to the third visit schedule). The vehicle capacity permits two visits on each route. Each driver has to perform two different routes.

Evaluating the cost of person-oriented consistency

In Groër et al. [75], the average incremental cost of visiting customers by a single driver over the entire planning horizon is between 6.6% and 15% (the cost of arrival time consistency also contributes to this result); Haughton [82], report an average increase of 17.4%. For the IRP, Coelho et al. [32] and Coelho and Laporte [35], indicate an average increase between 0.31% and 9.85% and 0.55%, respectively. Feillet et al. [59], demonstrate cost savings of up to 7.5% when driver consistency is relaxed.

Enforcing driver consistency only for 75% of the customers decreases the average increase in travel cost from 12.7% to 2.9% (Spliet [159]). In Chapter 4 and 5, we demonstrate that travel cost decreases by up to 6.5% if customers may be visited by at least two different

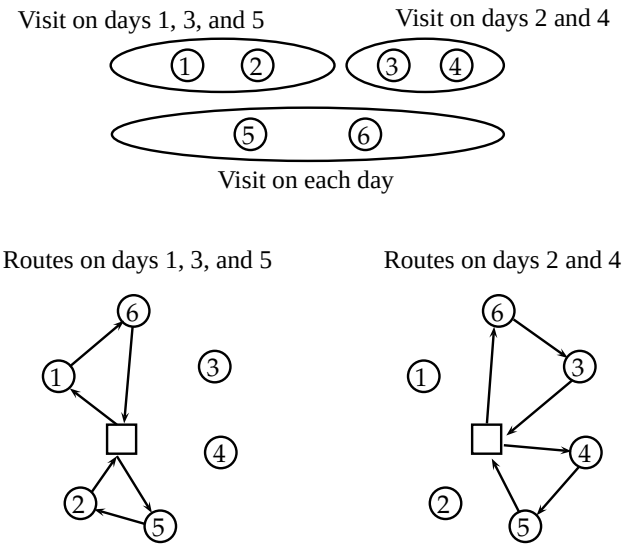


Figure 2.19 Reducing the number of different routes per driver by proper selection of visit day combinations. Each driver can visit two customers a day.

drivers per customer instead of only one; the cost savings obtained by allowing more than two drivers per customer is negligible (Kovacs et al. [103, 106]).

Coelho et al. [32], Coelho and Laporte [34] and Smilowitz et al. [158] incorporate person-oriented consistency measurements into the objective function. For the IRP, Coelho et al. [32] and Coelho and Laporte [34] report an increase in cost when consistency is optimized between 0.07% and 3.54% and between 0.07% and 6.71%, respectively. According to Francis et al. [67], the additional travel cost of visiting each customer by the same driver is less than 2% in the PVRP-SC. However, the average number of drivers per customer increases by up to 6.1% when person-oriented consistency is ignored; the average size of the region a driver has to be familiar with increases by up to 7.8%. In Smilowitz et al. [158], the authors show that even a small emphasis on person-oriented consistency significantly improves routing plan regularity with minimal impact on cost: Focusing on consistency increases routing cost by no more than 5.2%; in contrast, ignoring consistency reduces both customer and regional consistency by far more. This suggests that solution approaches devised for improving consistency perform better with regard to cost than cost-driven approaches with regard to consistency. For example, solution approaches that generate routes first and improve consistency by matching drivers to routes in a post-processing phase generate low cost routing plans (e.g., Feillet et al. [59] and Haughton [83]). Yet, minimizing cost often implies using fewer vehicles than would be needed to provide high consistency; therefore, there is less room for improvement in the post-processing phase.

2.5.3 Delivery consistency

Consistency in the quantity delivered or in the frequency of deliveries has mainly been modeled with hard constraints. In Coelho et al. [32], Coelho and Laporte [35], variations in the delivery quantities are restricted to be within a specific interval. The interval for each customer depends on the respective average daily demand. In the PVRP literature, consistency in delivery quantity is not considered explicitly, but it is assumed that the same quantity is delivered to each customer at each visit regardless of the interval between two consecutive visits (Francis et al. [66], Francis et al. [67]). The order-up-to-level policy is investigated in Bertazzi et al. [9], Coelho et al. [32], and Coelho and Laporte [35]. This policy requires that each customer's inventory be filled at each visit; the delivery quantity is equal to the difference between inventory capacity and the current inventory position. Here, the focus is rather on stable inventory levels than on stable delivery quantities.

Regularity in the visit spacing is achieved by bounding the minimum and maximum time between two consecutive visits (Coelho et al. [32], Coelho and Laporte [34], Gaudioso and Paletta [70]). In Day et al. [44], consistency in the timing of the delivery is achieved by assigning customers to cyclic visit schedules. Depending on the inventory capacity and the daily demand, customers are visited either daily, every second day, every third day, and so on. For example, a customer with an inventory of 500 units and a daily demand of roughly 150 units is visited in a three-day cycle.

The cost of improving delivery consistency is examined in Coelho et al. [32] and Coelho and Laporte [35]. Coelho et al. [32] report an increase in inventory and transportation cost of between 1.27% and 27.38%, on average, if the delivery amount is kept stable; in Coelho and Laporte [35], this cost increase is approximately 1.27%. The average increase in cost of enforcing the order-up-to-level policy is 8.6% according to Coelho and Laporte [35] and around 9% according to Coelho et al. [32]. In Coelho et al. [32], the average cost of regular visit intervals is estimated to be between 0.96% and 17.48%.

Francis et al. [67] compare two delivery strategies for the PVRP-SC: the first strategy is to provide each customer with his true demand at each visit (i.e., the demand accumulated since the last visit); the second strategy offers a choice between visiting each customer either with his true demand or with his average demand. The latter strategy is more flexible and decreases the cost (composed of routing, inventory holding, and shortages) between 3.6% and 4.8%. In all experiments, less than 10% of the customers are delivered using stable delivery quantities. So, in this case, cost is decreased by offering delivery consistency for some customers (even if the number of these customers is small).

The actual cost of improving delivery consistency depends on the input parameters such

as number of customers, length of the planning period, cost structure, and the required level of consistency.

2.6 Inconsistency

Service consistency improves customer satisfaction in a large number of applications. Nevertheless, there are applications in which consistency is unwanted. Consider, for example, security companies such as Brinks, G4S, and Loomis specializing in the transportation of valuable objects. Banknotes, coins, diamonds, and important documents are moved by armored vehicles on a regular basis (e.g., ATMs are filled daily, revenues of retailers are collected at the end of each day). Vehicle capacity is rarely restrictive; instead, the total value of goods per vehicle is limited. The vehicle routes must vary from day to day in order to reduce the predictability of service and, therefore, to reduce the risk of robbery (e.g., Michallet et al. [118], Ngueveu et al. [126], Ngueveu et al. [127], Yan et al. [187], Yan et al. [188]). Another example is introduced in Wolfler Calvo and Cordone [183]. A company offers overnight security service (e.g., patrol service and inspection of buildings and company grounds). Each night, guards visit the assigned customers on vehicle routes. Management puts emphasis on varying the nightly routes in order to decrease the predictability of visits. At the same time, guards are assigned to the same customers repeatedly to increase their familiarity with the assigned tasks (i.e., customer consistency is desired).

In Michallet et al. [118], Ngueveu et al. [126], and Wolfler Calvo and Cordone [183], the same set of customers is visited each day; the delivery quantities and the service times are steady and the objective is to minimize cost. If inconsistency is disregarded, efficient solution approaches will provide the same routing plan each day.

Typically, inconsistency in multi-period routing problems is achieved by altering the sequence in which customers are visited. An example with a planning horizon of two days and four customers is given in Figure 2.20. The figure at the left shows the solution on day 1; the figure in the middle shows the solution on day 2. Each route segment between two customers is used only once; the visit sequence changes from 0-1-2-3-4-0 to 0-2-4-1-3-0. As a result, each customer is visited at different times on day 1 and day 2 (shown in the right figure). The problem of finding routes for a planning horizon of m days that cover the same customers each day without using any road segment more than once is referred to as the m-Peripatetic Vehicle Routing Problem (m-PVRP) (Ngueveu et al. [126], Ngueveu et al. [127]). The m-PVRP generalizes the m-Peripatetic Salesman Problem (m-PSP) by introducing constraints on the vehicle capacity. The m-PSP was introduced in Krarup [108]

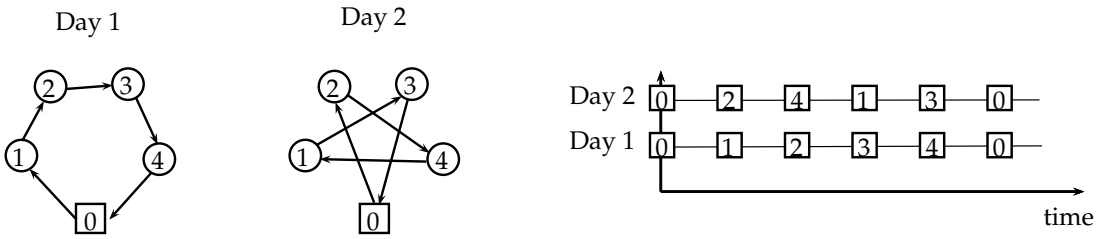


Figure 2.20 Inconsistency in the sequence in which customers are visited. Each customer is visited at different arrival times on day 1 and day 2.

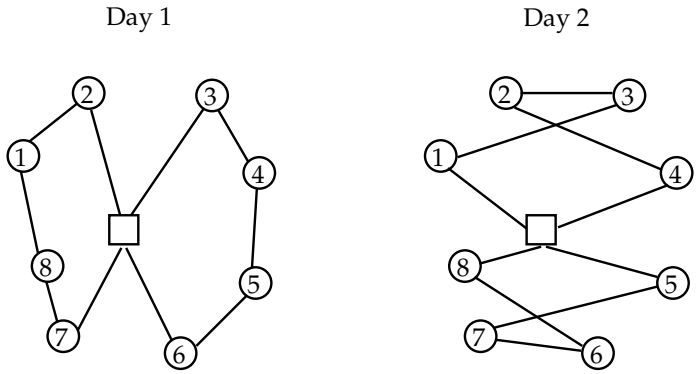


Figure 2.21 Example of a 2-PVRP. Each vehicle can visit four customers.

and further examined in, e.g., Ageev and Pyatkin [3], De Kort [47], De Kort [48], Duchenne et al. [52], and Duchenne et al. [53]; an example for a 2-PSP is given in Figure 2.20.

In any road network, the maximum number of "peripatetic" solutions is bounded. Let the road network be denoted by a complete undirected graph $G = (\{0\} \cup V, E)$; V is the set of customers, 0 the depot, and E the set of edges. The number of feasible edge disjoint solutions m is bounded by $m \leq \frac{|V|}{2\lambda}$ where λ is the minimum number of routes required per day (Ngueveu et al. [126]). Figure 2.21 shows an example of a 2-PVRP with eight customers. There are two vehicles with capacity for four customers each. Therefore, $m \leq \frac{8}{2*2}$.

Michallet et al. [118] and Wolfler Calvo and Cordone [183] argue that it is the inconsistency in arrival times that makes routes unpredictable and secure; the routing as such is less important. For example, in cash transportation, the risk of a robbery is highest when vehicles stand still while serving a customer (Michallet et al. [118]). Often, it is impossible to avoid unattractive routes if the goal is inconsistency in the customers' visit sequence. Duchenne et al. [52] demonstrate that the optimal cost of a 4-PSP can be up to 250% higher than four times the optimal TSP cost. Ngueveu et al. [126] indicate that the difference between m times the optimal VRP cost and the optimal m-PVRP increases with m. However,

inconsistency in the arrival times can be achieved with a modest increase in transportation cost. In Figure 2.20, for example, the route of day 1 can be repeated on day 2 by visiting customers in reverse order if the goal is inconsistency only in the arrival times. The resulting solution fulfills the inconsistency requirements without increasing cost (assuming a symmetric distance matrix).

In Michallet et al. [118], arrival time inconsistency is enforced by so-called time spread constraints; here, the arrival times at each customer over the planning period must differ by a given value. Vehicle idling is prohibited to avoid robberies. The time spread constraints are relaxed in Yan et al. [187, 188]. Routes are planned day after day; a routing plan is feasible if a given fraction of customers is visited outside of the prohibited time windows; the time windows are determined by the routing plans of the last couple of days. The solution approach in Wolfler Calvo and Cordone [183] first solves the single-period routing problem without considering arrival time inconsistency. Then, the working period is divided into four time windows of equal width and customers are partitioned into four groups according to their visit times; the customer-driver assignment is fixed in order to increase customer consistency. There are 24 (i.e., 4!) possibilities of assigning customer groups to time windows; the initial routing plan is one of them. The pool of inconsistent routing plans is extended by solving the single-period routing problem 23 more times, each time changing the assignment of customer groups to time windows. Each day, each driver picks one out of his 24 routing plans at random. The authors point out that despite the high diversity of the routing plans the average increase in travel time is less than 20% compared to the best known routing plan.

2.7 Summary

Service consistency was already addressed in the 1970's; yet, only in the last couple of years have authors investigated measurements that quantify the consistency of multi-period routing plans. In this chapter, we described early approaches that acknowledge service consistency as a side benefit. These approaches have been revisited in recent publications that target service consistency explicitly. We grouped these approaches into three categories: a priori approaches that derive daily routes from a set of preplanned routes, districting approaches that assign each customer to a fixed service territory serviced by a single driver, and approaches that ignore fluctuations in the demand but accept shortages and inventories at the customers.

We argued that service consistency can be achieved in three dimensions: First, reduc-

ing variations in the arrival times at the customers. Second, visiting customers in regular time intervals with similar delivery quantities. Third, visiting customers by the same driver repeatedly (for consistency from the customer's point of view) or assigning drivers to the same customers or regions as often as possible in order to increase their familiarity (for consistency from the driver's point of view). We reviewed the relevant literature and classified recent papers according to the considered dimensions of consistency. For each dimension, we have described how the problems have been modeled and solved. Additionally, we reported the increase in routing cost that is to be expected from focusing on service consistency.

In some applications, decision makers want to make the routing plans unpredictable in order to increase safety (e.g., in cash-transportation, vehicles and drivers are exposed to a large risk of robbery if vehicle routes can be anticipated). We also reviewed vehicle routing problems in which routes have to be different from day to day, i.e., consistency is unwanted.

In the next chapters, we present vehicle routing problems that combine traditional vehicle routing constraints with the requirements for service consistency. For each problem, we devise specialized solution algorithms and perform computational experiments in order to examine the increase in cost of improving service consistency.

Chapter 3

The consistent vehicle routing problem

3.1 Introduction

The consistent vehicle routing problem (ConVRP), as defined in Groër et al. [75], models services that are performed by companies in the small package shipping industry. The problem involves the construction of routes over a given time period, e.g., several days, such that customer demands are met. In addition to traditional vehicle routing constraints, the ConVRP also accounts for service consistency requirements. Consistency is achieved in two ways: First, in order to form a stronger relationship between the supplier and the customer, a customer can be assigned to only one driver. Second, to enable the customers to be prepared for a delivery, they have to be serviced at about the same time of the day. Consistency requirements link the days in the planning period together. Therefore, it is not possible to divide the multi-period problem into several single-period problems.

In Groër et al. [75], the ConVRP is solved by a template approach, i.e., template routes are generated in advance and the routing plan for each day is obtained by adapting the template according to the daily requirements. Tarantilis et al. [170] solve the ConVRP by a template-based tabu search algorithm. The algorithm is divided into two phases: templates are built in the first phase; daily routing plans are derived from the templates and then post-optimized in the second phase. Sungur et al. [167] address the courier delivery problem (CDP) that considers arrival time consistency and driver consistency; customer requests and service times are stochastic. The CDP integrates soft time windows for customer visits but driver consistency is not restricted to one driver per customer. The problem is solved by a master and daily scheduler heuristic (MADS). Similar to the template concept, MADS generates a master plan that can be converted into daily schedules with few modifications.

In this chapter, we present a new solution method for the ConVRP called template-based

adaptive large neighborhood search (TALNS). It combines the adaptive large neighborhood search (ALNS) by Ropke and Pisinger [145] with the template concept presented in Groër et al. [75]. ALNS is introduced by Ropke and Pisinger [145]. The algorithm is an extension of the large neighborhood search (LNS) by Shaw [156] and it is also related to the ruin and recreate principle by Schrimpf et al. [155]. ALNS has been applied successfully to solve different variants of the vehicle routing problem (Pisinger and Ropke [135], Ropke and Pisinger [144, 145]), the technician and task scheduling problem (Cordeau et al. [37]), and the service technician routing and scheduling problem (Kovacs et al. [105]). For a survey on large neighborhood search see Pisinger and Ropke [136]. Additionally, we introduce a relaxed variant of the ConVRP in which flexible departures times from the depot are permitted. Relaxing the starting times leads to improved arrival time consistency at minimum routing cost. We test our algorithm on available data sets from the literature and show that it is highly competitive. Based on the benchmark instances, we generate new data sets in which time consistency is modeled in a more realistic way. We provide heuristic solutions for these new data sets.

3.2 Problem definition

The ConVRP is defined on a complete directed graph $G = (N \cup \{0\}, A)$, where $N = \{1, ..., n\}$ is the set of customers and 0 is the depot; $N^0 = N \cup \{0\}$. Set $A = \{(i, j) : i, j \in N^0, i \neq j\}$ is the set of arcs. Customers are visited on routes traversed by a homogeneous fleet of vehicles in the set K. There is a sufficient number of vehicles (i.e., $|K| = |N|$). Each vehicle $k \in K$ is located at the depot from where it departs at time zero and where it must return to before time T; each vehicle has a capacity of Q. The planning horizon involves $|D|$ days where D is the set of days. On each day $d \in D$, each customer $i \in N$ has a given demand q_{id} and a given service time s_{id}. We use auxiliary parameters w_{id} equal to one if customer i requires service on day d ($q_{id} > 0$) and equal to zero, otherwise. Each arc $(i, j) \in A$ is associated with travel time t_{ij}. We assume that the travel time matrix complies with the triangle inequality. Each customer $i \in N$ must be assigned to the same vehicle over the entire planning period in order to improve service consistency. Furthermore, the difference between the earliest and the latest arrival time at each customer over all days, denoted by maximum arrival time difference l_{max}, is bounded by L. Vehicle idling to reduce the arrival time difference is not allowed.

The model uses following binary variables:

$$x_{ijkd} = \begin{cases} 1, \text{if arc } (i,j) \text{ is traversed by vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

$$y_{ikd} = \begin{cases} 1, \text{if customer } i \text{ is assigned to vehicle } k \text{ on day } d, \\ 0, \text{otherwise.} \end{cases}$$

The continuous variables a_{ikd} denote the arrival time at customer i by vehicle k on day d. Given this notation, the ConVRP can be formulated in the following model. The only modification to the original model of Groër et al. [75] is the extension of the a_{id} variables with index k to better comply with our applications.

$$\textit{Minimize} \quad \sum_{d\in D}\sum_{k\in K}\sum_{i\in N^0}\sum_{j\in N^0} t_{ij}x_{ijkd} \tag{3.1}$$

subject to:

$$y_{0kd} = 1 \qquad \forall\, k \in K, d \in D, \tag{3.2}$$

$$a_{0kd} = 0 \qquad \forall\, k \in K, d \in D, \tag{3.3}$$

$$\sum_{k\in K} y_{ikd} = w_{id} \qquad \forall\, i \in N, d \in D, \tag{3.4}$$

$$\sum_{i\in N} q_{id}y_{ikd} \le Q \qquad \forall\, k \in K, d \in D, \tag{3.5}$$

$$\sum_{i\in N^0} x_{ijkd} = \sum_{i\in N^0} x_{jikd} = y_{jkd} \qquad \forall\, j \in N^0, k \in K, d \in D, \tag{3.6}$$

$$w_{i\alpha} + w_{i\beta} - 2 \le y_{ik\alpha} - y_{ik\beta} \qquad \forall\, i \in N, k \in K, \alpha, \beta \in D, \alpha \ne \beta, \tag{3.7}$$

$$a_{ikd} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \le a_{jkd} \qquad \forall\, i \in N, j \in N, k \in K, d \in D, \tag{3.8}$$

$$a_{ikd} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \ge a_{jkd} \qquad \forall\, i \in N, j \in N, k \in K, d \in D, \tag{3.9}$$

$$a_{ikd} + w_{id}(s_{id} + t_{i0}) \le Tw_{id} \qquad \forall\, i \in N, k \in K, d \in D, \tag{3.10}$$

$$L - T(w_{i\alpha} + w_{i\beta} - 2) \ge a_{ik\alpha} - a_{ik\beta} \qquad \forall\, i \in N, k \in K, \alpha, \beta \in D, \alpha \ne \beta, \tag{3.11}$$

$$x_{ijkd} \in \{0,1\} \qquad \forall\, i, j \in N^0, k \in K, d \in D, \tag{3.12}$$

$$y_{ikd} \in \{0,1\} \qquad \forall\, i \in N, k \in K, d \in D, \tag{3.13}$$

$$a_{ikd} \ge 0 \qquad \forall\, i \in N, k \in K, d \in D. \tag{3.14}$$

The objective function (3.1) minimizes the total travel time. Inequalities (3.2) and (3.3) define that each route starts from the depot at time zero. Constraints (3.4) guarantee that each customer is serviced on each day he requires a service and inequalities (3.5) make sure that the vehicle capacity is not exceeded. Constraints (3.6) ensure that all assigned customers have exactly one predecessor and one successor. Driver consistency is guaranteed in (3.7). Inequalities (3.8) and (3.9) set the arrival times at the customers; vehicle idling to improve time consistency is not allowed. Inequalities (3.8) also prevent sub-tours. The maximum travel time is restricted by inequalities (3.10). Constraints (3.11) ensure that the arrival time difference for each customer is not greater than L. Constraints (3.12) - (3.14) define the domains of the decision variables.

3.3 Solution framework

We propose a template-based adaptive large neighborhood search (TALNS) for the ConVRP. Given an initial solution, ALNS integrates several destroy methods to repeatedly remove parts of a solution and several repair methods to rebuild the partial solution (Pisinger and Ropke [136]). In contrast to standard vehicle routing problems, we do not apply ALNS to the actual routing plan but to the template on the basis of which the routing plan is generated. In the following, we describe the template concept and the different design elements of the proposed TALNS.

3.3.1 The template concept

In the ConVRP, the daily routing plans are linked by the customers that require service on several days. To handle these interdependencies, we use the solution approach suggested by Groër et al. [75]. It relies on the generation of template routes from which the actual daily routes are derived. The template contains all customers that are relevant for consistency, i.e., customers that require service on at least two days. We refer to these customers as frequent customers and assign them to set N^f. The daily routes are obtained by resolving the template routes: First, excessive customers are removed. Excessive customers are those, that are considered in the template but do not require service on a particular day. Second, customers that request service only on one day are inserted on the corresponding day. This process is illustrated in following example: Assume a set of customers that require service over three days as given in Table 3.1. The columns give the customers from 1 to 8; the first three rows define whether a customer requests a service (value 1) or not (value 0) on

Customer	1	2	3	4	5	6	7	8
Day 1	1	0	1	0	0	1	0	1
Day 2	1	1	0	1	0	1	1	0
Day 3	0	1	0	0	1	1	1	1
Number of services	2	2	1	1	1	3	2	2
Template	1	1	0	0	0	1	1	1

Table 3.1 Demand pattern.

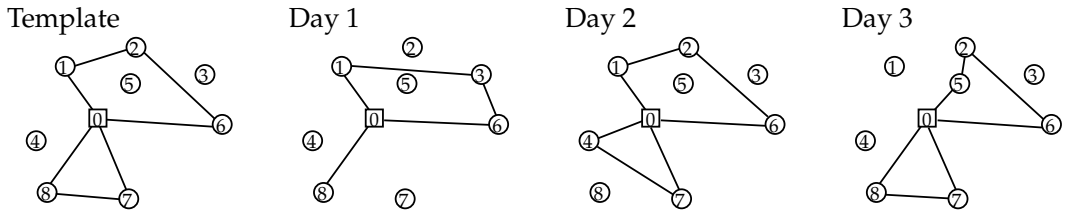


Figure 3.1 Example: template and corresponding solution.

the corresponding day. The last but one row sums the number of services over the entire planning horizon and the last row defines the customers that are considered in the template. Frequent customers are assigned to a template route (value 1); non-frequent customers are ignored (value 0). Each vehicle can visit at most three customers a day. A possible solution for our example is given in Figure 3.1. All frequent customers are served by two routes in the template. The routing plan for each day is derived from the template: On the first day, customers 2 and 7 are removed and customer 3 is inserted. The same approach is repeated on each day. The template concept guarantees driver consistency since the customer-driver assignment is not changed during the resolution of the template. Additionally, the sequence in which customers are visited in the template routes is transferred to each daily route. The resulting precedence principle supports arrival time consistency.

In order to create template routes that can be transformed into a feasible solution with low travel time, we incorporate daily demand information into the template construction. First, for each frequent customer, we compute an artificial demand q_{a_i} and an artificial service time s_{a_i}. These values are used throughout the template construction. In contrast to Groër et al. [75], they are defined as the mean of the original demand and service time over all days in D and not only the days a customer requests service. Therefore, our variant implicitly considers the service frequency of each customer; q_{a_i} and s_{a_i} are computed as follows:

$$q_{a_i} = \frac{\sum_{d \in D} q_{id}}{|D|} \; \forall i \in N^f, \tag{3.15}$$

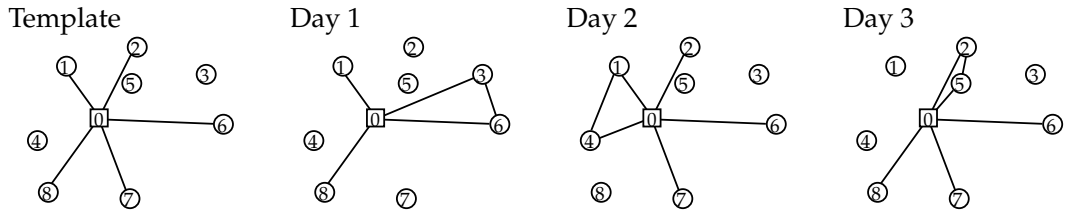


Figure 3.2 Artificial constraints, T_a and Q_a, are set to small values. The solution is feasible but the total travel time is very high.

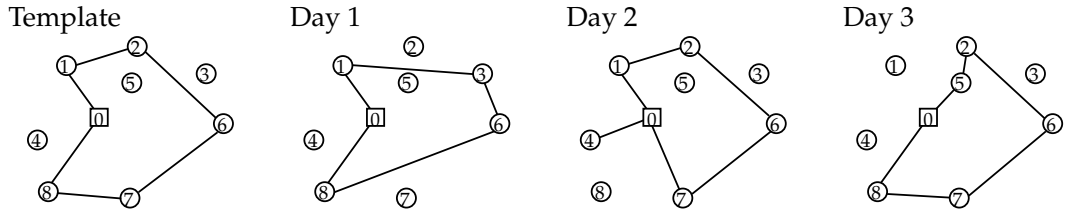


Figure 3.3 Artificial constraints, T_a and Q_a, are set to large values. The solution has a short total travel time but it might violate the maximum travel time T, the maximum capacity Q, or the time consistency requirement L.

$$s_{ai} = \frac{\sum_{d \in D} s_{id}}{|D|} \; \forall i \in N^f. \tag{3.16}$$

Second, we use an artificial maximum tour length (T_a) and an artificial vehicle capacity (Q_a) when constructing the template. The constraints control the number of routes in the template and, therefore, the number of routes in the daily schedules. Each customer will be visited on a separate route in the template and in the corresponding solution if T_a and Q_a are tight. Such a solution would always be feasible with perfect arrival time consistency ($l_{max} = 0$). However, the total travel time would be very large. Loose T_a and Q_a values lead to long template routes. Given the triangle inequality, long routes have shorter travel times. Yet, when the template is resolved, it is more likely that the resulting solution violates the real T and Q constraints and the time consistency requirement. The effect of using either tight or loose artificial constraints is illustrated in Figure 3.2 and Figure 3.3, respectively.

We use different T_a and Q_a values every time a new template is generated in order to diversify the search. The way we set the values deviates from the approach by Groër et al. [75]. In Groër et al. [75], an estimate for both artificial constraints is made on the basis of the mean number of visits per day. Depending on whether or not the obtained solution is feasible, the constraints are either relaxed or tightened. We define upper and lower bounds for both, T_a and Q_a, and search the interval between the bounds. The upper bounds, $\overline{Q_a}$

and $\overline{T_a}$, are calculated as follows: First, for each day d, two lower bounds on the number of vehicles are defined: one with regard to tour length and one with regard to capacity. They are denoted by $\underline{NV_{Td}}$ and $\underline{NV_{Q_d}}$, respectively. The lower bounds with regard to capacity are defined by

$$\underline{NV_{Q_d}} = \left\lceil \frac{\sum_{i \in N^f} q_{id}}{Q} \right\rceil \quad \forall d \in D. \tag{3.17}$$

In order to compute the daily lower bounds that are based on the maximum tour length ($\underline{NV_{Td}}$), we first estimate the travel time on each day. This is done by applying Kruskal's algorithm (Kruskal [109]) to define the minimal spanning tree (MST) among all frequent customers requesting service on day d and the depot. We use the property that the cost of the MST on day d, denoted by $f(MST_d)$, plus the cost of the shortest edge from the depot to any customer is a lower bound for the cost of the optimal TSP solution and, therefore, also a lower bound for the cost of the optimal VRP solution. By adding the aggregated service times on day d, we get a lower bound for the total duration of all routes. Dividing this value by the maximum tour length yields a lower bound for the number of vehicles $\underline{NV_{Td}}$:

$$\underline{NV_{Td}} = \left\lceil \frac{f(MST_d) + \min_{i \in N} t_{0i} + \sum_{i \in N^f} s_{id}}{T} \right\rceil \quad \forall d \in D. \tag{3.18}$$

Let $f(MST_{template})$ be the cost of the MST among all frequent customers $i \in N^f$ and the depot. By using artificial demands q_{a_i} and service times s_{a_i}, and bounds $\underline{NV_{Q_d}}$ and $\underline{NV_{Td}}$, we compute $\overline{Q_a}$ and $\overline{T_a}$ as follows:

$$\overline{Q_a} = \frac{\sum_{i \in N^f} q_{ai}}{\max_{d \in D} \underline{NV_{Q_d}} - (1-\varepsilon)}, \tag{3.19}$$

$$\overline{T_a} = \frac{f(MST_{template}) + \min_{i \in N} t_{0i} + \sum_{i \in N^f} s_{ai}}{\max_{d \in D} \underline{NV_{Td}} - (1-\varepsilon)}. \tag{3.20}$$

Using artificial constraints $\overline{Q_a}$ and $\overline{T_a}$ during the template generation, guarantees that the number of routes in the template is never less than defined by (3.19) and (3.20), respectively. To avoid solutions with more vehicles than the optimal number of vehicles, we reduce the maximum of the daily lower bounds by a value that is asymptotic to one. This value is expressed by $1-\varepsilon$, where ε is a small number. In our experiments, ε is set to 10^{-4}.

Non-frequent customers are not considered in the calculation of $\underline{NV_{Q_d}}$ and $\underline{NV_{Td}}$. Including all customers in the calculation would lead to lower $\overline{Q_a}$ and $\overline{T_a}$ values and to more template routes. This approach would prevent solutions in which frequent customers are served by the minimum number of routes and all non-frequent customers are served on sep-

arate routes.

Lower bounds, $\underline{Q_a}$ and $\underline{T_a}$, are set to the artificial constraint values that yield the first feasible solution.

3.3.2 Constructing an initial solution

The initial template routes are built by a construction heuristic. A template is accepted as the initial one if it results in a solution that satisfies the capacity and the tour length constraints; arrival time consistency may be violated. It is much harder to comply with the arrival time consistency requirement as it is only considered implicitly by the template concept.

Artificial constraints T_a and Q_a are initially set to the respective upper bounds (equations (3.19) and (3.20)). Then, we iteratively generate and resolve different templates by tightening the artificial tour length and capacity constraints until an initial template is found: T_a and Q_a are tightened by 1% each time the respective constraint is violated.

During the construction phase, we use a greedy approach (Pisinger and Ropke [135], Ropke and Pisinger [145]) to insert customers into the template. The heuristic inserts customers one after another into their cheapest position until all customers have been assigned. Every feasible insertion position is checked for every unscheduled customer. The customer who causes the least increase in the total travel time is inserted into his best position. Let c_{ik} represent the change in the total travel time if customer i is inserted at his cheapest position into route k. For customers that cannot be inserted into feasible positions, we set c_{ik} to infinity. In each iteration, customer i^* is inserted into template route k^* for which

$$(i^*, k^*) := \arg \min_{i \in N^f, k \in K} c_{ik}. \tag{3.21}$$

The same approach is used to insert non-frequent customers during the resolution of the template. Here, non-frequent customer i^* is inserted at his best position into route k^* for which

$$(i^*, k^*) := \arg \min_{i \in N \setminus N^f, k \in K} c_{ik}. \tag{3.22}$$

We add an empty route if there is no feasible insertion position in the existing routes.

Feasibility refers to the artificial capacity and tour length constraints during the template construction and to the real constraints during the resolution. Arrival time consistency is checked only when the routing plan is completed.

3.3.3 Template-based adaptive large neighborhood search

We use the TALNS to improve the initial template and consequently the initial solution. TALNS integrates several operators that are used to repeatedly destroy and repair the current template (Pisinger and Ropke [135], Ropke and Pisinger [144, 145]).

The pseudocode of the TALNS is presented in Algorithm 1. In each iteration, one destroy and repair operator pair is selected based on its performance in past iterations (line 3). The chosen pair is applied to generate a new template (line 5). The destroy operator removes assigned customers from the template routes. The removed customers are reinserted into the template by using the according repair operator. The reinsertion is performed with regard to artificial constraints T_a and Q_a (line 4). The new template is resolved (line 6), i.e, excessive customers are removed and non-frequent customers are inserted as described in Section 3.3.2. The decision to move to the new template (respectively solution) or not is made by a simulated annealing acceptance criterion (line 7).

In case the initial solution violates the time consistency constraint, the first priority is to find the first feasible solution. Therefore, we tighten the artificial template constraints and force the template routes to become shorter. We reduce both, T_a and Q_a, by 1% every 50^{th} iteration until we find a solution that complies with all constraints. The artificial constraints that led to the first feasible solution are set as lower bounds, $\underline{T_a}$ and $\underline{Q_a}$, for T_a and Q_a.

In the remaining iterations, the artificial limits T_a and Q_a are selected randomly between the corresponding lower and upper bounds (line 4):

$$Q_a = \underline{Q_a} + y(\overline{Q_a} - \underline{Q_a}), \tag{3.23}$$

$$T_a = \underline{T_a} + y(\overline{T_a} - \underline{T_a}). \tag{3.24}$$

Random number y is chosen in the interval [0,1]. The last step is to update the performance of the applied destroy and repair operator pair (line 14). The TALNS terminates as soon as a given number of iterations is reached.

Acceptance criterion

The simulated annealing acceptance criterion (Kirkpatrick et al. [97]) is used to decide whether or not a new template τ' that produces solution s' should become the current incumbent template τ. We accept a new template τ' when the corresponding solution's objective value $f(s')$ is lower than that of the current incumbent solution $f(s)$. Inferior solutions

Algorithm 1 *TALNS*

Require: initial template τ and corresponding solution s ▷ Section 3.3.2
$s^{best} = s$ ▷ $f(s) = \infty$ if s is infeasible
repeat
  select a pair of destroy and repair operators (d, r) ▷ Section 3.3.3
  select $T_a \in [\underline{T_a}, \overline{T_a}]$ and $Q_a \in [\underline{Q_a}, \overline{Q_a}]$ ▷ Equations (3.23) and (3.24)
  τ' = generateNewTemplate(τ, d, r, T_a, Q_a)
  s' = resolveTemplate(τ')
  if accept(s',s) **then** ▷ Section 3.3.3
    $\tau = \tau'$
    $s = s'$
  end if
  if $f(s') < f(s^{best})$ **then**
    $s^{best} = s'$
  end if
  update joint performance of operators d and r ▷ Section 3.3.3
until maximum number of iterations is reached
return s^{best}

are accepted with probability $e^{-(f(s')-f(s))/\hat{t}}$. Parameter $\hat{t}$ is the current temperature; it is initialized only when the first feasible solution (with respect to all constraints) is found:

$$\hat{t} = -\frac{w_{\hat{t}}}{\ln 0.5} f(s) \tag{3.25}$$

Based on Pisinger and Ropke [135] and Ropke and Pisinger [145], we set $\hat{t}$ such that a $w_{\hat{t}}$% worse solution is accepted with a probability of 50%; $w_{\hat{t}}$ is a parameter to be determined. The temperature is decreased by the geometric annealing schedule, $\hat{t} = \hat{t}c$; c denotes the cooling rate.

Infeasible solutions are rejected until $\hat{t}$ has been initialized (i.e., $f(s')$ is set to infinity). Afterwards, we tolerate violations of the arrival time consistency constraint. The probability of accepting solutions with $l_{max} > L$ is based on an artificial objective function $g(s)$:

$$g(s) = f(s) + (l_{max} - L)e^{(\frac{i_{TALNS}}{\delta} - p_{penalty})}. \tag{3.26}$$

The penalty is controlled by parameters $p_{penalty}$, δ, and the current number of TALNS iteration i_{TALNS}. The higher i_{TALNS} (i.e., the further the search process), the lower the probability of accepting infeasible solutions. Setting $\delta = 6000$ was found to work well during the implementation of the algorithm.

The best found solution s^{best} is replaced if s' is feasible and $f(s') < f(s^{best})$.

Selection of destroy and repair operators

In each iteration of the TALNS, we generate a new template by applying one destroy operator d, and one repair operator r. The selection of these operators is performed by the adaptive selection mechanism proposed by Pisinger and Ropke [135]. In our case, however, the selection and remuneration of the destroy and repair operators are done pairwise rather than separately. Slightly better results are achieved with this selection mechanism in Kovacs et al. [105]. We record the joint performance of the operator pairs and assign weights ρ_{dr} according to their past performance. The better the results obtained by a pair, the higher the weight and, therefore, the probability ϕ_{dr} for being chosen in future selections. We define the selection probabilities as follows:

$$\phi_{dr} = \frac{\rho_{dr}}{\sum_{d=1}^{\eta_d} \sum_{r=1}^{\eta_r} \rho_{dr}}. \tag{3.27}$$

The number of available destroy and repair operators is η_d and η_r, respectively. Roulette wheel selection is used to choose one pair in each iteration.

Pairs can earn scores, ψ_{dr}, each time they are applied. The scores are collected during time segments of 100 iterations according to the following pattern:

$$\psi_{dr} = \begin{cases} \psi_{dr} + \sigma_1, & \text{if the destroy-repair pair yielded a solution that} \\ & \text{improved the global best solution } s^{best}; \\ \psi_{dr} + \sigma_2, & \text{if the destroy-repair pair yielded a solution that} \\ & \text{was not visited before and improved the incumbent solution } s; \\ \psi_{dr} + \sigma_3, & \text{if the destroy-repair pair yielded a solution that was not visited before} \\ & \text{and was accepted as the incumbent solution } s, \text{ although it was worse;} \\ \psi_{dr}, & \text{otherwise.} \end{cases}$$

Parameters σ_1, σ_2 and σ_3 are defined by the user. To find out whether or not a solution was visited before, we store the objective values of previously found solutions in a hash table.

Initially, weights ρ_{dr} are set to one and the scores ψ_{dr} are set to zero. After each time segment, the weights are updated and all ψ_{dr} values are reset to zero:

$$\rho_{dr} = p_{react} \frac{\psi_{dr}}{\max(1, \Theta_{dr})} + (1 - p_{react}) \rho_{dr}. \tag{3.28}$$

The influence of new solutions on the weights is controlled by the reaction factor p_{react}; Θ_{dr} counts how often pair (d,r) was selected in the current segment.

Destroy operators

Before the chosen destroy operator can be applied, we have to determine the number of customers to remove, u. We choose u randomly in the interval $[\min\{0.1|N^f|,30\},\min\{0.4|N^f|,60\}]$ as suggested in Pisinger and Ropke [135].

In the following sections, the applied destroy operators are presented and their characteristics to work on different types of neighborhoods are pointed out. They are all based on Pisinger and Ropke [135], Ropke and Pisinger [145], and Ropke and Pisinger [144].

Random removal The random removal operator randomly removes customers from the template routes. This is done repeatedly until u customers have been removed. The aim of the operator is to diversify the search.

Worst removal The worst removal operator iteratively removes customers that contribute the most to the template's total travel time. The idea is to favor the reinsertion of customers at cheaper insertion positions than the current ones. Let h and j be the predecessor and the successor of customer i, respectively. Then,

$$d(i,\tau) = t_{hi} + t_{ij} - t_{hj} \tag{3.29}$$

represents the saving that is obtained by temporarily removing customer i from the template τ. The removal is randomized to avoid outlying customers to be removed over and over again. The $d(i,\tau)$ values are sorted in list L in decreasing order. In each iteration, customer $i := L[y^{p_{worst}}|L|]$ is removed; y is a random number in the interval $[0,1)$ and p_{worst} is the parameter that controls the impact of randomization. The $d(i,\tau)$ values are updated and customers are removed until u customers have been removed.

Related removal The related removal operator (Shaw [156]) is based on the observation that it is easier to interchange customers within a schedule when they are somehow related. The relatedness $\mathscr{R}(i,j)$ between two customers i and j is composed of geographical vicinity, t_{ij}, difference in demand, q_{ai}, and similarity in visit frequency, γ_{ij}. The smaller $\mathscr{R}(i,j)$, the higher the similarity between two customers. The impact of each measure is controlled by

parameters λ, μ, and ν, respectively.

$$\mathscr{R}(i,j) = \lambda t_{ij} + \mu |q_{a_i} - q_{a_j}| - \nu \gamma_{ij} \tag{3.30}$$

Relatedness in terms of visit frequency (γ_{ij}) is characterized by the number of days on which either both customers i and j are serviced or neither of them.

$$\gamma_{ij} = |D| - \sum_{d \in D} \left| w_{id} - w_{jd} \right| \tag{3.31}$$

Including γ_{ij} in $\mathscr{R}(i,j)$ does not favor the interchangeability of customers in the template. Yet, it supports the grouping of similar customers which again supports the compliance with the arrival time consistency requirement when the template is resolved. The procedure is initialized by removing a randomly chosen customer from the template and inserting it into the set of removed customers S. In each iteration, one customer is chosen randomly from S to calculate the $\mathscr{R}(i,j)$ values. The removal operator is randomized to obtain a certain degree of diversification. Therefore, all $\mathscr{R}(i,j)$ values are sorted in list L in increasing order, and customer $i := L[y^{p_{related}}|L|]$ is removed from the template and added to S. Again, y is a random number in the interval $[0,1)$ and $p_{related}$ is the parameter that controls the impact of randomization. The removal is repeated until u customers have been removed.

Cluster removal The cluster removal operator (Ropke and Pisinger [144]) shares the idea of removing related customers; so, it can be interpreted as a variant of the related removal operator. However, relatedness is defined only in terms of geographical distance. The cluster removal operator removes complete clusters of customers even if u would be exceeded. Clusters are identified by computing the minimal spanning tree among all customers in the same template route[1] (Kruskal [109]). By deleting the longest arc of the tree, we obtain two clusters. One of them is selected randomly and all customers belonging to that cluster are removed. This approach is illustrated in Figure 3.4.

The first route is selected randomly. The following routes are selected as follows: A customer from the currently removed cluster is chosen randomly. The route that contains the closest customer to the previously selected customer and has at least three customers is destroyed next. The procedure is continued either until at least u customers have been removed or no more routes with at least three customers exist.

[1]The ConVRP is defined on a digraph. We convert the digraph into an undirected graph by setting $t_{ij} = t_{ji} = \min\{t_{ij}, t_{ji}\}$.

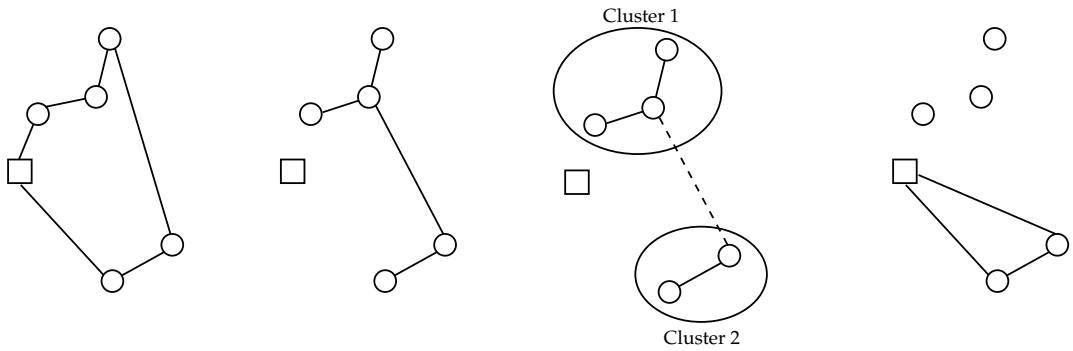


Figure 3.4 Cluster removal operator.

Repair operators

After customers have been removed, we use the greedy heuristic and four variants of the regret heuristic (Pisinger and Ropke [135], Ropke and Pisinger [144, 145]) to reinsert all removed customers back into the template routes. The operators work at the template level. So, only the artificial tour length and capacity constraints T_a and Q_a are considered during the repair phase.

Greedy heuristics The greedy heuristic for constructing the initial template is used as a repair operator. It inserts customers, one after another, until all frequent customers are inserted. For each unassigned customer $\in N^f$, we check each feasible insertion position and assign the customer who can be inserted the cheapest into his cheapest position. For further details see Section 3.3.2.

Regret heuristics Similar to the greedy approach, the regret heuristic (Potvin and Rousseau [137]) inserts frequent customers one after another by checking every feasible insertion position. However, it includes a look ahead component denoted by regret. This value represents the possible loss that may arise if a customer's insertion is postponed to later iterations. In the basic variant of the heuristic, the customer with the largest difference between inserting him into his best route at the best position and inserting him into his second best route at the best position is inserted in each iteration. This concept is extendable to consider more than two routes (Pisinger and Ropke [135], Ropke and Pisinger [145]). So, difficulties in future insertions can be identified earlier. Let c_i^q denote the change in the total travel time for inserting customer i at his cheapest position in his q-cheapest route. If

it is not possible to insert a customer into a route, c_i^q is set to infinity. In each iteration, the customer i^* to be inserted is given by:

$$i^* := \arg\max_{i \in N^f} \left\{ \sum_{h=2}^{\min(q,m)} (c_i^h - c_i^1) \right\} \tag{3.32}$$

Parameter q defines the number of routes considered in the current regret heuristic variant and m is the number of currently available routes. An empty route is added whenever it is not possible to assign further customers.

Following Pisinger and Ropke [136], we use four regret heuristics with $q \in \{2,3,4,m\}$.

3.3.4 Further improvements

Due to the interdependencies between the days in the ConVRP, it is difficult to predict reasonable insertion positions based on partial solutions. Therefore, we randomize the route construction as suggested by Ropke and Pisinger [145]. The randomization is done by drawing a random number y in the interval $[-\eta \max_{i,j \in N} t_{ij}, \eta \max_{i,j \in N} t_{ij}]$ and adding it to the insertion cost of each customer. Parameter η controls the amount of randomization. Whether a repair operator inserts customers according to c or $c+y$ into the template, depends on the past performance of the heuristics with and without noise. The decision is made by the adaptive selection mechanism as described in Section 3.3.3. We apply noise with a probability of 50% when inserting non-frequent customers into the routing plan.

The TALNS puts emphasis on constructing minimal cost template routes and relies on the template concept to achieve service consistency. However, a low cost template does not always result in a low cost routing plan. This is especially true when only a small portion of the customers is considered in the template. Also, the template concept might perform poorly when the majority of the customers is inserted into the template but many customers have to be deleted from the routes during the resolution. This issue increases with longer planning periods.

We apply a truncated 2-opt operator (Lin [116]) to the daily routes each time we obtain a solution that complies with the tour length and the capacity restrictions. Truncated means that only sequences with a maximum of three customers may be reversed. The time consistency requirement prevents the reversal of long sequences. Additionally, a lot of computational time can be saved by this restriction.

The effect of this operator is twofold: First, the routes do not comply with the precedence principle after applying the 2-opt operator, i.e., customers $\in N^f$ are not visited in the

same order on each day. Experiments by Groër et al. [75] for the ConVRP show that in 2 out of 10 instances deviations from the precedence principle are needed to obtain the optimal solution. Second, holes caused by the removal of excessive customers may be filled by moving non-frequent customers to different positions in the near neighborhood.

The 2-opt operator accepts only feasible modifications. Therefore, besides improving the objective function value, it may also repair previously infeasible solutions.

In contrast to the TALNS, the consistent record-to-record solution approach presented in Groër et al. [75] operates only at the template level and does not include any post optimization, i.e., the solutions comply with the precedence principle of the template. The template-based tabu search of Tarantilis et al. [170] integrates a post-optimization phase that is executed each time a new template is resolved. Different neighborhood operators are used to move non-frequent customers to different positions; frequent customers are not moved. So, the precedence principle of the underlying template is not altered either.

3.4 The ConVRP with shiftable starting times

In this section, we propose a modification of the original ConVRP model and two solution approaches. In the ConVRP, all vehicles have to leave the depot at time zero and waiting times between customer visits are not allowed (see inequalities (3.3) and (3.9) in Section 3.2). The effect of these constraints is illustrated in following example: Two customers, 1 and 2, with service time zero have to be serviced over a planning period of three days. The maximum arrival time difference is bounded by one ($L = 1$). Tour length and capacity is unbounded. Both customers require service on the first day, on the second day only customer 1, and on the third day only customer 2. All possible connections require a travel time of one time unit. The optimal solution with one vehicle is presented in Figure 3.5.

If vehicles leave the depot at time zero, the obtained solution results in a maximum arrival time difference of one ($l_{max} = 1$) and the total travel time (TT) is 7. If L was set to a value smaller than one, two vehicles, one for each customer, would be needed to solve the example. The resulting cost would be $TT = 8$. The departure time from the depot may be delayed in Figure 3.6. The vehicle on the third day departs at time one which results in a solution with $l_{max} = 0$.

In our relaxed ConVRP variant, the constraint for the starting times to be zero is omitted (inequality (3.3)), i.e., a later departure from the depot is allowed. Solutions that are infeasible because of the arrival time consistency requirement can be repaired by adjusting the departure time of each vehicle and, therefore, the arrival times at the customers. This

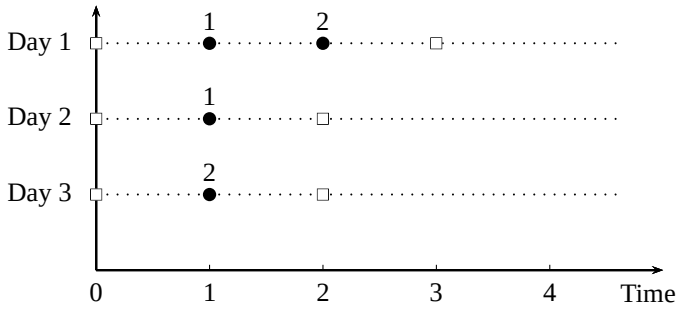


Figure 3.5 Departure time from depot is zero, $l_{max} = 1$.

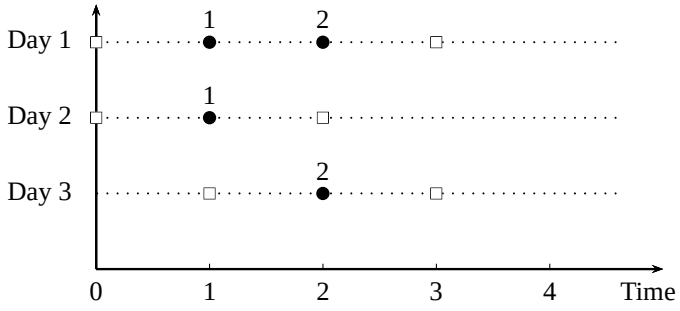


Figure 3.6 Departure time from depot is flexible, $l_{max} = 0$.

concept is integrated into the solution method by calling a repair mechanism each time the TALNS produces a solution that violates the arrival time consistency requirement. The repair mechanism tries to make the routing plan feasible by shifting the departure times from the depot. The template routes are not affected by the repair mechanism.

In the next two sections, we propose an exact and a heuristic repair approach.

3.4.1 An exact repair approach

To optimize the starting times of each route on each day (a_{0kd}), we solve following LP model with IBM's ILOG Cplex 12.1.

$$Minimize \quad l_{max} \tag{3.33}$$

subject to:

$$(a_{i\alpha} - a_{i\beta}) w_{i\alpha} w_{i\beta} \leq l_{max} \qquad \forall\, i \in N^f, \alpha, \beta \in D, \alpha \neq \beta, \tag{3.34}$$

$$a_{ikd} + s_{id} + t_{ij} = a_{jkd} \qquad \forall\, (i,j) \in A_{kd}, j \neq 0, d \in D, k \in K, \tag{3.35}$$

$$a_{ikd} + w_{id}(s_{id} + t_{i0}) \leq T w_{id} \qquad \forall\, i \in N, k \in K, d \in D, \tag{3.36}$$

$$a_{ikd} \geq 0 \qquad \forall\, i \in N, k \in K, d \in D. \tag{3.37}$$

The objective (3.33) is to minimize the maximum arrival time difference, l_{max}. The arrival time difference for each frequent customer is defined by constraints (3.34). Equations (3.35) (derived from (3.8) and (3.9)) link the arrival times of consecutive customers; set A_{kd} gives the arcs traversed by vehicle k on day d. Constraints (3.36) ensure that each vehicles returns to the depot before time T. To avoid deliveries beyond usual business hours, we do not bound the tour duration but the time interval in which a tour can be executed. Non-negativity is stated in constraints (3.37).

3.4.2 A heuristic approach

Our heuristic repair mechanism iteratively identifies the customer i^* with the largest difference in his arrival times (computed as in constraints (3.34)). The day when customer $i^{*\prime}s$ service starts earliest is denoted by α and the day when $i^{*\prime}s$ visit starts the latest is denoted by β. If $i^{*\prime}s$ route k on day β is already delayed, we first try to reduce that delay by decreasing the departure time from the depot as follows:

$$a_{0k\beta} = a_{0k\beta} - (l_{max} - L) \tag{3.38}$$

If this is not possible because of the non-negativity constraints (3.37), we try to delay the departure time of vehicle k on day α:

$$a_{0k\alpha} = a_{0k\alpha} + (l_{max} - L) \tag{3.39}$$

The departure times are shifted until the routing plan is either feasible ($l_{max} \leq L$), l_{max} is not decreasing from one iteration to the next, or when the latest return to the depot (T) would be exceeded.

3.5 Computational results

The TALNS is implemented in C++ and run on Intel Xeon X5550 computers with 2.67GHz. Results are obtained by running the algorithm with 30000 iterations ten times per instance. All results reported in the following sections are average values over these ten runs.

3.5.1 Data sets

The data set used to test the described algorithm is proposed by Groër et al. [75]. It is based on the Christofides benchmark instances for the vehicle routing problem (Christofides and Eilon [31]) and consists of 12 instances with a planning horizon of five days each. The number of customers is between 50 and 199 and the visit frequency is 70%. The visit frequency is the probability with which requests were assigned to customers and days when generating the instances. We refer to this data set as data set *A*.

For more extensive testing, we extend the original benchmark data set and generate two modified sets. The modifications concern the visit frequency, which is set to 50% and 90%, respectively. The demands and the service times are set to their original values. The lower the visit frequency in the instance generation, the higher the probability that a customer has no request at all. We assigned one visit on a random day to each customer without demand.

In order to investigate the influence of the maximum arrival time difference constraint on the results, we tested four different L values for each instance. To define reasonable L values, we ran the algorithm on each instance without bounding the arrival time difference. The vector of maximum arrival time differences obtained during these runs is denoted by L_1. The vectors with the reduced maximum arrival time differences, i.e., $L_{0.8}$, $L_{0.6}$, and $L_{0.4}$, are calculated by multiplying L_1 by 0.8, 0.6, and 0.4, respectively. For easier handling, we rounded all values to integers. The set of modified instances is referred to as data set $B = \{B_{0.5}, B_{0.7}, B_{0.9}\}$; B contains data sets with 50%, 70%, and 90% service frequencies. Table 3.2 presents the number of customers, n, that have to be served in each instance. The number of frequent customers, $|N^f|$, i.e., the number of customers in the template is given for each instance with the corresponding service frequencies, SF. The last row shows the average numbers of customers. Table 3.3 reports the applied L values.

The real-world data set used by Groër et al. [75] is not available. So, we adapted Gehring and Homberger's [71] benchmark instances for the vehicle routing problem with time windows (VRPTW) to test the algorithm's behavior on large problem instances. We refer to the large data set as data set B_{large}. The instances are divided into six different classes. Each class is characterized by a different spatial distribution of the customer locations (clustered (C1, C2), random (R1, R2) and random clustered (RC1, RC2)) and by different scheduling horizons (short scheduling horizon (R1, C1, RC1), long scheduling horizon (C2, R2, RC2)). We used one instance of each class with 1000 customers and extended them to a planning horizon of 25 days (i.e., five consecutive weeks with five days each). In order to turn the single period instances into periodic ones, we chose a service frequency of 50%. The demands and service times were chosen randomly according to a Poisson distribution with the mean

Instances	n	$\lvert N^f \rvert$ $SF = 50\%$	$SF = 70\%$	$SF = 90\%$
Christofides_1_5	50	39	48	50
Christofides_2_5	75	59	74	75
Christofides_3_5	100	79	95	100
Christofides_4_5	150	116	149	150
Christofides_5_5	199	153	195	199
Christofides_6_5	50	40	48	50
Christofides_7_5	75	59	75	75
Christofides_8_5	100	79	98	100
Christofides_9_5	150	116	147	150
Christofides_10_5	199	153	193	199
Christofides_11_5	120	91	116	120
Christofides_12_5	100	79	97	100
Average	114	88.58	111.25	114

Table 3.2 Total number of customers and number of frequent customers in instances with different service frequencies SF.

equal to the demands and service times in the original VRPTW instances; time windows were omitted.

3.5.2 Parameter tuning

The TALNS is controlled by several parameters. The initial setting was chosen according to the values suggested by Ropke and Pisinger [145]. We then used the benchmark data set of Groër et al. [75] and their L values (i.e., data set A) to fine tune the parameters. A sequential approach was used to tune one parameter after another. The parameter value that resulted in the best average objective value over five runs was chosen.

We tuned parameter $w_{\hat{t}}$ that controls the starting temperature of the simulated annealing, cooling rate c, and the penalty factor for infeasible solutions, $p_{penalty}$. The parameters that were tuned for the adaptive selection of operators are the reaction factor p_{react} and the scores for performance σ_1, σ_2, and σ_3. In the repair phase, we adjusted the noise parameter η. In the destroy phase, the randomization parameters p_{worst} and $p_{related}$, and the weights for computing the relatedness measurement, i.e., λ, μ, and ν, were set. The parameters were tested in the same order as listed in Table 3.4. The sequential tuning of all parameters was repeated three times. Table 3.4 provides the final values.

The parameter tuning led to an improvement of 0.1% on average. Therefore, we conclude that the algorithm behaves robustly with respect to the chosen parameters.

	$SF = 50\%$			$SF = 70\%$			$SF = 90\%$		
Instances	$L_{0.8}$	$L_{0.6}$	$L_{0.4}$	$L_{0.8}$	$L_{0.6}$	$L_{0.4}$	$L_{0.8}$	$L_{0.6}$	$L_{0.4}$
Christofides_1_5	54	41	27	34	26	17	32	24	16
Christofides_2_5	41	31	20	35	26	17	20	15	10
Christofides_3_5	46	34	23	30	22	15	23	17	11
Christofides_4_5	37	28	18	22	17	11	18	14	9
Christofides_5_5	31	23	15	28	21	14	10	7	5
Christofides_6_5	70	53	35	64	48	32	52	39	26
Christofides_7_5	61	46	30	67	50	33	56	42	28
Christofides_8_5	83	62	41	59	44	29	49	36	24
Christofides_9_5	78	59	39	71	53	35	43	32	21
Christofides_10_5	58	43	29	61	46	30	34	25	17
Christofides_11_5	38	29	19	15	11	7	16	12	8
Christofides_12_5	22	17	11	17	13	8	10	7	5

Table 3.3 Maximum arrival time differences for instances with different service frequencies SF.

$w_{\hat{t}}$	c	p_{react}	σ_1	σ_2	σ_3	$p_{penalty}$	η	p_{worst}	$p_{related}$	λ	μ	ν
0.01	0.9999	0.18	24	3	4	2	0.025	2	4	6	7	12

Table 3.4 Parameter setting TALNS.

3.5.3 Results for benchmark instances (data set *A*)

In a first step, we examine the impact of the number of TALNS iterations on the solution quality. Table 3.5 shows the total travel times plus the aggregated service times[2], TT, of the best solutions found during the development of the algorithm. In the first column, we indicate the instance names and the corresponding number of customers that have to be served. The remaining columns give the average gap to the best results and the computation time, *CPU*, in seconds for the solutions found after 10, 30, 50, and 70 thousand iterations, respectively.

Clearly, the solution quality improves with longer computation times. To be competitive in terms of solution quality and running time, we chose 30000 iterations for our experiments. This number of iterations is sufficient to obtain stable results over several runs. The resulting average variation coefficient (standard deviation / mean) is 0.0097 for a sample of ten runs.

An analysis of the destroy and repair operators is presented in Tables 3.6 and 3.7, respectively. The tables list the application frequency (number of calls per TALNS iterations), *AF*, for each operator and for each instance. The average computation time, CPU, for a single

[2]We add the aggregated service times to the objective value to be consistent with the literature.

Instances (# Customers)	Best found TT	10000 iterations Gap(%)	10000 iterations $CPU(s)$	30000 iterations Gap(%)	30000 iterations $CPU(s)$	50000 iterations Gap(%)	50000 iterations $CPU(s)$	70000 iterations Gap(%)	70000 iterations $CPU(s)$
Christofides_1_5_0.7 (50)	2124.21	6.09	1.80	3.33	5.45	1.80	9.25	1.45	12.79
Christofides_2_5_0.7 (75)	3530.01	3.00	4.92	2.13	14.69	1.75	24.35	1.56	33.74
Christofides_3_5_0.7 (100)	3285.55	2.79	8.74	1.60	25.58	1.44	43.46	1.39	61.22
Christofides_4_5_0.7 (150)	4484.89	4.27	29.13	2.54	84.31	2.10	139.14	2.05	189.63
Christofides_5_5_0.7 (199)	5556.13	4.31	44.84	2.33	122.24	2.05	199.47	1.93	269.09
Christofides_6_5_0.7 (50)	4051.48	0.01	2.19	0.00	6.63	0.00	11.11	0.00	15.41
Christofides_7_5_0.7 (75)	6653.48	3.51	6.20	2.29	18.33	2.11	30.52	1.93	43.21
Christofides_8_5_0.7 (100)	7096.88	1.96	10.65	1.35	32.24	1.29	53.41	1.22	75.53
Christofides_9_5_0.7 (150)	10331.80	3.06	34.32	1.14	97.39	0.96	153.42	0.76	206.32
Christofides_10_5_0.7 (199)	12973.60	3.76	54.81	2.04	146.32	1.67	232.13	1.55	316.02
Christofides_11_5_0.7 (120)	4459.06	2.33	13.15	1.07	35.96	0.69	56.96	0.68	76.73
Christofides_12_5_0.7 (100)	3487.50	2.24	8.92	1.22	25.60	1.08	42.72	0.88	58.45
Average	5669.55	3.11	18.31	1.75	51.23	1.41	82.99	1.28	113.18

Table 3.5 Impact of TALNS iterations on solution quality, TT, and computation time, CPU.

call is given in milliseconds. The average application frequency and the average computation time over all instances are shown in the last row of the tables.

With regard to the destroy operators, the random removal operator is chosen in the majority of the cases with 41% on average. This result indicates the need for diversification in the ConVRP. The regret operators with $q = 2$ and $q = 3$ are together used in 57% of the iterations to repair the destroyed templates. The application frequency of the least often applied destroy and repair operator is 10% and 11%, respectively. These values indicate that all operators contribute to the actual performance of the algorithm. The repair operators are randomized in 49% of the cases (Section 3.3.4).

There are significant differences in the computation times of the operators that can lead to unpredictable total running times. The TALNS adapts the application frequency of the operators on the basis of the test instances to be solved. Yet, it is not possible to estimate the share of each operator in advance. As a result, instances with similar parameters (e.g., number of customers) may require different computation times. Stability in terms of running time can be achieved by restricting the selection of complicated operator pairs, e.g., by normalizing the score ψ_{dr} in equation (3.28) (Pisinger and Ropke [135, 136]).

Stable running times are secondary in our applications. Therefore, we allow the adaptive mechanism to choose freely without any bias.

In a next step, we compare the results of different TALNS configurations: the TALNS as described in Section 3.3, the TALNS without using the truncated 2-opt operator from Section 3.3.4, and the TALNS with the exact repair mechanism (TALNS+ER) and the heuristic repair mechanism (TALNS+HR).

We also examine a variant of the TALNS that allows a partial violation of the driver

Instances	Random removal		Worst removal		Related removal		Cluster removal	
	AF	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)
Christofides_1_5_0.7	0.37	0.002	0.23	0.066	0.30	0.052	0.10	0.035
Christofides_2_5_0.7	0.48	0.003	0.12	0.153	0.30	0.128	0.11	0.037
Christofides_3_5_0.7	0.37	0.004	0.25	0.296	0.31	0.230	0.07	0.085
Christofides_4_5_0.7	0.49	0.006	0.18	0.751	0.28	0.584	0.04	0.098
Christofides_5_5_0.7	0.52	0.008	0.21	1.151	0.23	0.872	0.04	0.126
Christofides_6_5_0.7	0.30	0.003	0.25	0.064	0.27	0.045	0.17	0.029
Christofides_7_5_0.7	0.47	0.003	0.15	0.163	0.17	0.130	0.22	0.030
Christofides_8_5_0.7	0.38	0.004	0.24	0.307	0.31	0.237	0.08	0.074
Christofides_9_5_0.7	0.41	0.005	0.12	0.714	0.33	0.545	0.14	0.078
Christofides_10_5_0.7	0.41	0.007	0.09	1.119	0.36	0.836	0.15	0.110
Christofides_11_5_0.7	0.36	0.005	0.26	0.450	0.34	0.341	0.03	0.139
Christofides_12_5_0.7	0.42	0.004	0.17	0.281	0.34	0.233	0.07	0.058
Average	0.41	0.004	0.19	0.460	0.30	0.353	0.10	0.075

Table 3.6 Analysis of destroy operators (*AF* denotes the application frequency of the respective operator).

Instances	Greedy heuristic		Regret q=2		Regret q=3		Regret q=4		Regret q=m	
	AF	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)
Christofides_1_5_0.7	0.21	0.056	0.22	0.089	0.19	0.084	0.19	0.091	0.20	0.095
Christofides_2_5_0.7	0.10	0.176	0.36	0.366	0.28	0.357	0.20	0.357	0.06	0.364
Christofides_3_5_0.7	0.13	0.370	0.27	0.578	0.23	0.569	0.23	0.574	0.14	0.609
Christofides_4_5_0.7	0.12	1.503	0.30	2.358	0.23	2.402	0.24	2.439	0.11	2.514
Christofides_5_5_0.7	0.04	2.495	0.41	3.437	0.31	3.538	0.20	3.626	0.04	3.889
Christofides_6_5_0.7	0.19	0.071	0.23	0.111	0.20	0.116	0.19	0.114	0.19	0.118
Christofides_7_5_0.7	0.15	0.297	0.31	0.526	0.24	0.526	0.22	0.544	0.09	0.582
Christofides_8_5_0.7	0.16	0.536	0.26	0.827	0.22	0.836	0.20	0.834	0.16	0.827
Christofides_9_5_0.7	0.07	1.743	0.39	2.662	0.29	2.938	0.20	2.909	0.05	2.946
Christofides_10_5_0.7	0.10	2.655	0.48	4.263	0.20	4.526	0.17	4.554	0.05	4.886
Christofides_11_5_0.7	0.05	0.550	0.32	0.808	0.30	0.824	0.23	0.855	0.10	0.854
Christofides_12_5_0.7	0.12	0.381	0.24	0.627	0.23	0.612	0.26	0.623	0.15	0.650
Average	0.12	0.903	0.32	1.388	0.25	1.444	0.21	1.460	0.11	1.528

Table 3.7 Analysis of repair operators (*AF*, denotes the application frequency of the respective operator).

consistency. More precisely, we perform a post processing step on the final solution found by the TALNS (TALNS+PP). In this step, a small subset of the customers may be assigned to a second driver to avoid extremely expensive insertion positions. We sort all customers in decreasing order of the saving that could be obtained by temporarily removing each of them on the day he causes the highest insertion cost. The customers are then reassigned, one after another, to their cheapest position in a different driver's route. The insertion is performed by the greedy heuristic described in Section 3.3.2. The procedure stops when 5% of the customers are served by two drivers or if the solution cannot be improved by further reassignments.

In Table 3.8, we report the total travel times plus the aggregated service times, TT, and the computation times, *CPU*, for the mentioned TALNS configurations on the 12 bench-

	TALNS wo. 2-opt		TALNS		TALNS+ER		TALNS+HR		TALNS+PP	
Instances	TT	$CPU(s)$	TT	$CPU(s)$	TT	$CPU(s)$	TT	$CPU(s)$	TT	$CPU(s)$
Christofides_1_5_0.7	2136.20	4.93	2194.93	5.45	2130.99	40.50	2128.76	5.50	2186.05	5.45
Christofides_2_5_0.7	3605.04	14.65	3605.03	14.69	3589.88	30.04	3584.70	15.12	3580.13	14.69
Christofides_3_5_0.7	3361.24	24.26	3338.03	25.58	3324.94	81.86	3330.32	26.23	3332.53	25.58
Christofides_4_5_0.7	4617.69	80.97	4598.78	84.31	4600.26	132.90	4575.78	86.86	4584.61	84.31
Christofides_5_5_0.7	5695.15	125.54	5685.55	122.24	5706.27	161.86	5739.10	132.15	5662.88	122.24
Christofides_6_5_0.7	4064.34	5.79	4051.50	6.63	4051.48	20.57	4051.48	6.65	4050.86	6.63
Christofides_7_5_0.7	6828.52	18.61	6805.99	18.33	6779.03	26.53	6779.03	19.74	6756.08	18.33
Christofides_8_5_0.7	7234.34	30.28	7192.45	32.24	7199.34	49.38	7207.18	32.63	7173.27	32.24
Christofides_9_5_0.7	10524.83	94.83	10450.04	97.39	10488.87	110.40	10488.87	97.97	10411.60	97.39
Christofides_10_5_0.7	13273.72	148.74	13238.57	146.32	13193.79	207.79	13185.10	144.05	13200.49	146.32
Christofides_11_5_0.7	4531.70	34.65	4506.59	35.96	4481.99	130.18	4481.91	37.66	4499.16	35.96
Christofides_12_5_0.7	3545.98	23.65	3530.16	25.60	3543.34	63.50	3532.73	25.87	3519.06	25.60
Average	5784.90	50.57	5766.47	51.23	5757.52	87.96	5757.08	52.53	5746.39	51.23

Table 3.8 Comparison of TALNS variants.

mark instances from data set *A*. The table shows the benefit of the truncated 2-opt operator that comes with a small increase in the average computation time. As expected, the variants that integrate a method to repair infeasible solutions perform better than the pure TALNS. Nevertheless, there is little difference between the average results of the TALNS and the TALNS+ER. This is because the loose bounds on the maximum arrival time difference in the instances allow good results even without adjusting the departure times. Noticeable is the comparison between TALNS+ER and TALNS+HR since for some instances the heuristic repair achieves better results than the exact repair approach. There are several reasons for this observation. First, the repair approaches play only a minor role due to the loose bounds on the maximum arrival time difference. Second, the randomization of the TALNS causes the repair methods to face different solutions with different opportunities to adjust the departure times. Furthermore, the repair methods generate solutions with different maximum arrival time differences, l_{max}; since l_{max} is considered in the acceptance criterion (equation (3.26)), it also effects the score ψ_{dr} that can be earned by the destroy and repair operators.

The post processing phase (TALNS+PP) improves the TALNS solutions by 0.34% on average. This small improvement indicates that the TALNS avoids extremely inconvenient insertion positions despite the strict driver consistency requirement.

To test the performance of the developed algorithm, we compare it to the template-based record-to-record travel algorithm for the ConVRP (ConRTR) (Groër et al. [75]) and to the tabu search algorithm (TTS) (Tarantilis et al. [170]) that is also template-based. The results of the ConRTR and the TTS are taken from the respective papers.

The performance of the three algorithms for the benchmark instances is listed in Table 3.9. The first column gives the names of the test instances. In the following columns, TT and the obtained maximum arrival time difference, l_{max}, are listed for each algorithm.

The TTS is stochastic and the reported results are the best of five runs (TT_{min}). The ConRTR algorithm is deterministic. For the TALNS, we report the average results over ten runs (TT_{avg}) and the best of five randomly chosen runs (TT_{min}). The computation time of the best of five runs is given in seconds for TTS (CPU_{min}) and the average computation time is given for the TALNS (CPU_{avg}). The computation times of the ConRTR are not available. The last two columns show the improvement (Imp) of the TALNS over the ConRTR (with respect to TT_{avg}) and over the TTS (with respect to TT_{min}). In the last row, the average results over all instances are given.

The results reported for the ConRTR approach were obtained without bounding the maximum arrival time differences. The L values for the TTS and the TALNS are set to the l_{max} values produced by the ConRTR method. The authors of the ConRTR pointed out that their goal is to develop a simple algorithm that relies on the template concept. The structure of the TALNS is more complex; yet, it generates solutions that are on average 5.84% better than those of the ConRTR. Furthermore, all results obtained by the ConRTR are improved. The difference between the results is smaller when comparing the TALNS to the TTS approach; the TALNS performs on average 1.89% better than the TTS. Additionally, the TTS requires an average computation time of 408 seconds while the TALNS requires only 51 seconds on a similar processor. The master and daily scheduler heuristic for the courier delivery problem (Sungur et al. [167]) has also been applied to the ConVRP benchmark instances. The authors do not bound the maximum arrival time difference; so, only instances in which TALNS obtains smaller or equal l_{max} values are compared. This is the case in 9 out of 12 instances. Here, TALNS improves the total travel time by 6.5% on average and it decreases l_{max} by 28.7%.

These comparisons indicate that the TALNS is a competitive solution method for the ConVRP.

3.5.4 Results for new instances (data set *B*)

In this section, we examine the effect of decreasing maximum arrival time differences on the results obtained by the pure TALNS and the TALNS with the integrated repair mechanisms. Experiments are performed on the new instances, i.e., data set *B*.

Tables 3.10, 3.11, and 3.12 contain the average results of the TALNS variants over all modified benchmark instances with service frequencies 50%, 70%, and 90%, respectively. The first column gives the L vectors (Section 3.5.1) that are applied on the corresponding instances. In the following columns, the average results achieved by the TALNS with the

	ConRTR		TTS			TALNS					
Instances	TT	l_{max}	TT_{min}	l_{max}	CPU_{min}	TT_{avg}	TT_{min}	l_{max}	CPU_{avg}	Imp (%) ConRTR	Imp (%) TTS
Christofides_1_5_0.7	2282.14	24.38	2210.56	21.99	80.00	2194.93	2124.21	23.72	5.45	3.82	3.91
Christofides_2_5_0.7	3872.86	34.26	3622.71	27.75	93.00	3605.03	3600.41	31.86	14.69	6.92	0.62
Christofides_3_5_0.7	3628.22	22.87	3451.10	21.92	369.00	3338.03	3326.12	22.21	25.58	8.00	3.62
Christofides_4_5_0.7	4952.91	27.53	4572.00	25.15	388.00	4598.78	4556.33	24.19	84.31	7.15	0.34
Christofides_5_5_0.7	6416.77	26.93	5732.62	19.99	550.00	5685.55	5664.06	22.69	122.24	11.40	1.20
Christofides_6_5_0.7	4084.24	63.47	4096.87	55.38	70.00	4051.50	4051.48	63.26	6.63	0.80	-.11
Christofides_7_5_0.7	7126.07	83.96	6752.36	63.28	161.00	6805.99	6770.49	76.62	18.33	4.49	-0.27
Christofides_8_5_0.7	7456.19	73.04	7279.39	62.01	539.00	7192.45	7129.79	65.97	32.24	3.54	2.06
Christofides_9_5_0.7	11033.54	106.43	10585.10	84.76	947.00	10450.04	10381.9	88.85	97.39	5.29	1.92
Christofides_10_5_0.7	13916.80	60.17	13120.40	57.17	1052.00	13238.57	13102.7	57.95	146.32	4.87	0.13
Christofides_11_5_0.7	4753.89	16.10	4721.09	15.68	480.00	4506.59	4485.37	15.33	35.96	5.20	4.99
Christofides_12_5_0.7	3861.35	17.58	3607.88	16.91	172.00	3530.16	3497.93	16.50	25.60	8.58	3.05
Average	6115.42	46.39	5812.67	39.33	408.42	5766.47	5724.23	42.43	51.23	5.84	1.89

Table 3.9 Comparison to existing approaches on data set *A*.

exact repair mechanism (TALNS+ER), with the heuristic repair mechanism (TALNS+HR), and without any repair mechanism are presented with the corresponding computation times, *CPU*. The gaps to the results obtained by the TALNS+ER are given for the TALNS+HR and the pure TALNS. In the last but one row, we show the average results over all L vectors. The last row gives the gaps between the results obtained with constraints L_1 and $L_{0.4}$ for the three solution approaches. A comparison between the different solution approaches reveals large differences in the results and highlights interesting saving potentials for companies that are facing similar problems.

The pure TALNS provides good solutions for wide L constraints. However, with tighter constraints it is more difficult to obtain solutions with short travel times. In the worst case, a 60% decrease in L leads to a 186.16% increase in the total travel time (see Table 3.11). The substantial increase in the objective values, as well as in the computation times, is due to the large number of vehicles needed to cope with the tight arrival time difference constraints.

With flexible departure times, we can optimize the total travel time almost independently from the maximum allowed arrival time difference. With small differences, this is true for the exact and the heuristic adjustment of the departure times. The total travel time increases on average between 0.63% and 1.62% with the TALNS+ER method and between 1.55% and 5.46% with the TALNS+HR method, while the L values decrease by 60%. As explained above, the results obtained by the TALNS+HR method might be better than those of TALNS+ER due to the randomization of the TALNS and the effect of the maximum arrival time difference on the scores that can be earned by the repair and destroy operators.

The conflict between the total travel time and the maximum arrival time difference decreases with larger service frequency. Test instances with high service frequency have a

	TALNS+ER	$CPU(s)$	TALNS+HR	$CPU(s)$	Gap (%) to TALNS+ER	TALNS	$CPU(s)$	Gap (%) to TALNS+ER
L_1	4192.47	34.01	4192.47	34.01	0.00	4192.47	34.01	0.00
$L_{0.8}$	4204.81	115.21	4207.88	35.73	0.07	4215.56	35.15	0.26
$L_{0.6}$	4222.67	132.13	4218.23	34.97	-0.11	5176.83	47.52	22.60
$L_{0.4}$	4260.50	134.23	4421.42	39.80	3.78	10085.78	92.92	136.73
Average	4220.11	103.90	4260.00	36.13	0.94	5917.66	52.40	39.89
Gap (%) $L_1 - L_{0.4}$	1.62		5.46			140.57		

Table 3.10 Comparison of TALNS variants on data set $B_{0.5}$.

	TALNS+ER	$CPU(s)$	TALNS+HR	$CPU(s)$	Gap (%) to TALNS+ER	TALNS	$CPU(s)$	Gap (%) to TALNS+ER
L_1	5752.85	52.01	5752.85	52.01	0.00	5752.85	52.01	0.00
$L_{0.8}$	5760.89	121.58	5756.37	53.28	-0.08	5770.88	50.70	0.17
$L_{0.6}$	5773.59	149.36	5763.57	52.70	-0.17	6266.31	61.51	8.53
$L_{0.4}$	5811.67	147.40	5894.96	52.35	1.43	16462.33	133.28	183.26
Average	5774.75	117.59	5791.94	52.59	0.30	8563.09	74.38	47.99
Gap (%) $L_1 - L_{0.4}$	1.02		2.47			186.16		

Table 3.11 Comparison of TALNS variants on data set $B_{0.7}$.

larger number of frequent customers. Therefore, the corresponding template contains more customers and is not altered much during the resolution. A high service frequency results in templates that give an accurate reflection of the daily schedules and produce better results.

To obtain good solutions for instances with low service frequencies, interdependencies during the insertion of non-frequent customers must be considered. One can think of interdependencies when the insertion of a customer causes a violation of the arrival time consistency, but the next insertion on another day makes the solution feasible. The TALNS considers this issue indirectly by randomizing the insertion through the noise term and by applying the truncated 2-opt operator (Section 3.3.4). The tabu search algorithm by Tarantilis et al. [170] integrates a post optimization phase that moves non-frequent customers to different insertion positions. Unfortunately, it is not possible to compare the two approaches on the basis of the existing benchmark instances: the service frequency of 70% results in instances with only 3% non-frequent customers on average. Problem instances with a larger number of non-frequent customers would be more appropriate for a meaningful comparison.

3.5.5 Results for large instances (data set B_{large})

The TALNS performs well on problems with a short planning horizon. However, our approach is inappropriate in a long-term environment with fluctuating demand, e.g., when we have to generate routing plans for consecutive weeks. Solving each week separately

	TALNS+ER	$CPU(s)$	TALNS+HR	$CPU(s)$	Gap (%) to TALNS+ER	TALNS	$CPU(s)$	Gap (%) to TALNS+ER
L_1	6729.94	55.75	6729.94	55.75	0.00	6729.94	55.75	0.00
$L_{0.8}$	6730.55	120.92	6735.30	60.00	0.07	6757.17	54.70	0.40
$L_{0.6}$	6734.61	170.40	6743.22	56.42	0.13	7047.31	62.80	4.64
$L_{0.4}$	6772.21	180.96	6833.95	58.19	0.91	15113.41	149.77	123.17
Average	6741.83	132.01	6760.60	57.59	0.28	8911.96	80.76	32.05
Gap (%) $L_1 - L_{0.4}$	0.63		1.55			124.57		

Table 3.12 Comparison of TALNS variants on data set $B_{0.9}$.

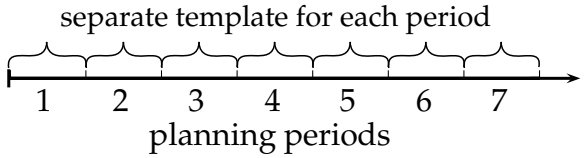


Figure 3.7 Solving each period separately results in inconsistent long-term solutions.

would disregard both, long-term time and driver consistency. Figure 3.7 shows an example in which customers are serviced over several planning periods. The demand is fluctuating and new customers place orders over time. So, the solution algorithm is rerun each period (1-7). The changing input data leads to different templates and, therefore, to inconsistent solutions.

We follow Groër et al. [75] to cope with this issue: First, we design a template that is based on historical data; Then, we apply the same template to derive the solutions for several future periods. This approach is illustrated in Figure 3.8. A template is generated by using data from periods 1 to 4. The template is then used to generate solutions in periods 5, 6, and 7.

By using the same template for several periods it is guaranteed that all customers that are assigned to the template receive consistent service. However, the service consistency of new customers that are not assigned to any template route is ignored. The same is true for customers that have to be removed from the template routes in order to make them feasible, i.e., customers that cannot be visited without violating the capacity and tour length constraints. Frequent customers that are not considered in the template are inserted into their best insertion positions only with regard to the total travel time.

In this section, we investigate the behavior of the algorithm when no reasonable bound on the maximum arrival time difference can be predicted in advance. Typically, this is the case in long-term planning environments with fluctuating demand. We perform experiments on the large instances (i.e., data set B_{large}) without bounding l_{max} and examine the consis-

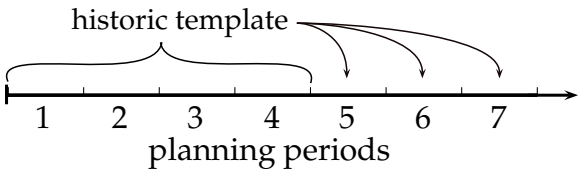


Figure 3.8 Consistency over the long term by using a historic template.

tency of the resulting solutions. Additionally, we compare the solutions obtained by using the historic template with those obtained by using a template based on current data.

Tables 3.13 and 3.14 show the average results of five runs for week five of the large instances. Like Groër et al. [75], we rely on the template's precedence principle to obtain time consistent routing plans when the maximum arrival time difference is not bounded explicitly.

The results in Table 3.13 are obtained without using the 2-opt operator and those in Table 3.14 with using it. The first columns of both tables indicate the instance names and the corresponding maximum shift length, T; T can be interpreted as a trivial upper bound for the maximum arrival time difference. In columns 2 and 3, we give the total travel time plus service times, TT, and the number of vehicles, NV. Columns 4-7 show the average arrival time difference, l_{avg}, the maximum arrival time difference, l_{max}, the ratio between l_{max} and T, and the computation time in minutes, CPU, when vehicle departure times are fixed to zero (TALNS). Columns 8-11 show the same items but with flexible departure times (TALNS+ER). We have no bound on the maximum arrival time difference; so, we apply the exact repair approach described in Section 3.4.1 only to the final solution. Accordingly, the adjustment of departure times has a minor effect on the total computation time.

The results highlight the trade-off between total travel time and the maximum arrival time difference. Applying the truncated 2-opt operator decreases the total travel time at the expense of larger arrival time differences: TT decreases by 0.75% on average while the average ratio between l_{max} and T increases from 19% to 23%. A preference to use either of the two approaches depends on the preference of the decision makers; so, no evaluation can be made here. Shifting the departure times of the vehicles almost halves the maximum arrival time difference and should be considered whenever possible.

To investigate the ability of the TALNS to produce templates for future planning periods, we solve the ConVRP for week 1 - 4 (planning horizon $|D|$ = 20 days) and apply the obtained template to solve week five. Our aim is to provide long-term consistency for a large number of frequent customers. Therefore, each customer with an average of at least

two visits per week is assigned to one template route. On each day of week five, we resolve the template by removing customers that do not require service. If the capacity or tour length constraints are exceeded despite the removals, we remove further customers from the routes; customers are removed in ascending order of service frequency until all routes become feasible. Finally, we insert new customers and customers that are not assigned to any template route but require service by using the greedy heuristic (Section 3.3.2); service consistency is ignore in this phase.

Columns 2 and 3 of Table 3.15 show the number of customers in the template that is generated from data of week 1-4 and the computation time in minutes, CPU. The remaining columns show the number of frequent customers $|N^f|$ in week five and the number of frequent customers that are not represented in the template.

Tables 3.16 and 3.17 give the results of week five produced from the historic template. The results in Table 3.16 are obtained by resolving the template as described above. The results in Table 3.17 are obtained by additionally performing a post-optimization with the truncated 2-opt operator. The structure of the tables is similar to Tables 3.13 and 3.14. The only exceptions are that the computation times refer to the time for resolving the template in seconds and that the number of customers that have been removed in order to make routes feasible is indicated.

When comparing the solutions from Tables 3.16 and 3.17 to those in Tables 3.13 and 3.14 one must pay attention to two antagonistic factors. On the one hand, the solutions in Tables 3.16 and 3.17 are derived from a template based exclusively on historical data. This is suboptimal in terms of total travel time even though it enables long-term consistency. On the other hand, there is a higher degree of freedom because almost 70 customers (on average) are inserted by ignoring service consistency. These customers are placed into their best positions with regard to travel time. Therefore, a low total travel time is associated with low consistency at least for some customers.

If a decision maker is willing to make this compromise, TALNS is eligible. It provides historic templates that can be applied to future planning periods with approximately 1.3% increase in the total travel times compared to solutions produced from specially built templates. With respect to time consistency both templates, historic and current, lead to comparable average arrival time differences. The repair mechanism that adjusts the departure times almost halves the maximum arrival time differences. However, in some cases, minimizing l_{max} increases the average maximum arrival time difference (see, e.g., instances R2 and RC2 in Table 3.17). Furthermore, driver consistency is ignored for approximately 7% of the customers.

			TALNS				TALNS+ER			
Instances (T)	TT	NV	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$
R1_10_1_5 (1925)	325789.00	66.80	42.89	432.74	0.22	25.22	42.68	205.18	0.11	25.22
R2_10_1_5 (7697)	95627.32	10.40	68.54	331.22	0.04	36.26	55.76	168.97	0.02	36.26
C1_10_1_5 (1824)	551133.60	90.20	128.44	674.08	0.37	28.82	117.82	327.01	0.18	28.82
C2_10_1_5 (3914)	319889.60	18.20	232.81	1166.82	0.30	36.46	205.17	639.34	0.16	36.46
RC1_10_1_5 (1821)	322549.40	66.20	43.01	343.06	0.19	27.72	41.78	163.98	0.09	27.72
RC2_10_1_5 (7284)	94075.44	10.40	58.70	306.72	0.04	31.87	52.64	157.62	0.02	31.87
Average	284844.06	43.70	95.73	542.44	0.19	31.06	85.97	277.02	0.10	31.06

Table 3.13 Results for week five of the large instances without 2-opt.

			TALNS				TALNS+ER			
Instances (T)	TT	NV	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$
R1_10_1_5 (1925)	323128.60	68.20	49.93	473.47	0.25	30.08	51.18	281.09	0.15	30.08
R2_10_1_5 (7697)	95599.68	10.20	65.70	320.73	0.04	32.62	57.18	170.76	0.02	32.62
C1_10_1_5 (1824)	544885.20	88.60	138.49	810.66	0.44	34.67	149.39	521.22	0.29	34.68
C2_10_1_5 (3914)	319547.80	18.00	240.59	1290.64	0.33	36.09	217.57	673.77	0.17	36.09
RC1_10_1_5 (1821)	319051.80	65.00	49.04	509.51	0.28	28.08	50.16	294.19	0.16	28.08
RC2_10_1_5 (7284)	94053.22	10.20	53.46	259.77	0.04	31.67	47.00	133.65	0.02	31.67
Average	282711.05	43.37	99.54	610.80	0.23	32.20	95.41	345.78	0.13	32.20

Table 3.14 Results for week five of the large instances with 2-opt.

3.6 Summary

In this chapter, we first presented a solution approach called template-based adaptive large neighborhood search (TALNS) for the ConVRP. It embeds the principle of deriving a multi-day routing plan from a set of template routes into the adaptive large neighborhood search framework. Comparisons with other algorithms indicate the competitiveness of the TALNS in terms of solution quality and computation time. The algorithm provides slightly better results than the best competitor while being about eight times faster.

We constructed new test instances by varying the service frequencies of customers and by setting different bounds on the maximum arrival time difference. The requirements on

	template (week 1-4)		week 5	
Instances	# customers	CPU(min)	$\|N^f\|$	$\|N^f\|$ not in template
R1_10_1	864	73.85	821	52
R2_10_1	876	142.71	810	58
C1_10_1	850	160.96	820	72
C2_10_1	875	123.92	806	65
RC1_10_1	860	69.39	819	75
RC2_10_1	856	182.35	816	74
Average	863.50	125.53	815.33	66.00

Table 3.15 Properties of historic templates and week five of the large instances.

				TALNS				TALNS+ER			
Instances (T)	TT	NV	removed	l_{avg}	l_{max}	l_{max}/T	CPU(s)	l_{avg}	l_{max}	l_{max}/T	CPU(s)
R1_10_1_5	344930.80	75.80	7.20	39.93	368.61	0.19	0.00	38.63	179.71	0.09	0.05
R2_10_1_5	99239.68	11.00	1.00	75.74	1684.64	0.22	0.10	226.76	924.34	0.12	0.15
C1_10_1_5	552262.20	105.00	4.20	102.83	707.77	0.39	0.00	94.85	297.05	0.16	0.05
C2_10_1_5	294020.40	19.60	1.20	223.38	1335.50	0.34	0.01	196.34	639.31	0.16	0.05
RC1_10_1_5	343862.00	75.40	5.80	34.16	369.22	0.20	0.00	31.77	153.73	0.08	0.06
RC2_10_1_5	91613.80	10.40	1.80	62.03	700.25	0.10	0.06	90.39	389.83	0.05	0.11
Average	287654.81	49.53	3.53	89.68	861.00	0.24	0.03	113.12	430.66	0.11	0.08

Table 3.16 Results for week five of the large instances produced from historic template without 2-opt.

				TALNS				TALNS+ER			
Instances (T)	TT	NV	removed	l_{avg}	l_{max}	l_{max}/T	$CPU(s)$	l_{avg}	l_{max}	l_{max}/T	$CPU(s)$
R1_10_1_5	342652.00	75.80	7.20	44.91	406.60	0.21	0.02	44.23	235.33	0.12	0.08
R2_10_1_5	98945.14	11.00	1.00	76.31	1685.97	0.22	0.11	243.57	928.88	0.12	0.17
C1_10_1_5	551463.80	105.00	4.20	114.15	798.11	0.44	0.01	116.49	433.91	0.24	0.06
C2_10_1_5	293263.80	19.60	1.20	227.95	1337.25	0.34	0.01	199.78	656.41	0.17	0.05
RC1_10_1_5	342081.60	75.40	5.80	38.09	445.65	0.24	0.01	37.88	261.42	0.14	0.06
RC2_10_1_5	91297.00	10.40	1.80	61.72	694.00	0.10	0.07	89.69	388.64	0.05	0.12
Average	286617.22	49.53	3.53	93.86	894.59	0.26	0.04	121.94	484.10	0.14	0.09

Table 3.17 Results for week five of the large instances produced from historic template with 2-opt.

arrival time consistency are very loose in the benchmark instances by Groër et al. [75]; also, the service frequencies are high, i.e., there are very few customers that require only one visit in the planning period (on average 3%). The new instances enabled us to examine interesting effects on the trade-off between service consistency and travel cost.

The numerical experiments showed that the TALNS generates very good solutions when arrival time consistency is loose. However, with tight maximum arrival time difference constraints the cost increases rapidly. We modified the original ConVRP model and allowed delayed departure times from the depot. This relaxation mitigates the conflict between total travel time and time consistency, i.e., arrival time consistency can be improved significantly with minor increase in routing cost when departure times are flexible.

Chapter 4

The generalized consistent vehicle routing problem

4.1 Introduction

In the previous chapter, we dealt with the consistent vehicle routing problem (ConVRP, Groër et al. [75]) in which customer satisfaction is maintained by providing driver and time consistent service. Driver consistency is expressed by the number of different drivers that visit a customer. Only one driver per customer is allowed in the ConVRP. Time consistency is achieved by bounding the maximum difference between the earliest and the latest arrival time at each customer. Vehicle idling to reduce the arrival time difference is not allowed. The objective is to find a minimum cost routing plan that complies with the classical routing constraints and the consistency requirements.

Since some assumptions in the ConVRP may be too restrictive in practice, we introduce the generalized consistent vehicle routing problem (GenConVRP). It extends and relaxes the ConVRP as follows:

- Each customer may be served by a limited number of drivers, instead of only one. This assumption is more realistic, since specific drivers are not always available (e.g., due to vacations or illnesses). In many cases, it is impossible to assign only one driver per customer over a long period of time. Typically, at least two drivers are regularly visiting a certain customer. Coelho et al. [32] propose a partial driver consistency feature where the number of different drivers per customer is minimized. They show that greater flexibility in the driver-to-customer assignment leads to slightly cheaper routing plans.

- The maximum arrival time difference, i.e., time consistency, is not enforced by constraints but is penalized in the objective function. The total number of drivers in the ConVRP is not restrictive, so it is always possible to generate a solution with a maximum arrival time difference of zero (e.g., if each driver visits only one customer). We include the maximum arrival time difference in the objective function because it is difficult to predict bounds that are sufficiently tight to guarantee high service quality without leading to a disproportional increase in travel cost.
- Customers are associated with AM/PM time windows. This constraint reflects the requirement of companies which hire part-time workers to handle package deliveries (Wong [185]).
- Vehicle departure time is flexible. This extension is necessary because of the AM/PM time windows in combination with constraints that prohibit waiting times. Additionally, time consistency can be improved by adjusting the vehicle departure times (see Chapter 3 and Kovacs et al. [107]).

Motivated by Ropke and Pisinger [145], Schrimpf et al. [155], and Shaw [156], we propose a large neighborhood search algorithm (LNS) for the GenConVRP. LNS is a local search technique that performs powerful moves in the search space by iteratively removing large parts of a solution and assembling a new and, hopefully, better solution.

In this chapter, we first introduce the GenConVRP, which is formally defined in Section 4.2. Second, we devise an LNS metaheuristic tailored to the GenConVRP (Section 4.3). It integrates a new greedy algorithm to minimize the maximum arrival time differences at the customers by adjusting the vehicle starting times. The LNS performs robustly for various versions and settings of the original ConVRP and dominates all previous (template-based) approaches, thus providing a new state of the art for this problem class. For the GenConVRP, we show that the gap with respect to the optimal solutions (on small instances that can be solved by CPLEX) is small. Finally, we examine the trade-off between travel cost and customer satisfaction on the basis of a variety of new and existing benchmark instances; the impact of a changing emphasis on time and driver consistency on variable and fixed costs (i.e., routing cost and fleet size) of service providers is analyzed in Section 4.4.

4.2 Problem definition

The GenConVRP is defined on a complete directed graph $G = (N \cup \{0\}, A)$, where $N = \{1, \ldots, n\}$ is the set of customers and 0 is the depot; $N^0 = N \cup \{0\}$. Set $N^{am} \subseteq N$ contains

customers that can only be visited in the morning and set $N^{pm} \subseteq N$ contains customers that require a visit in the afternoon ($N^{am} \cap N^{pm} = \{\}, N^{am} \cup N^{pm} = N$). $A = \{(i,j) : i,j \in N^0, i \neq j\}$ is the set of arcs. (Arcs from customers $\in N^{pm}$ to customers $\in N^{am}$ can be omitted from the graph.) Customers are visited by a homogeneous fleet of vehicles in the set K. The number of vehicles is not restrictive (i.e., $|K| = |N|$). Each vehicle $k \in K$ has a capacity of Q. Drivers and vehicles share a one-to-one relationship. We use drivers and vehicles interchangeably, depending on the context. Vehicles start and end their routes at the depot. The departure time is arbitrary, but vehicles must return to the depot before time T. The planning horizon involves $|D|$ days where D is the set of days. On each day $d \in D$, each customer $i \in N$ has demand q_{id} and service time s_{id}. We use auxiliary parameters w_{id} equal to one if customer i requires service on day d ($q_{id} > 0$) and equal to zero, otherwise. Each arc $(i,j) \in A$ is associated with travel time t_{ij}. The travel time matrix satisfies the triangle inequality. Driver consistency is enforced by assigning each customer $i \in N$ to at most W ($W \geq 1$) different drivers over the planning period. Parameter α is the weight of the total travel time and $(1-\alpha)$ gives the weight of the maximum arrival time difference (l_{max}) in the objective function. The model uses the following binary variables:

$$x_{ijkd} = \begin{cases} 1, \text{if arc } (i,j) \text{ is traversed by vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

$$y_{ikd} = \begin{cases} 1, \text{if customer } i \text{ is assigned to vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

$$z_{ik} = \begin{cases} 1, \text{if customer } i \text{ is assigned to vehicle } k, \\ 0, \text{otherwise.} \end{cases}$$

Continuous variables a_{id} denote the arrival time at customer $i \in N$ on day d.

$$\textit{Minimize} \quad \alpha \sum_{d \in D} \sum_{k \in K} \sum_{i \in N^0} \sum_{j \in N^0} t_{ij} x_{ijkd} + (1-\alpha) l_{max} \tag{4.1}$$

subject to:

$$y_{0kd} = 1 \qquad \forall\, k \in K, d \in D, \tag{4.2}$$

$$\sum_{k \in K} y_{ikd} = w_{id} \qquad \forall\, i \in N, d \in D, \tag{4.3}$$

$$\sum_{i \in N} q_{id} y_{ikd} \leq Q \qquad \forall\, k \in K, d \in D, \tag{4.4}$$

$$\sum_{i \in N^0} x_{ijkd} = \sum_{i \in N^0} x_{jikd} = y_{jkd} \qquad \forall\, j \in N^0, k \in K, d \in D, \tag{4.5}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \leq a_{jd} \qquad \forall\, i, j \in N, k \in K, d \in D, \tag{4.6}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \geq a_{jd} \qquad \forall\, i, j \in N, k \in K, d \in D, \tag{4.7}$$

$$z_{ik} \geq y_{ikd} \qquad \forall\, i \in N, k \in K, d \in D, \tag{4.8}$$

$$\sum_{k \in K} z_{ik} \leq W \qquad \forall\, i \in N, \tag{4.9}$$

$$(a_{i\alpha} - a_{i\beta}) w_{i\alpha} w_{i\beta} \leq l_{max} \qquad \forall\, i \in N, \alpha, \beta \in D, \tag{4.10}$$

$$a_{id} + s_{id} + t_{i0} \leq T \qquad \forall\, i \in N, d \in D, \tag{4.11}$$

$$a_{id} \leq \frac{T}{2} \qquad \forall\, i \in N^{am}, d \in D, \tag{4.12}$$

$$a_{id} \geq \frac{T}{2} \qquad \forall\, i \in N^{pm}, d \in D, \tag{4.13}$$

$$a_{id} \geq t_{0i} \qquad \forall\, i \in N, d \in D, \tag{4.14}$$

$$x_{ijkd} \in \{0, 1\} \qquad \forall\, i, j \in N^0, k \in K, d \in D, \tag{4.15}$$

$$y_{ikd} \in \{0, 1\} \qquad \forall\, i \in N^0, k \in K, d \in D, \tag{4.16}$$

$$z_{ik} \in \{0, 1\} \qquad \forall\, i \in N, k \in K. \tag{4.17}$$

The objective function (4.1) minimizes the weighted average of the total travel time and the maximum arrival time difference. Inequalities (4.2) ensure that each route starts from the depot. Constraints (4.3) guarantee that each customer is serviced on each day he requires service. Inequalities (4.4) limit the vehicle capacity to Q. Equalities (4.5) are flow conservation constraints. Inequalities (4.6) and (4.7) set the arrival times at the customers. Vehicle idling to improve time consistency is not allowed (4.7). Inequalities (4.6) also prevent sub-tours. Driver consistency is ensured in (4.8) and (4.9). Constraints (4.8) set the z variables and constraints (4.9) make sure that the number of drivers per customer does not exceed the allowed limit. Constraints (4.10) set the maximum arrival time difference. Inequalities (4.11) enforce that vehicles return to the depot in time. Time window feasibility is ensured in (4.12) and (4.13). Constraints (4.14) set the earliest possible arrival time at each customer. Finally, constraints (4.15)-(4.17) define the domains of the decision variables.

4.3 Solution framework

A major difficulty in solving the ConVRP and its variants is the interrelation between the days in the planning horizon. State-of-the-art metaheuristics (Groër et al. [75], Kovacs et al. [107], Tarantilis et al. [170], and Chapter 3) rely on the template concept to generate good solutions. A template is a set of routes from which the daily routing plans are derived. On each day, the template is resolved by deleting customers that appear in the template but do not require service and by inserting customers that are not represented in the template but require service. Solutions derived from a template comply with driver consistency since the driver-to-customer assignment is not changed during the resolution. Additionally, the customer sequence in the template routes is transferred to the daily routes which supports time consistency.

Our solution approach for the GenConVRP does not make use of template routes. The template concept restricts the search space and, therefore, good solutions are found quickly. Yet, better solutions might be missed. We propose a large neighborhood search (LNS) (Shaw [156]) that is more flexible than template-based approaches because it operates on the entire routing plan. The pseudocode of the LNS is presented in Algorithm 2.

Given an initial solution, LNS iteratively destroys the current incumbent solution by removing a large number of customers. It then generates a new solution by using a repair operator to reinsert the removed customers into the routes (line 4). Several destroy and repair operators with different algorithmic structures are available. In each iteration, one destroy operator and one repair operator are selected randomly (line 3).

The vehicle departure times are adjusted in order to eliminate waiting times and to reduce the maximum arrival time difference (line 5). Time consistency of feasible solutions is improved by iteratively reversing parts of the route that contains the customer with the maximum arrival time difference (line 7). New solutions are accepted as current incumbents according to a simulated annealing criterion (line 9).

In the following, we describe the modules of the LNS algorithm in detail. The LNS is similar to the template-based adaptive large neighborhood search presented in Chapter 3. Nevertheless, the modules have to be adapted in order to substitute for the template concept.

4.3.1 Destroy operators

We use several destroy operators to remove customers from the solution. Each of them adopts a different approach to improve the current solution. Customers are selected with regard to all days in the planning horizon and are removed from several days. Following

Algorithm 2 *LNS*

Require: initial solution s ▷ Section 4.3.8
1: $s^{best} = s$
2: **repeat**
3: select one destroy operator d and one repair operator r randomly
4: s' = generateNewSolution(s, d, r)
5: adjustDepartureTimes(s') ▷ Section 4.3.3
6: **if** isDriverFeasible(s') **then**
7: improveTimeConsistency(s') ▷ Section 4.3.5
8: **end if**
9: **if** accept(s',s) **then** ▷ Section 4.3.7
10: $s = s'$
11: **end if**
12: **if** $f(s') < f(s^{best})$ **then**
13: $s^{best} = s'$
14: **end if**
15: **until** maximum number of iterations is reached
16: **return** s^{best}

Pisinger and Ropke [135], the total number of removals, u, is chosen from the interval $[\min\{0.1\sum_{d\in D}\sum_{i\in N} w_{id}, 30\}, \min\{0.4\sum_{d\in D}\sum_{i\in N} w_{id}, 60\}]$. In large instances, the number of customers to remove is chosen from the interval [30,60]; Pisinger and Ropke [135] argue that removing less than 30 or more than 60 customers rarely leads to improving solutions.

We use five different destroy operators: Random removal, related removal, and cluster removal are based on Pisinger and Ropke [135], Ropke and Pisinger [145], Shaw [156], and Ropke and Pisinger [144]. They are general operators for a variety of routing problems. Two versions of the worst removal operator are designed especially for the GenConVRP. In the following, we describe the destroy operators in more detail.

Random removal The random removal operator selects customers randomly and removes them from each day until u removals have been made. The purpose of this operator is to diversify the search process.

Related removal The idea of the related removal operator is to remove customers that share similarities (Shaw [156]). This approach facilitates the interchange of customers and favors the grouping of similar customers on the same route. We define similarity between two customers i and j, $\mathscr{R}(i, j)$, in terms of geographical distance, difference in the demand, difference in the service time, and difference in the visit patterns (Kovacs et al. [107], Ropke

and Pisinger [145]). Parameters λ, μ, ν, and ξ control the weight of each measurement.

$$\mathscr{R}(i,j) = \lambda t_{ij} + \mu|\overline{q_i} - \overline{q_j}| + \nu|\overline{s_i} - \overline{s_j}| - \xi\gamma_{ij}, \tag{4.18}$$

where $\overline{q_i}$ and $\overline{s_i}$ are the average demand and the average service time of customer i among all days he is visited, respectively; γ_{ij}, is a measure of similarity in terms of visit frequency (see Chapter 3 and Kovacs et al. [107]). It is given by the number of days on which either both customers i and j are serviced or neither of them are serviced, i.e.,

$$\gamma_{ij} = |D| - \sum_{d \in D} \left| w_{id} - w_{jd} \right|. \tag{4.19}$$

The related removal operator starts by removing a randomly chosen customer from the solution and inserting him into the set of removed customers S. In each iteration, a customer is selected randomly from S and the similarity values between the chosen customer and all assigned customers are calculated. The customer with the highest similarity, i.e., lowest $\mathscr{R}(i,j)$, is removed from each day and added to set S. The process stops when at least u removals have been made.

Cluster removal The cluster removal operator (Ropke and Pisinger [144]) is applied on each day, one after another. The number of customers to remove on day d, u_d, is chosen from the interval $[\min\{0.1\sum_{i\in N} w_{id}, 30\}, \min\{0.4\sum_{i\in N} w_{id}, 60\}]$. On a given day, a route with at least three customers is selected randomly. The customers on that route are grouped into two geographical clusters by applying Kruskal's algorithm (Kruskal [109]) to find the minimum spanning tree among them[1]. We obtain two components that represent two clusters by deleting the longest edge in the tree. One of them is selected randomly and all customers in that cluster are removed.

The next route to be destroyed is the one that contains the customer with the smallest distance to a randomly chosen customer from the previously removed cluster. The process of finding clusters and removing them continues until at least u_d customers have been removed or until no route with at least three customers is left. More than u_d customers may be removed in order to remove complete clusters.

Worst removal – driver consistency The first worst removal operator uses a simple approach to reduce the number of different drivers per customer. Each customer i is associated

[1]The GenConVRP is defined on a digraph. We convert the digraph into an undirected graph by setting $t_{ij} = t_{ji} = \min\{t_{ij}, t_{ji}\}$.

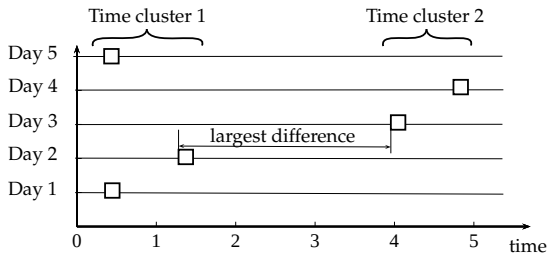


Figure 4.1 Time clusters in worst removal operator. The squares represent the customer's arrival times on the respective days.

with value $\mathscr{D}r(i)$. This value is composed of the number of drivers per customer z_i and customer i's arrival time difference l_i. It is given by

$$\mathscr{D}r(i) = z_i + \frac{l_i}{l_{max} + \varepsilon}, \tag{4.20}$$

where ε is a small number > 0.

Customers are sorted in decreasing order of $\mathscr{D}r(i)$ (i.e., in a lexicographic order of z_i first and l_i second). Customers are then removed, one after another, from the solution until at least u removals have been made and all assigned customers comply with the driver consistency requirement (constraints 4.9).

Worst removal – time consistency A second worst removal operator improves time consistency by repeatedly removing customers with large arrival time differences. The operator starts by dividing the days on which a customer requires service into two clusters. For each customer i, the arrival times are sorted in list L_i and the largest difference between two consecutive arrival times is identified. We obtain clusters, L_i^1 and L_i^2, by splitting L_i at the position with the largest difference in the arrival times. This approach is illustrated in Figure 4.1.

In a second step, we sort all customers in decreasing order of their maximum arrival time difference, l_i. Customers are then removed consecutively from the days in one of the clusters. Cluster L_i^1 is selected with probability $1 - \frac{|L_i^1|}{|L_i|}$ and cluster L_i^2 is selected with the complementary probability, $1 - \frac{|L_i^2|}{|L_i|}$. So, the more elements in a cluster, the lower the probability that it is selected for removal.

The operator stops when at least u removals have been made.

4.3.2 Repair operators

Repair operators are used to insert unassigned customers into the routing plan. We use six operators that are either greedy-based or regret-based. They are motivated by Pisinger and Ropke [135], Ropke and Pisinger [145], and Ropke and Pisinger [144]. The operators are applied to each day sequentially. Customers are inserted only with regard to travel time and driver consistency. Considering time consistency on the basis of partial solutions is not easy. (This is why most algorithms use a template-based approach.) Our approach is to ignore time consistency until all requests have been assigned and to perform a post-processing step on the complete solution in order to reduce the maximum arrival time difference. Section 4.3.5 explains how the improvement of time consistency is accomplished.

The sequence in which days are repaired is random. This approach diversifies the search process because the customer-to-driver assignment on one day affects the feasibility of assignments on other days. We also randomize the process of inserting customers (Pisinger and Ropke [135], Ropke and Pisinger [145]). Therefore, we change a customer's insertion cost c to $c+y$; y is a randomly chosen number in the interval $[-\eta c, \eta c]$. Parameter η controls the scale of randomization. A repair operator is randomized with a probability of 50%.

In all repair operators, customers are scheduled at their earliest possible position with regard to the time windows.

Greedy operators The applied greedy operators are based on Pisinger and Ropke [135] and Ropke and Pisinger [145]. The idea is to iteratively insert the customer that causes the least increase in the total travel time. Let c_{ik} be the change in the total travel time for inserting customer i at his cheapest position into route k; c_{ik} is set to infinity if an insertion violates the time windows, the maximum vehicle capacity Q, or the allowed travel time T. A new vehicle is added to the solution if there is no feasible insertion position. In each iteration, we insert the customer with the lowest insertion cost (i^*) into his cheapest position. That is,

$$i^* := \arg \min_{i \in N, k \in K} \{c_{ik}\}. \tag{4.21}$$

We add a large penalty ($M = 1000$) to c_{ik} if an insertion violates driver consistency. Furthermore, we reduce c_{ik} by M if i may not be assigned to additional drivers and k is already assigned to serve i on another day. In this way, we favor the insertion of customers that might be assigned to too many drivers in later iterations.

Two variants of the greedy operator are used: One as described above and one in

which we artificially reduce the maximum vehicle capacity Q. The lower Q, the fewer customers are visited on a route, and the smaller is the arrival time difference. In the modified greedy operator, we select the artificial vehicle capacity randomly in the interval $[\max\{\max_{i\in N,d\in D}\{q_{id}\},\frac{Q}{2}\},Q]$. A lower interval boundary of $\frac{Q}{2}$ proved to work well during the implementation of the algorithm. However, the lower bound for the artificial capacity must be at least $\max_{i\in N,d\in D}\{q_{id}\}$ to ensure that all customers can be served.

Regret operators Regret operators extend the greedy approach by taking into account the loss that might arise from postponing an insertion to later iterations (Potvin and Rousseau [137]). For each customer, we consider several insertion positions and insert the customer with the largest difference between his best insertion position and alternative positions. Let c_i^q denote the change in the total travel time for inserting customer i at his cheapest position into his q-cheapest route. In each iteration, customer i^* is inserted into his cheapest position according to:

$$i^* := \arg\max_{i\in N,k\in K}\left\{\sum_{h=2}^{\min(q,m)}(c_i^h - c_i^1)\right\}. \tag{4.22}$$

Parameter q defines the number of routes that are considered in the regret operator. Expensive insertions can be identified and avoided earlier with larger q values; m is the number of routes in the current solution.

We use four versions of the regret operator with $q = \{2,3,4,m\}$ as suggested in Pisinger and Ropke [135] and Ropke and Pisinger [145]. Driver consistency is taken into account by modifying the insertion costs as described above.

4.3.3 Adjustment of vehicle departure times

In the GenConVRP, we may adapt the vehicle departure times from the depot. Flexible departure times are necessary because of the AM/PM time windows in combination with the constraints that prohibit waiting times (e.g., it is not possible to visit only PM customers if departure times are fixed at zero.) Additionally, time consistency can be improved significantly by adjusting the vehicle departure times (Chapter 3, Kovacs et al. [107]). The problem of adjusting the departure times of a given solution is modeled as follows.

$$\textit{Minimize} \quad l_{max} \tag{4.23}$$

subject to:

$$a_{id} + s_{id} + t_{ij} = a_{jd} \qquad \forall\, (i,j) \in A_{kd}, i, j \in N, d \in D, \tag{4.24}$$

$$a_{id} + s_{id} + t_{i0} \leq T \qquad \forall\, i \in N, d \in D, \tag{4.25}$$

$$(a_{i\alpha} - a_{i\beta}) w_{i\alpha} w_{i\beta} \leq l_{max} \qquad \forall\, i \in N, \alpha, \beta \in D, \tag{4.26}$$

$$a_{id} \leq \frac{T}{2} \qquad \forall\, i \in N^{am}, d \in D, \tag{4.27}$$

$$a_{id} \geq \frac{T}{2} \qquad \forall\, i \in N^{pm}, d \in D, \tag{4.28}$$

$$a_{id} \geq t_{0i} \qquad \forall\, i \in N, d \in D. \tag{4.29}$$

The objective is to minimize the maximum arrival time difference l_{max} (4.23). Constraints (4.24) set the arrival times at the customers. Set A_{kd} contains all arcs that are traversed by vehicle k on day d. Inequalities (4.25) guarantee that vehicles return to the depot in time. The maximum arrival time difference is set by inequalities (4.26). AM/PM time windows are enforced by inequalities (4.27) and (4.28). Constraints (4.29) define the earliest possible arrival time at each customer.

The adjustment of the departure times can be performed by any linear programming (LP) solver. Yet, the LP becomes the bottleneck of the algorithm when it is integrated into the LNS. Instead of the LP, we use a fast greedy algorithm to reduce the maximum arrival time difference. The idea is to iteratively shift the departure time of the vehicle that visits the customer with the maximum arrival time difference. This approach is motivated by the algorithm described in Section 3.4.2 (Kovacs et al. [107]). However, our algorithm is more sophisticated. In Section 3.4.2, the focus is on the customer that violates the maximum allowed arrival time difference the most; the arrival times of other customers are disregarded. Therefore, the algorithm might reduce the arrival time difference at one customer but increase l_{max} at another. In our algorithm, l_{max} never deteriorates from one iteration to the other; additionally, in our experiments, it has always provided the optimal solution. The pseudocode of our approach is given in Algorithm 3.

The repair operators insert customers at their earliest possible positions. We start by eliminating the waiting times between the last AM customer and the first PM customer on each route (line 1). If i is the last AM customer on route k on day d and customer j is the first PM customer on the same route, we delay the departure time of that route by $a_{jd} - (a_{id} + s_i + t_{ij})$.

In each iteration of the algorithm, the customer with the current maximum arrival time difference i^{max} is used to initialize set BC (line 3 and 4). BC contains all customers that have

Algorithm 3 *adjustDepartureTimes*

Require: solution s
1: $eliminateWaitingTimes(s)$ ▷ waiting times are not allowed
2: **repeat**
3: $i^{max} = taskWithMaxATD(s)$
4: $BC = \{i^{max}\}$ ▷ blocking customers
5: $maxPF = maxATD(s)$
6: **for all** $j \in BC$ **do**
7: $maxPF = min\{maxPF, getMaxPF(j, BC, s)\}$
8: **end for**
9: **if** $(maxPF > \varepsilon)$ **then** ▷ $\varepsilon = 10^{-4}$
10: $applyPF(maxPF, BC, s)$
11: **else** ▷ pushing forward is not possible ⇒ try to pull backwards
12: $BC = \{i^{max}\}$
13: $maxPB = maxATD(s)$
14: **for all** $j \in BC$ **do**
15: $maxPB = min\{maxPB, getMaxPB(j, BC, s)\}$
16: **end for**
17: **if** $(maxPB > \varepsilon)$ **then**
18: $applyPB(maxPB, BC, s)$
19: **end if**
20: **end if**
21: **until** $maxPF > \varepsilon \vee maxPB > \varepsilon$

to be moved in order to reduce l_{max}. We call these customers blocking customers. For each customer in BC, we compute the amount of time he can be pushed forward without violating any constraint or increasing the current maximum arrival time difference (line 7)[2].

The computation is done in function $getMaxPF(j, BC, s)$. The function first identifies the day (or the days) on which customer j is visited earliest. The maximum push forward $pf(j,k)$ of j's route on the respective day (or days) is computed with respect to all other customers k on the same route. The upper bound for $pf(j,k)$ is the value that puts the arrival time difference of k on a level with the arrival time difference of j. Consider, for example, two routes $0-a-b-0$ and $0-b-c-a-0$ that are executed on two days, respectively; customer a has the largest difference in the arrival times. The algorithm reduces l_a by pushing forward the route on day 1. This, however, increases l_b. The maximum push forward $pf(j,k)$ is limited by the value that sets $l_a = l_b$.

Customers that block the pushing forward of task j (i.e., $pf(j,k) < \varepsilon$) are added to set

[2]Waiting times are not allowed. So, shifting customer j's starting time on a given day is equivalent to shifting the departure time of the vehicle that visits j.

BC. We obtain the maximum push forward *maxPF* by evaluating the push forward of all customers in set *BC* with respect to all customers that might block the shifting and then take the minimum. Details on how the $pf(j,k)$ values are computed are given in Section 4.3.4.

By being greedy and shifting routes forward as much as possible, we can get trapped in a local optimum. Lines 12 - 18 of the algorithm serve as an escape mechanism. Effectively, we try to reduce the delay that we have applied before.

The algorithm stops when neither pushing routes forward nor pulling them backward can reduce the maximum arrival time difference.

Figure 4.2 illustrates how the algorithm works. The figure at the top shows the routing plan for each day in a three-day planning horizon; seven customers are visited. The three tables show three iterations of the algorithm. We report the arrival times at each customer on each day, the arrival time difference for each customer, *maxPF*, *maxPB*, and set *BC*. In iteration 1, customers 2 and 5 have an arrival time difference of 4; i^{max} is set to 5, arbitrarily. In order to reduce l_{max}, customer 5's arrival time on day 3 has to be pushed forward. Customer 2 is a blocking customer because its latest arrival time is on day 3 and its arrival time difference is 4. The algorithm delays the routes on day 1 and day 3 by 0.2; a larger *maxPF* would violate the shift length constraint on day 1. In iteration 2, l_{max} is still 4 at customer 2. So, i^{max} is set to 2. Reducing l_{max} by pushing routes forward is infeasible; we are in a local optimum. Therefore, the route on day 3 is pulled backward by 0.1. This reduces l_{max} to 3.9. In iteration 3, customers 2 and 5 have an arrival time difference of 3.9 but both *maxPF* and *maxPB* are zero; the algorithm stops.

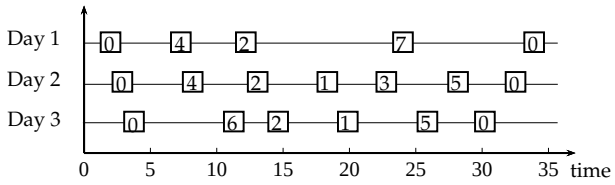
4.3.4 Computing the shifts in the vehicle departure times

Algorithm 3 in Section 4.3.3 reduces the maximum arrival time difference by iteratively shifting the vehicle departure times. In order to compute the maximum amount of time customer $j's$ starting time can be pushed forward with respect to customer k, $pf(j,k)$, we have to distinguish two scenarios.

In the first scenario, the day on which customer k is visited earliest is not the day on which customer j is visited earliest. This scenario is illustrated in Figure 4.3. In this case we solve following equation to get $pf(j,k)$:

$$l_j - pf(j,k) = l_k + pf(j,k) - (a_k^{latest} - a_k^{current}). \tag{4.30}$$

Here, $a_k^{current}$ denotes the arrival time at customer k on the current day and a_k^{latest} is the latest arrival time at customer k. The left-hand side is the arrival time difference of j and



Iteration 1

Day	1	**2**	3	4	**5**	6	7
1		11.7		6.9			22
2	17.5	14.5	22.7	9.7	29.3		
3	18.7	15.7			25.3	12	
l_i	1.2	**4**	0	2.8	**4**	0	0

i^{max}	5
BC	5,2
maxPF / *maxPB*	0.2/-

Iteration 2

Day	1	**2**	3	4	5	6	7
1		11.9		7.1			22.2
2	17.5	14.5	22.7	9.7	29.3		
3	18.9	15.9			25.5	12.2	
l_i	1.4	**4**	0	2.6	3.8	0	0

i^{max}	2
BC	2
maxPF / *maxPB*	0/0.1

Iteration 3

Day	1	**2**	3	4	**5**	6	7
1		11.9		7.1			22.2
2	17.5	14.5	22.7	9.7	29.3		
3	18.8	15.8			25.4	12.1	
l_i	1.3	**3.9**	0	2.6	**3.9**	0	0

i^{max}	2
BC	2,5
maxPF / *maxPB*	0/0

Figure 4.2 Three iterations of Algorithm 3.

the right-hand side is the arrival time difference of k after delaying j's route on its earliest day by $pf(i,k)$. The push forward that leads to a maximum reduction in l_j with respect to customer k is

$$pf(j,k) = \frac{l_j - l_k + (a_k^{latest} - a_k^{current})}{2}. \tag{4.31}$$

In the second scenario, the day on which customer k is visited earliest is the day on which customer j is visited earliest (see Figure 4.4). We set $pf(j,k)$ by solving

$$l_j - pf(j,k) = (a_k^{current} + pf(j,k)) - a_k^{2^{nd}earliest}, \tag{4.32}$$

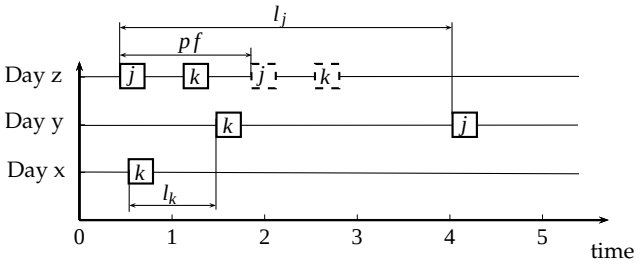


Figure 4.3 Departure time adjustment: scenario 1.

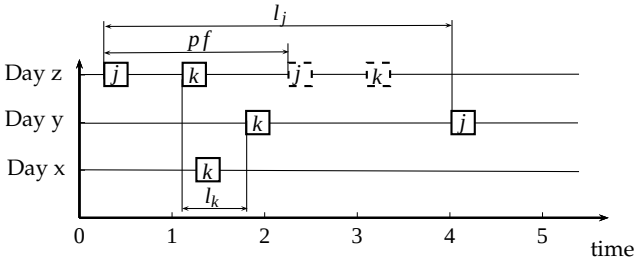


Figure 4.4 Departure time adjustment: scenario 2.

where $a_k^{2^{nd}earliest}$ is the second earliest arrival time of customer k. The push forward that leads to a maximum reduction in l_j with respect to customer k is

$$pf(j,k) = \frac{l_j + a_k^{2^{nd}earliest} - a_k^{current}}{2}. \tag{4.33}$$

The approach of pulling routes backward follows the same principle as for pushing routes forward. Again, we have to distinguish two scenarios. First, customer k's arrival time is not the latest on the day j is visited latest. In this case the pull backward is

$$pb(j,k) = \frac{l_j - l_k + (a_k^{current} - a_k^{2^{nd}earliest})}{2}. \tag{4.34}$$

Otherwise, the pull backward is

$$pb(j,k) = \frac{l_j + a_k^{current} - a_k^{2^{nd}earliest}}{2}. \tag{4.35}$$

Algorithm 4 *improveTimeConsistency*

Require: feasible solution s

1: **repeat**
2: $f' = f(s)$
3: $i^{max} = taskWithMaxATD(s)$
4: $aat = averageArrivalTime(i^{max}, s)$
5: $eat = earliestArrivalTime(i^{max}, s)$
6: $lat = latestArrivalTime(i^{max}, s)$
7: **if** $(aat - eat > lat - aat)$ **then**
8: $d = dayEarliestArrival(i^{max}, s)$
9: **else**
10: $d = dayLatestArrival(i^{max}, s)$
11: **end if**
12: $apply2opt(i^{max}, d, s)$
13: **until** $f(s) < f'$ or we have a special case as shown in Figure 4.6

4.3.5 Improvement of time consistency

The repair operators insert customers into the routing plan only with regard to travel time. So, new solutions might be poor in terms of time consistency. If a new solution is feasible, we reduce the customers' arrival time differences by reversing parts of selected routes. This approach corresponds to repeatedly executing a 2-opt operator (Lin [116]) on the route that contains the customer with the current maximum arrival time difference. The pseudocode is presented in Algorithm 4.

First, we identify the customer with the maximum arrival time difference i^{max} (line 3). Then, we use the earliest, average, and latest arrival times of i^{max} to select a day to reverse parts of the route that contains i^{max}. By using the selection criterion in line 7, we select the day that has the highest chance to reduce the variance in the arrival times. This is the day on which customer i^{max} is visited either latest or earliest. The 2-opt operator is applied to the route that contains i^{max} on the selected day (line 12). A reversal may affect either only customers $\in N^{am}$ if $i^{max} \in N^{am}$ or only customers $\in N^{pm}$ if $i^{max} \in N^{pm}$.

A route that improves the solution is accepted as the basis for further reversals. Large reductions in the maximum arrival time difference can be expected by reversing long sequences. Therefore, it is important to start the search with the longest sequence and then reduce the length of the sequence until all 2-opt moves have been tried.

The desired result after one iteration is shown in Figure 4.5. The black squares represent the starting times of customer i^{max}. The route selected for the 2-opt operator is the one that contains i^{max} on day 1. The procedure stops when no improvement is obtained by reversing

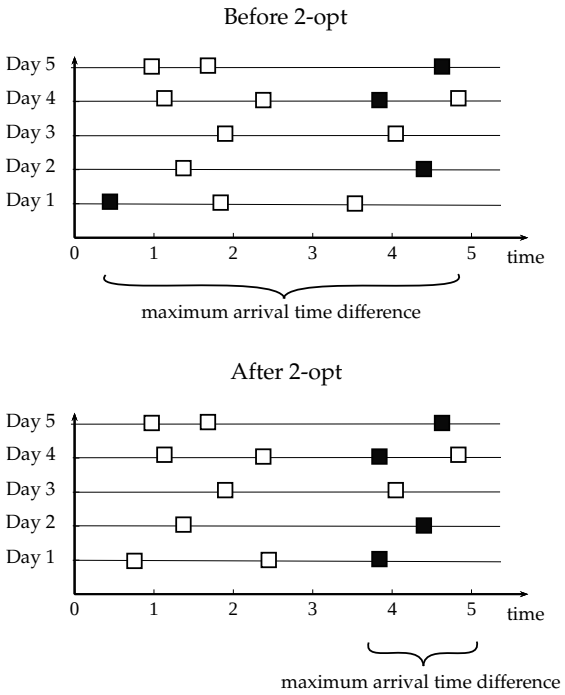


Figure 4.5 2-opt operator to improve time consistency. The black squares represent the customer with the maximum arrival time difference. The white squares represent other customers.

parts of only one route. However, with this stopping criterion the algorithm fails to improve solutions like in Figure 4.6. In this example, the difference in the starting time on the selected day and at least one other day is close to zero. The solution would not improve by reversing only one route. In such cases, we ignore all days where i^{max} has the same arrival time as the day on which 2-opt is performed. With this modification, we can improve the solution step-by-step.

The 2-opt operator is costly in terms of computation time because the evaluation of one reversal involves the entire solution and requires the readjustment of vehicle departure times. In Section 4.3.6, we describe an approach to speed up the algorithm by avoiding the execution of reversals that cannot lead to improved solutions.

4.3.6 Speeding up the 2-opt operator

The 2-opt operator to improve time consistency in Section 4.3.5 is costly in terms of computation time. We can reduce the computational burden significantly by detecting unnecessary reversals in advance. If the resulting increase in travel time (Δtt) would outweigh the result-

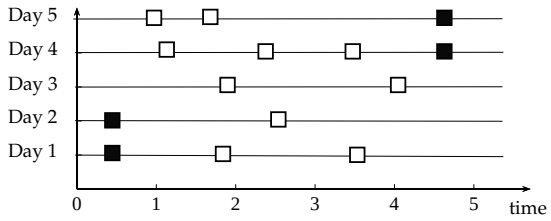


Figure 4.6 Special case for 2-opt operator. The black squares represent the customer with the maximum arrival time difference. The white squares represent arbitrary customers.

ing reduction in the maximum arrival time difference (Δl_{max}), $\alpha \Delta tt > (1-\alpha)\Delta l_{max}$, we can skip the current reversal. The computation of Δtt is fast because in the worst case it is linear in the number of customers on the route that is affected by the reversal. (The computation requires constant time if the distance matrix is symmetric.) Yet, Δl_{max} has to be estimated.

Proposition For each pair of customers (i, j) and each pair of days (x,y) where the customer pair is visited by the same driver, without taking the route on which 2-opt is to be performed into account, we can calculate a lower bound $\underline{\Delta l_{max}}$ for Δl_{max} as follows:

$$\underline{\Delta l_{max}} = \max_{\substack{x,y \in D, x \neq y \\ i,j \in N, i \neq j \\ k_{ix}=k_{jx}, x \neq d' \vee k_{ix} \neq k' \\ k_{iy}=k_{jy}, y \neq d' \vee k_{iy} \neq k'}} \frac{|a_{jy}-a_{iy}|-|a_{jx}-a_{ix}|}{2}. \tag{4.36}$$

Parameter a_{ix} is the arrival time at customer i on day x and k_{ix} gives the corresponding driver. Route k' is the route and day d' is the day on which the 2-opt operator is performed.

Proof Each pair of customers (i, j) that is visited on days x and y by the same driver may either be visited in the same order on both days (scenario 1 in Figure 4.7) or in different order (scenario 2 in Figure 4.7). In the first scenario, the arrival time difference that cannot be eliminated by shifting the departure times is $\frac{(a_{jy}-a_{iy})-(a_{jx}-a_{ix})}{2}$. Similarly, in scenario 2, the maximum arrival time difference that cannot be eliminated by shifting is $(a_{jy}-a_{iy})-\frac{(a_{jy}-a_{iy})-(a_{ix}-a_{jx})}{2}$ which simplifies to $\frac{(a_{jy}-a_{iy})-(a_{jx}-a_{ix})}{2}$, yielding the same term as in scenario 1. By taking the maximum value across all pairs of customers (i, j) and pairs of days (x,y), expression (4.36) yields a valid lower bound for l_{max}. □

If the vehicle departure times are fixed to zero (e.g., as in the original ConVRP) a lower bound for Δl_{max} is the maximum arrival time difference among all customers that are not

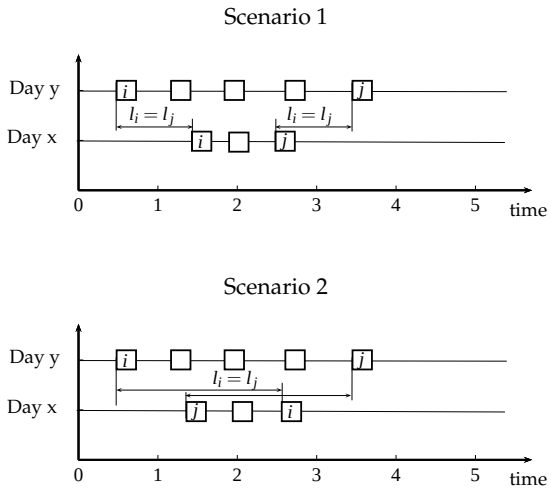


Figure 4.7 Upper bound on maximum arrival time difference.

affected by the reversal, i.e., are not visited on the route on which the 2-opt operator is applied.

4.3.7 Acceptance criterion

The decision whether or not a new solution s' is accepted as the current incumbent s is based on a simulated annealing acceptance criterion (Kirkpatrick et al. [97]). The evaluation function $g(s)$ is composed of the objective function $f(s)$ and a penalty term for violations of the driver consistency. That is,

$$g(s) = f(s) + (e^{\frac{max\{0, z^{max} - W\}}{p_{penalty}}} - 1) i_{LNS}. \tag{4.37}$$

z^{max} is the maximum number of drivers per customer ($z^{max} = \max_{i \in N}\{z_i\}$; z_i denotes the number of different drivers per customer i). W is the number of allowed drivers per customer. Parameter $p_{penalty}$ is set such that the penalty for violating the driver consistency by one driver in the last iteration, i_{LNSmax}, is equal to the objective value of the initial solution, $s^{initial}$. (The initial solution is feasible, i.e., $f(s^{initial}) = g(s^{initial})$.) This yields

$$p_{penalty} = \frac{1}{\ln \frac{f(s^{initial})}{i_{LNSmax}} + 1}. \tag{4.38}$$

Improving solutions (i.e., $g(x') < g(x)$) are always accepted as current incumbents. Using a simulated annealing rule, deteriorating solutions are accepted with probability

$e^{-(g(s')-g(s))/\hat{t}}$. Parameter $\hat{t}$ is the current temperature in the annealing process. It is initialized by the term

$$\hat{t} = -\frac{w_{\hat{t}}}{\ln 0.5} f(s^{initial}). \tag{4.39}$$

Thus, a $w_{\hat{t}}$ % worse solution is accepted with a probability of 50% (Pisinger and Ropke [135], Ropke and Pisinger [145]). The cooling rate is controlled by parameter c, i.e., $\hat{t}$ is replaced by $\hat{t}c$ in each LNS iteration.

The best found solution s^{best} is replaced if s' complies with all constraints and $f(s') < f(s^{best})$.

4.3.8 Initial solution

Parameter α in the objective function has a strong impact on the structure of a good solution. A solution that visits each customer on a separate route is optimal if α is zero. If $\alpha = 1$, several customers are consolidated on each route. Due to this difference in the solution structure, better final solutions can be obtained by using a proper starting solution. This is especially true when α is very small and the maximum arrival time difference is close to zero in the optimal solution.

We construct two initial solutions, one that is travel-time oriented and one that is time-consistency oriented. The LNS is started with the solution that has a lower objective value.

Travel-time oriented solution. We apply the greedy repair operator (with the original vehicle capacity) from Section 4.3.2 and the worst driver destroy operator from Section 4.3.1 to construct a feasible initial solution with reasonable total travel time. The algorithm iteratively applies the greedy operator[3] to insert customers into routes and the worst driver destroy operator to remove all customers that violate the driver consistency requirement. One of the removed customers is assigned to a new vehicle and the remaining customers are inserted by the greedy operator into their cheapest position. The process stops when the solution complies with the driver consistency constraint. Time consistency is improved as described in Section 4.3.5.

Time-consistency oriented solution. An initial solution with a low arrival time difference is generated by assigning each customer that requires at least two visits in the planning

[3]Driver consistency is considered in the greedy operator. However, there is no guarantee that the allowed number of drivers per customer is not exceeded.

simulated annealing		repair operators	related destroy operator			
w_t	c	η	λ	μ	ν	ξ
0.05	0.9999	0.1	12	4	4	7

Table 4.1 Parameter setting LNS.

period to a separate route. Customers that require only one visit are then inserted by the greedy operator and time consistency is improved as described in Section 4.3.5.

4.3.9 Further improvements

The LNS is designed for commercial applications where cost is the most important aspect. Therefore, in its pure version, the algorithm performs poorly if the focus is on time consistency rather than on travel time (i.e., α is very small). Experiments show that we can improve the time consistency by artificially restricting the search space by reducing the number of allowed drivers per customer. With this extension, we improve the time consistency without losing much travel time for a broad range of α settings.

The procedure is as follows. Regardless of the number of allowed drivers W per customer, we start the search (including the initial solution) by restricting the driver consistency to one driver per customer. The restriction applies only to the repair operators and the worst driver destroy operator – the evaluation function is not affected. We increase the number of allowed drivers by one if the best found solution has not improved for $\frac{i_{LNSmax}}{5}$ iterations (i_{LNSmax} is the maximum number of LNS iterations). In the last 5% of the search, we again allow W drivers per customer.

4.4 Computational results

Experiments on different ConVRP variants are conducted by running a C++ implementation of the LNS on Intel Xeon X5550 computers with 2.67GHz. The number of LNS iterations is 50000. Results are obtained by taking the average of 10 runs per instance.

LNS algorithms tend to be robust with respect to the chosen control parameters. Therefore, we kept the efforts to tune them low. Each parameter was set to either an initial guess, to a value defined in Chapter 3, or to a value reported in the literature (Ropke and Pisinger [145]). Only the parameters listed in Table 4.1 were adjusted during the development of the algorithm. The same set of parameters is used for all experiments.

4.4.1 Data sets

We apply the LNS to different variants of the ConVRP in order to examine its robustness. The characteristics of the respective data sets are described below.

Instances for ConVRP. A benchmark data set for the ConVRP is provided by Groër et al. [75]. The data set consists of 12 instances with 50 to 199 customers and a planning horizon of five days. Customer requirements are fixed in advance for the entire planning period. The probability that a customer requires service on a given day was set to 70% when instances were generated. We refer to this probability as service frequency. The higher the service frequency the more customers are serviced at least twice, i.e., more customers demand consistent service. Furthermore, the daily routing plans become similar with higher service frequency. No bound on the maximum arrival time difference is given in the benchmark data set. We set the maximum arrival time difference according to the results obtained in Groër et al. [75]. We refer to the benchmark data set as data set *A*.

Instances for ConVRP with adaptable departure times. The original ConVRP instances are extended as described in Chapter 3 (Kovacs et al. [107]). The extension refers to the service frequency that was set to 50% and to 90%, respectively; the service frequency in the original instances is 70%. Also, different bounds on the maximum arrival time difference are suggested. We use the instances with the tightest bounds since these are the most challenging. The vehicle departure time may be adjusted to improve time consistency. We refer to the data sets with 50%, 70%, and 90% service frequency as $B_{0.5}$, $B_{0.7}$, and $B_{0.9}$, respectively.

Instances for GenConVRP. ConVRP instances are turned into GenConVRP instances by adding AM/PM time windows. In each instance, we associate customers with even IDs with AM time windows and customers with odd IDs with PM time windows. The shift duration T is not bounded in all ConVRP instances. In instances without bounds on the shift length, we set T to 1000. This constraint is very loose but we obtain reasonable time window ranges. The data sets with time windows are named $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, with respect to the service frequencies.

There are no reference solutions for the GenConVRP. Therefore, we also use 10 small test instances that can be solved by MIP solvers. The small instances contain 10 to 12 customers that are visited over a planning horizon of three days. They are derived from the

small ConVRP instances presented in Groër et al. [75]. We refer to the small GenConVRP data set as C_{small}.

Each instance is tested with different bounds on the number of allowed drivers (W) and with different parameters α in the objective function (i.e., different emphasis on time consistency).

4.4.2 Results for ConVRP benchmark instances (data set *A*)

The performance of the LNS on the ConVRP benchmark instances (data set *A*) is compared to three template-based algorithms: the record-to-record travel algorithm for the ConVRP (ConRTR) (Groër et al. [75]), the template-based tabu search (TTS) (Tarantilis et al. [170]), and the template-based adaptive large neighborhood search (TALNS) (Chapter 3, Kovacs et al. [107]). The results of the comparison are reported in Table 4.2 and Table 4.3. For each algorithm and each instance, we give the maximum arrival time difference (l_{max}) and the total travel time plus the total service time[4] (TT) in Table 4.2. All algorithms are stochastic with the exception of the ConRTR. TT_{avg} denotes the average results over 10 runs. TT_{min} is the best result out of five runs for the TTS and out of 10 runs for the LNS. If the computation time is available, we report it in seconds (CPU).

In Table 4.3, TT Imp(%) denotes the improvement in the total travel time that is obtained by the LNS compared to the respective algorithms. The improvement between LNS and TTS refers to TT_{min}, the improvement between the other algorithms to TT_{avg}. The average results over all instances are given in the last row of the table. The last column gives the improvement in the arrival time difference (l_{max} Imp(%)) obtained by LNS compared to TALNS.

The objective in the ConVRP is the minimization of the total travel time while bounding the maximum arrival time difference. LNS is modified to check the maximum arrival time difference for feasibility before replacing the best found solution and parameter α is set to 0.3.

The comparison of LNS to the template-based approaches shows the advantage of a non-template-based approach. LNS performs better than ConRTR and TTS on all instances with an average improvement of 6.16% and 2.48%, respectively. The difference in the total travel times is minor between LNS and TALNS. However, the arrival time difference is on average 20.26% lower with LNS than with TALNS. (The total travel time can be improved at the cost of a higher arrival time difference by increasing parameter α in the LNS.) For a

[4]We add the total service time to the total travel time to be consistent with the literature.

	ConRTR		TTS			TALNS			LNS			
Instances	TT_{avg}	l_{max}	TT_{min}	l_{max}	CPU	TT_{avg}	l_{max}	CPU	TT_{avg}	TT_{min}	l_{max}	CPU
Ch_1_5_0.7	2282.14	24.38	2210.56	21.99	80.00	2194.93	23.72	5.45	2132.47	2124.21	20.18	15.13
Ch_2_5_0.7	3872.86	34.26	3622.71	27.75	93.00	3605.03	31.86	14.69	3591.04	3540.80	25.41	18.82
Ch_3_5_0.7	3628.22	22.87	3451.10	21.92	369.00	3338.03	22.21	25.58	3310.20	3280.47	21.22	40.22
Ch_4_5_0.7	4952.91	27.53	4572.00	25.15	388.00	4598.78	24.19	84.31	4557.72	4473.31	19.85	62.71
Ch_5_5_0.7	6416.77	26.93	5732.62	19.99	550.00	5685.55	22.69	122.24	5675.16	5632.22	17.69	87.26
Ch_6_5_0.7	4084.24	63.47	4096.87	55.38	70.00	4051.50	63.26	6.63	4073.92	4070.72	40.39	14.58
Ch_7_5_0.7	7126.07	83.96	6752.36	63.28	161.00	6805.99	76.62	18.33	6713.15	6673.61	43.72	19.68
Ch_8_5_0.7	7456.19	73.04	7279.39	62.01	539.00	7192.45	65.97	32.24	7197.32	7126.29	50.27	31.27
Ch_9_5_0.7	11033.54	106.43	10585.10	84.76	947.00	10450.04	88.85	97.39	10444.95	10390.70	59.07	50.24
Ch_10_5_0.7	13916.80	60.17	13120.40	57.17	1052.00	13238.57	57.95	146.32	13042.63	12955.10	55.19	78.73
Ch_11_5_0.7	4753.89	16.10	4721.09	15.68	480.00	4506.59	15.33	35.96	4588.43	4471.22	13.91	83.63
Ch_12_5_0.7	3861.35	17.58	3607.88	16.91	172.00	3530.16	16.50	25.60	3585.73	3521.88	13.63	27.41
Average	6115.42	46.39	5812.67	39.33	408.42	5766.47	42.43	51.23	5742.73	5688.38	31.71	44.14

Table 4.2 Comparison with other algorithms on data set *A*; the names of the instances are abbreviated.

	TT Imp(%)			l_{max} Imp(%)
Instances	to ConRTR	to TTS	to TALNS	to TALNS
Christofides_1_5_0.7	6.56	3.91	2.85	14.94
Christofides_2_5_0.7	7.28	2.26	0.39	20.26
Christofides_3_5_0.7	8.77	4.94	0.83	4.44
Christofides_4_5_0.7	7.98	2.16	0.89	17.95
Christofides_5_5_0.7	11.56	1.75	0.18	22.04
Christofides_6_5_0.7	0.25	0.64	-0.55	36.15
Christofides_7_5_0.7	5.79	1.17	1.36	42.93
Christofides_8_5_0.7	3.47	2.1	-0.07	23.79
Christofides_9_5_0.7	5.33	1.84	0.05	33.52
Christofides_10_5_0.7	6.28	1.26	1.48	4.77
Christofides_11_5_0.7	3.48	5.29	-1.82	9.28
Christofides_12_5_0.7	7.14	2.38	-1.57	17.40
Average	6.16	2.48	0.34	20.62

Table 4.3 Improvement of LNS over other algorithms on data set *A*; TT is the total travel time and l_{max} is the maximum arrival time difference.

sample of ten runs, the average variation coefficient (standard deviation / mean) of the LNS is 0.0077; the reported variation coefficient for the TALNS is 0.0097 (see Chapter 3 and Kovacs et al. [107]). This result supports previous findings that LNS, if properly designed, is a very robust metaheuristic. Additionally, with an average computation time of 44 seconds, LNS is faster than the other algorithms.

4.4.3 Results for ConVRP with adaptable starting times (data sets $B_{0.5}$, $B_{0.7}$, and $B_{0.9}$)

Tables 4.4, 4.5, and 4.6 compare the LNS and the TALNS when the vehicle departure time is adaptable. The departure times are adjusted by an exact approach in the TALNS and

by Algorithm 3 (described in Section 4.3.3) in the LNS. LNS accepts only solutions that comply with the time consistency requirement as best found solutions; parameter α is set to 0.2. The tables present the average total travel time plus the total service time (TT_{avg}), the maximum arrival time difference (l_{max}), and the computation time (CPU(s)) for LNS and TALNS, respectively. The last two columns show the respective improvements in TT_{avg} and l_{max} of LNS over TALNS.

LNS performs well on this ConVRP variant despite tight bounds on l_{max}. An interesting pattern can be observed when comparing results with different service frequencies. LNS outperforms TALNS on data set $B_{0.5}$ with an average improvement of 1.42% in the total travel time and an average reduction of 16.41% in the maximum arrival time difference. The advantage of LNS diminishes to 0.67% with respect to TT and to 6.41% with respect to l_{max} on data set $B_{0.7}$. On data set $B_{0.9}$, TALNS is superior to LNS with respect to TT; TALNS performs, on average, 1.59% better than LNS. Nevertheless, the solutions found by LNS have, on average, 7.48% smaller maximum arrival time differences.

The template-based approaches are suitable when service frequency is high. With high service frequencies the variation in the customer requirements and, therefore, in the daily routing plans is low. Accordingly, the template gives a good approximation of the daily routes. With low service frequencies, there are more customers that require only one service during the planning horizon. LNS inserts all customers into the routing plan first, and improves the time consistency in a second step. In this way, there is a greater flexibility to place customers that are not linked to consistency requirements at positions that support the time consistency of frequent customers.

Flexible departure times improve arrival time consistency (Chapter 3, Kovacs et al. [107]). The variation in each driver's departure times is modest; the average ratio between the standard deviation of the departure times and the shift length (among all instances in which the shift length is bounded) is 5.26%, 3.97%, and 2.39% for data sets $B_{0.5}$, $B_{0.7}$, and $B_{0.9}$, respectively. For example, if the shift is 10 hours, than a ratio of 5% indicates that each driver starts his route with a deviation of $\pm$30 minutes from his average departure time.

4.4.4 Results for GenConVRP

In experiments on the GenConVRP, the focus is on examining the effect of the number of allowed drivers per customer (W) and the importance of time consistency (expressed by parameter α) on the results. We use a normalized objective function $f^n(s)$ in order to obtain similar dimensions for the total travel time and for the maximum arrival time difference

	TALNS			LNS			TT Imp (%)	l_{max} Imp (%)
Instances	TT_{avg}	l_{max}	CPU(s)	TT_{avg}	l_{max}	CPU(s)	to TALNS	to TALNS
Christofides_1_5_0.5	1685.19	25.80	103.00	1677.59	17.78	60.02	0.45	31.07
Christofides_2_5_0.5	2619.03	16.01	52.70	2597.02	12.20	44.48	0.84	23.79
Christofides_3_5_0.5	2705.88	20.93	163.77	2670.03	19.53	150.64	1.32	6.71
Christofides_4_5_0.5	3543.11	16.83	157.30	3376.09	13.26	148.27	4.71	21.18
Christofides_5_5_0.5	4090.38	13.17	231.89	4060.42	12.75	199.97	0.73	3.17
Christofides_6_5_0.5	2878.42	34.71	106.76	2891.35	30.70	30.96	-0.45	11.53
Christofides_7_5_0.5	4764.38	24.84	40.47	4678.29	18.73	33.18	1.81	24.61
Christofides_8_5_0.5	5355.78	36.71	114.25	5357.87	28.80	88.64	-0.04	21.55
Christofides_9_5_0.5	7493.43	36.53	119.10	7422.41	31.40	94.16	0.95	14.04
Christofides_10_5_0.5	9628.98	28.19	228.58	9402.92	27.31	175.03	2.35	3.13
Christofides_11_5_0.5	3456.14	16.01	184.07	3317.011	12.68	326.45	4.03	20.78
Christofides_12_5_0.5	2905.22	9.32	108.89	2894.34	7.89	91.62	0.37	15.33
Average	4260.50	23.25	134.23	4195.44	19.42	120.28	1.42	16.41

Table 4.4 Results for data set $B_{0.5}$.

	TALNS			LNS			TT Imp (%)	l_{max} Imp (%)
Instances	TT_{avg}	l_{max}	CPU(s)	TT_{avg}	l_{max}	CPU(s)	to TALNS	to TALNS
Christofides_1_5_0.7	2131.91	15.00	74.53	2123.02	13.31	73.33	0.42	11.23
Christofides_2_5_0.7	3615.43	15.43	47.85	3599.69	13.81	54.79	0.44	10.52
Christofides_3_5_0.7	3343.10	14.28	120.65	3308.12	14.04	224.61	1.05	1.69
Christofides_4_5_0.7	4691.71	10.50	199.10	4562.46	9.76	245.18	2.75	7.09
Christofides_5_5_0.7	5716.18	10.67	212.32	5682.84	11.06	264.21	0.58	-3.61
Christofides_6_5_0.7	4064.34	31.73	137.50	4092.89	21.10	60.85	-0.70	33.51
Christofides_7_5_0.7	6833.13	28.76	37.58	6693.43	23.87	41.64	2.04	16.98
Christofides_8_5_0.7	7225.29	27.39	141.94	7192.44	27.31	132.23	0.45	0.31
Christofides_9_5_0.7	10579.78	29.98	225.29	10461.29	30.88	132.07	1.12	-2.99
Christofides_10_5_0.7	13217.64	28.71	281.33	13050.14	27.76	184.03	1.27	3.31
Christofides_11_5_0.7	4772.98	6.40	177.24	4810.15	6.53	495.98	-0.78	-2.04
Christofides_12_5_0.7	3548.52	6.31	113.48	3571.40	6.25	91.29	-0.65	0.90
Average	5811.67	18.76	147.40	5762.32	17.14	166.68	0.67	6.41

Table 4.5 Results for data set $B_{0.7}$.

	TALNS			LNS			TT Imp (%)	l_{max} Imp (%)
Instances	TT_{avg}	l_{max}	CPU(s)	TT_{avg}	l_{max}	CPU(s)	to TALNS	to TALNS
Christofides_1_5_0.9	2499.17	11.71	107.22	2499.04	9.52	82.85	0.01	18.72
Christofides_2_5_0.9	4045.92	9.07	148.99	4051.71	8.88	64.91	-0.14	2.02
Christofides_3_5_0.9	4018.88	9.49	135.69	4035.72	8.56	281.54	-0.42	9.84
Christofides_4_5_0.9	5027.71	7.97	267.20	5049.91	7.37	375.27	-0.44	7.50
Christofides_5_5_0.9	6623.67	3.88	427.53	6882.86	3.96	390.74	-3.91	-2.07
Christofides_6_5_0.9	4767.65	20.99	145.76	4772.33	20.99	44.64	-0.10	0.00
Christofides_7_5_0.9	7741.18	15.93	111.90	7761.11	15.57	42.80	-0.26	2.27
Christofides_8_5_0.9	8758.24	21.95	164.40	8770.39	19.99	122.08	-0.14	8.95
Christofides_9_5_0.9	12648.34	19.09	159.07	12556.00	18.99	149.27	0.73	0.54
Christofides_10_5_0.9	16076.38	16.14	282.28	16146.57	16.52	218.61	-0.44	-2.37
Christofides_11_5_0.9	5043.93	7.46	98.69	5666.20	5.06	491.65	-12.34	32.19
Christofides_12_5_0.9	4015.40	3.72	122.73	4081.56	3.11	163.78	-1.65	16.34
Average	6772.21	12.28	180.96	6856.12	11.54	202.34	-1.59	7.83

Table 4.6 Results for data set $B_{0.9}$.

regardless of the number of customers and the length of the planning horizon:

$$f^n(s) = \alpha \frac{10 \sum_{d \in D} \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ijkd}}{\sum_{d \in D} \sum_{i \in N} w_{id}} + (1-\alpha) l_{max}. \tag{4.40}$$

$f^n(s)$ sets ten times the average travel time between two customers in relation to the maximum arrival time difference.

Comparison between LNS and CPLEX on data set C_{small}

In order to verify the quality of the solutions obtained by LNS, we solve the small instances (C_{small}) and evaluate the gap to the solutions found by CPLEX. Table 4.7 shows the average results over all instances in data set C_{small}. The number of allowed drivers per customer is set to one, two, and three, respectively. *OV* denotes the average objective value over all instances. The results are obtained by CPLEX 12.5; all instances are solved to proven optimality. Columns dev_{LNS} give the average deviation in the objective value between LNS and CPLEX. The last column shows the improvement in *OV* that is obtained by allowing two drivers per customer instead of one.

Increasing the number of allowed drivers per customer from one to two leads to an average improvement of at least 2%. Allowing three drivers per customer has no effect on the solutions. So, better solutions are obtained by slightly increasing the number of drivers per customer, regardless of the decision maker's preference. Yet, allowing too many drivers per customer deteriorates driver consistency without improving the objective value.

The number of drivers also affects the relationship between the total travel time (TT) and the maximum arrival time difference (l_{max}). Decreasing α from 0.999 to 0.001, i.e., decreasing l_{max} by 100%, causes an increase of 19% in TT with one driver (see Table 4.7). With two and three drivers, TT increases by only 14%.

The LNS algorithm is able to find the best known solutions for the majority of the settings; the average computation time is 1.7 seconds. LNS fails to find the highest quality solutions only when the focus is fully on time consistency, i.e., the optimal l_{max} is close to zero.

The average computation times required by CPLEX are shown in Figure 4.8 on a logarithmic scale. Multi-threading was enabled in our experiments; the reported values refer to the total computation time across all threads. The computation times are given as a function of α and the number of allowed drivers W per customer. If W is one, the computation time increases from 21.9 seconds to 92.7 seconds as α decreases from 0.999 to 0.001; the peak

	one driver				two drivers				three drivers				Imp(%)
α	TT	l_{max}	OV	$dev_{LNS}(\%)$	TT	l_{max}	OV	$dev_{LNS}(\%)$	TT	l_{max}	OV	$dev_{LNS}(\%)$	$1 \rightarrow 2$ drivers
0.999	127.88	3.87	54.01	0.00	124.94	5.49	52.77	0.00	124.94	5.49	52.77	0.00	2.31
0.9	127.97	3.50	49.04	0.01	125.14	3.81	48.00	0.00	125.14	3.81	48.00	0.00	2.13
0.8	128.18	2.98	43.94	0.00	125.55	2.91	43.05	0.00	125.55	2.91	43.05	0.00	2.04
0.7	128.18	2.98	38.82	0.00	125.67	2.76	38.02	0.00	125.67	2.76	38.02	0.00	2.06
0.6	128.92	2.46	33.67	0.02	126.39	2.20	32.93	0.00	126.39	2.20	32.93	0.00	2.19
0.5	129.31	2.24	28.44	0.00	126.68	2.03	27.78	0.03	126.68	2.03	27.78	0.03	2.31
0.4	130.04	1.97	23.16	0.00	129.18	1.06	22.46	0.46	129.18	1.06	22.46	0.37	3.00
0.3	133.05	1.31	17.79	0.17	129.18	1.06	17.11	0.30	129.18	1.06	17.11	0.26	3.81
0.2	135.44	0.97	12.23	0.13	133.84	0.43	11.65	0.53	133.84	0.43	11.65	0.52	4.76
0.1	146.22	0.20	6.36	0.75	138.83	0.08	5.94	1.51	138.83	0.08	5.94	1.58	6.52
0.001	152.70	0.00	0.06	0.75	142.44	0.00	0.06	2.23	142.44	0.00	0.06	2.28	6.72

Table 4.7 Results for data set C_{small} with different importances of consistency.

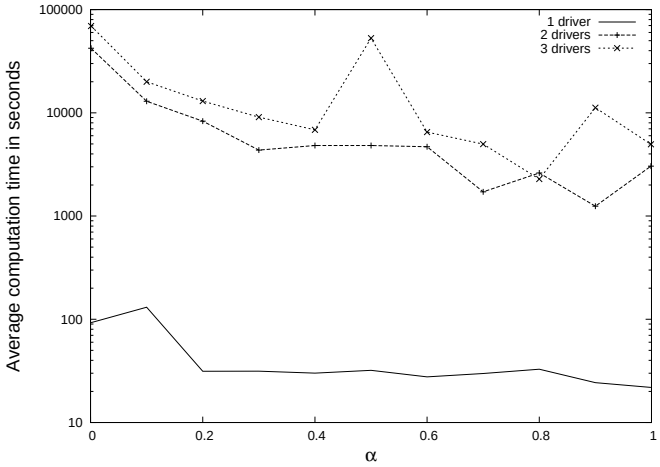


Figure 4.8 Average computation times required by CPLEX.

is at $\alpha = 0.1$ with 130.9 seconds. With two drivers per customer, the average computation time is more than 11 hours at $\alpha = 0.001$. With three drivers per customer there are two peaks: one at $\alpha = 0.001$ and one at $\alpha = 0.5$ with an average computation time of 19.2 hours and 14.7 hours, respectively. CPLEX can cancel a large number of binary variables from the model if W is set to one; therefore, problems can be solved quickly. The computation time increases significantly when W is larger than one. The same observation is made by Francis et al. [66]. The authors enforce driver consistency in order to make the problem easier to solve with the proposed mathematical programming approach.

Results for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$

In this section, we present results for large GenConVRP data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$. The results of all instances in the respective data set are aggregated for reasons of clarity and

comprehensibility. Figures 4.9, 4.12, and 4.15 show the average improvement to the best known solution among all instances in data set $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. Results are presented for different parameters α and for different numbers of allowed drivers per customer. The best known solutions refer to the best results found with the respective setting for parameter α (regardless of the number of drivers per customer). For each α, the first bar represents the result for one driver per customer, the second bar the result for two drivers per customer, and so on.

The pattern in the figures is similar regardless of the service frequency. The LNS results have high variation with low α values. This result is not surprising because LNS is designed with a commercial perspective. Interestingly, with low α, better solutions are obtained if the driver consistency is more restrictive. This observation led to the improvement described in Section 4.3.9.

As α increases, the results are more stable and the advantage of allowing more than one driver per customer becomes apparent. For each service frequency, there is a very small difference between allowing two or more drivers per customer. However, in most of the cases the results improve significantly when two drivers are allowed instead of one. The benefit of allowing more than one driver per customer decreases with larger service frequency. This is because with higher service frequency the daily routes become similar. So, not much can be saved by assigning a customer to different drivers on different days.

Figures 4.10, 4.13, and 4.16 give the average travel time and the maximum arrival time difference for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. The lines show the results with different numbers of allowed drivers per customer. When decreasing α from 0.999 to 0.9, there is a remarkable drop in the maximum arrival time difference. The total travel time, however, barely increases. With decreasing α, the maximum arrival time difference can be reduced further at the expense of a moderate increase in the total travel time. This observation is consistent with findings in Chapter 3 (Kovacs et al. [107]). LNS is able to generate solutions with a maximum arrival time difference of zero if the focus is fully on time consistency ($\alpha = 0.001$). Yet, this comes at the price of a drastic increase in the total travel time.

Figures 4.11, 4.14, and 4.17 show the average fleet (or staff) size (i.e., the average number of vehicles used in the solutions) and the variation in each driver's departure times for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. The variation in the departure times is expressed by the average ratio between the standard deviation of the vehicle departure times and the shift length; we consider only instances in which the shift length is bounded.

The figures show that driver consistency has a minor impact on the average fleet size.

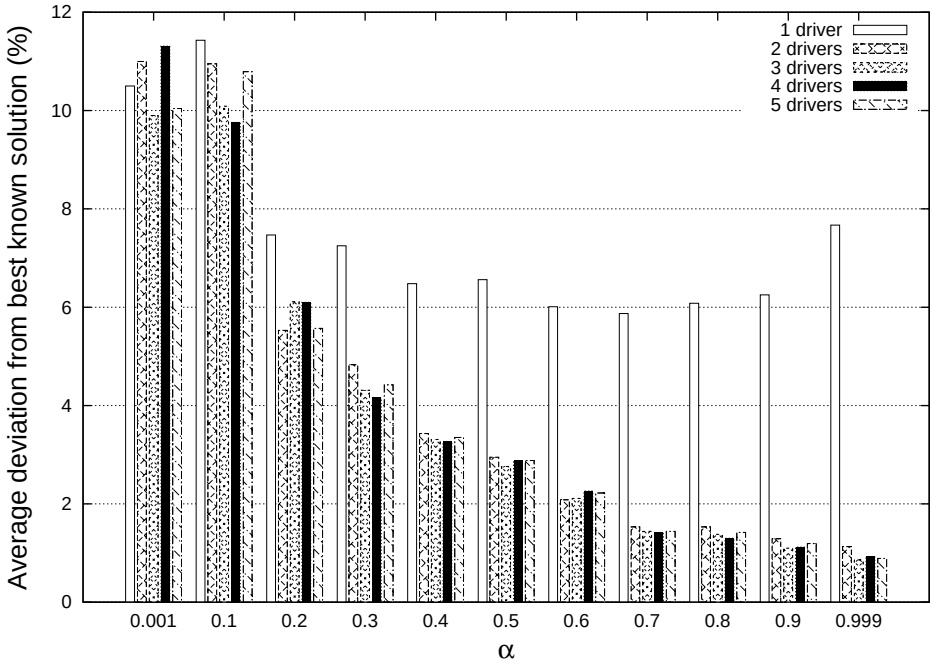


Figure 4.9 Average deviation in the objective value for different numbers of allowed drivers per customer and different importance of the time consistency on data set $C_{0.5}$. The lower α, the more important the time consistency.

For all tested service frequencies and parameters α, the average fleet size is almost the same regardless of the number of allowed drivers per customer. Time consistency can also be improved with a modest increase in the average fleet size. Yet, with $\alpha = 0.001$, an average fleet size that is 2.5 to 6 times larger (depending on the service frequency) than the fleet size without considering consistency is required. The need for a larger fleet explains the sharp increase in the total travel time in Figures 4.10, 4.13, and 4.16. The variation in the departure times is not affected much by the number of different drivers per customer and the emphasis on arrival time consistency. The ratios are roughly 7%, 5%, and 3% for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. These results are similar to the variations found in data sets $B_{0.5}$, $B_{0.7}$, and $B_{0.9}$ indicating that the effect of the time windows on the variation in the departure times is minor. The variations decrease only when $\alpha = 0.001$. Here, the number of routes increases but they become shorter and are, therefore, less affected by fluctuations in the demand.

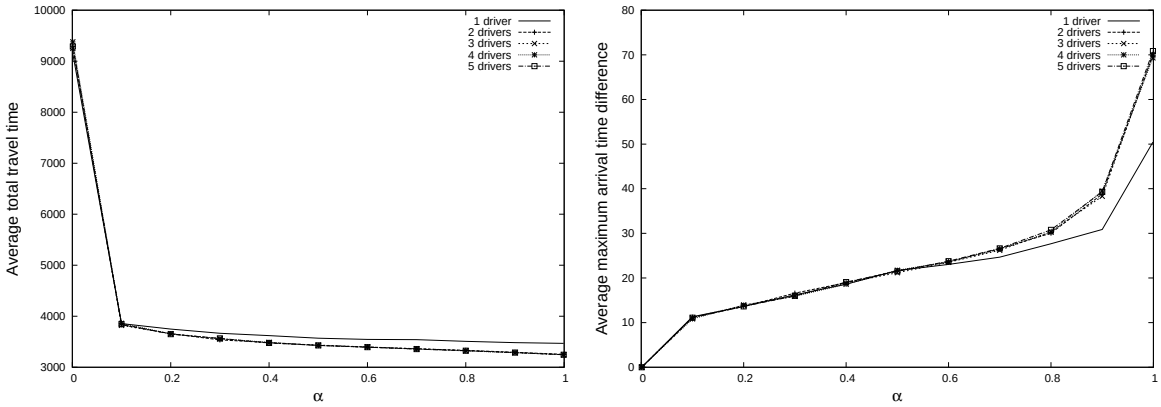


Figure 4.10 Average total travel time and average maximum arrival time difference for different numbers of allowed drivers per customer and different importance of the time consistency on data set $C_{0.5}$. The lower α, the more important the time consistency.

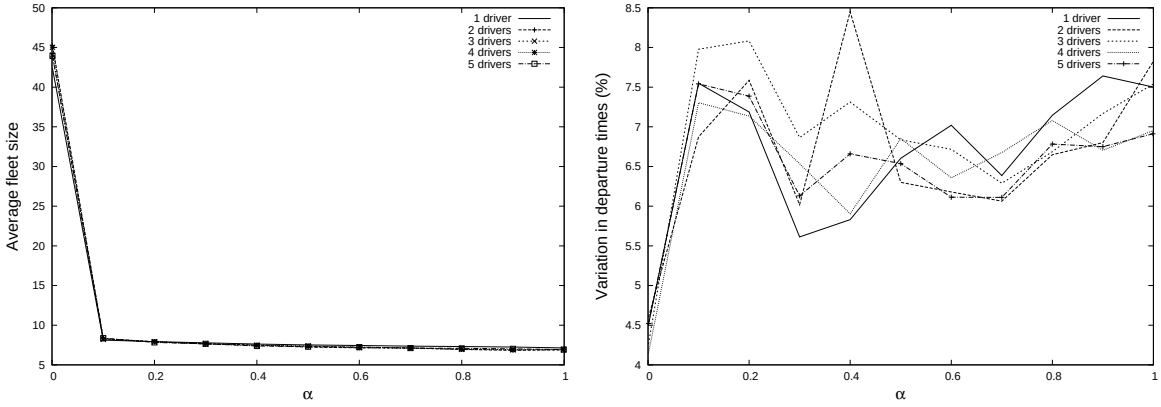


Figure 4.11 Average fleet size and average ratio between the standard deviation of the vehicle departure times and the shift length for data set $C_{0.5}$. The lower α, the more important the time consistency.

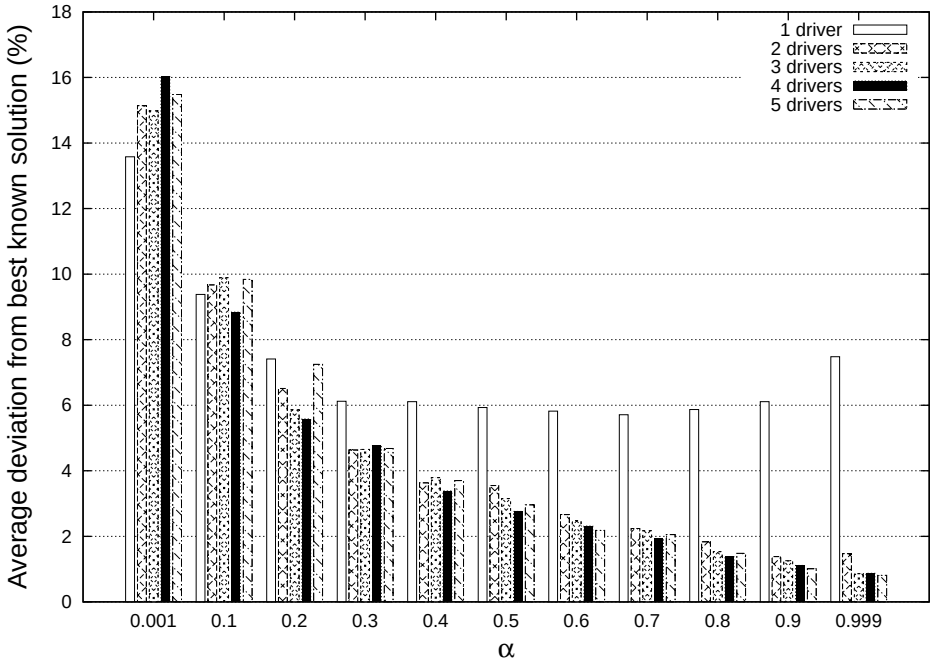


Figure 4.12 Average deviation in the objective value for different numbers of allowed drivers per customer and different importance of the time consistency on data set $C_{0.7}$. The lower α, the more important the time consistency.

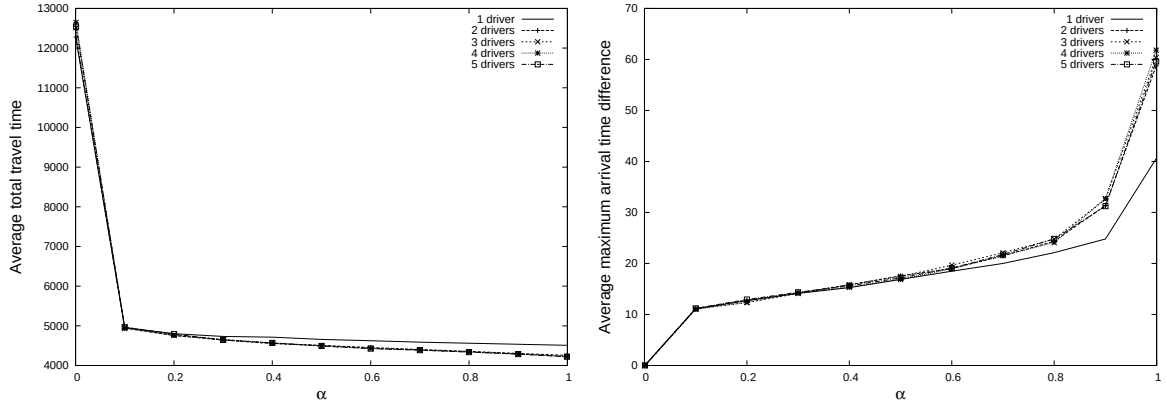


Figure 4.13 Average total travel time and average maximum arrival time difference for different numbers of allowed drivers per customer and different importance of the time consistency on data set $C_{0.7}$. The lower α, the more important the time consistency.

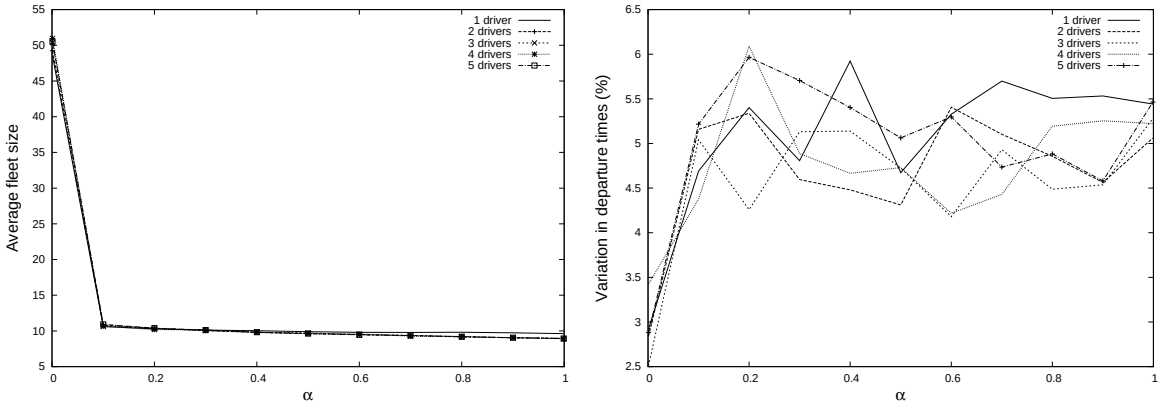


Figure 4.14 Average fleet size and average ratio between the standard deviation of the vehicle departure times and the shift length for data set $C_{0.7}$. The lower α, the more important the time consistency.

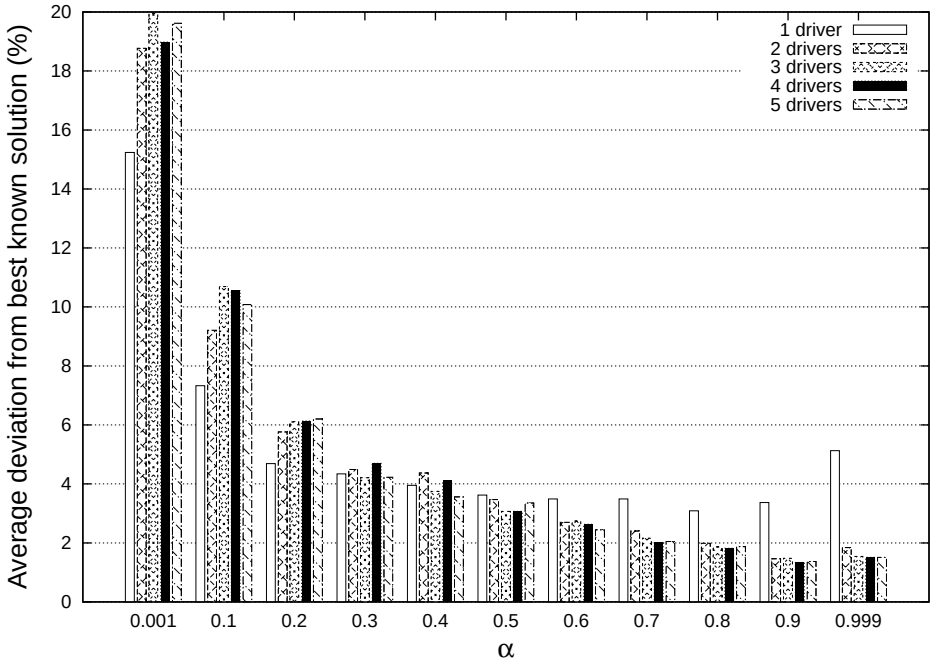


Figure 4.15 Average deviation in the objective value for different numbers of allowed drivers per customer and different importance of the time consistency on data set $C_{0.9}$. The lower α, the more important the time consistency.

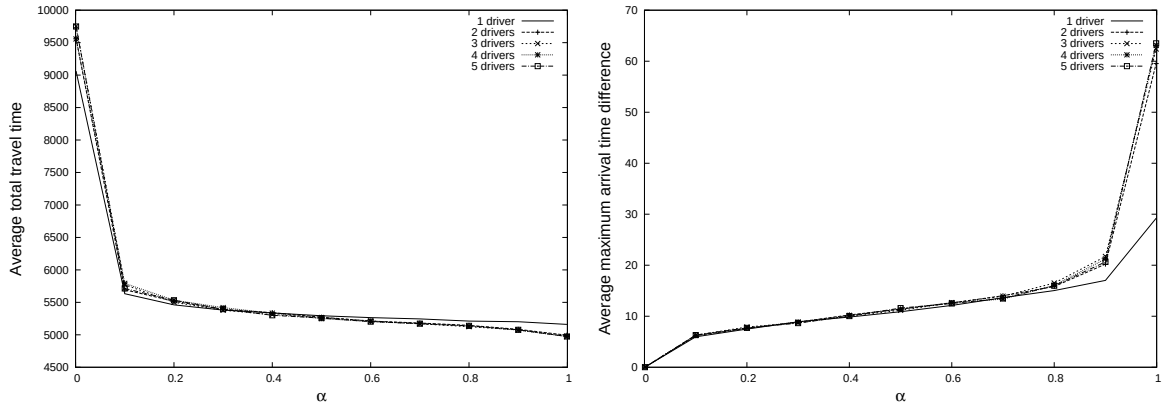


Figure 4.16 Average total travel time and average maximum arrival time difference for different numbers of allowed drivers per customer and different importance of the time consistency on data set $C_{0.9}$. The lower α, the more important the time consistency.

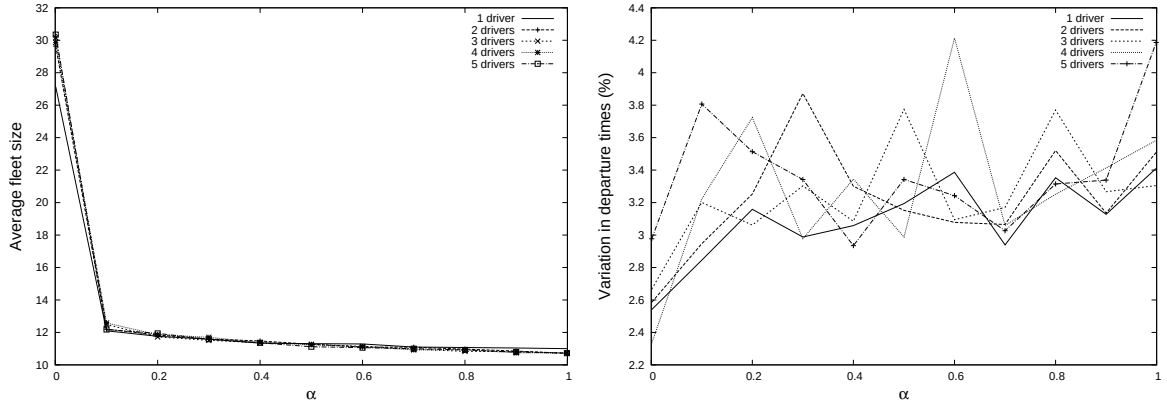


Figure 4.17 Average fleet size and average ratio between the standard deviation of the vehicle departure times and the shift length for data set $C_{0.9}$. The lower α, the more important the time consistency.

Effect of time windows

In this section, we examine the results with regard to the location of AM and PM customers, respectively. We compare three scenarios: First, the scenario described in Section 4.4.1, i.e., AM and PM customers are distributed homogeneously in the service region. In the second scenario, the depot is located in the city center and customers are assigned to time windows in a circular fashion; close business customers are visited in the morning while more remote private customers are visited in the afternoon. Time windows are ignored in the third scenario, i.e., each customer can be visited either in the morning or in the afternoon. Figure 4.18 illustrates the first and second scenario.

In Table 4.8, we present the interaction between parameters α, W, the service frequency, and the time window pattern. The values give the average percentage improvement in the objective value compared to scenario one for different α, W, and service frequency settings; "Circular" refers to scenario two and "No TW" refers to the scenario in which time windows are ignored. The benefit of a circular pattern is between 0.68% and 6.13%. The actual improvement depends on the service frequency and parameter α; the level of driver consistency has a minor impact. Unattractive routes with crossing edges are inevitable when AM and PM customers are distributed homogeneously and α is small, i.e., the emphasis on arrival time consistency is large. However, routing can be performed efficiently when AM and PM customers are assigned in a circular fashion without deteriorating arrival time consistency; the resulting routing plans show a flower-shaped structure. Routing cost outweighs the importance of arrival time consistency as α increases; routes are optimized with regard to cost and detours in order to improve consistency are avoided. Therefore, the benefit of the circular time window assignment diminishes.

Small service frequencies, i.e., large demand fluctuations, make it difficult to plan routes efficiently in terms of cost and achieve high arrival time consistency at the same time. As described above, the circular time window pattern reduces the conflict between arrival time consistency and routing cost. So, the lower the service frequency the higher the improvement of having a circular time window pattern. We assume that exceptions from this statement occur due to variations in the heuristic results.

Ignoring time windows improves the results between 9.83% and 14.62%. The influence of the parameters is similar to the circular time window pattern. However, the impact of α is lower than before. Without time windows, the trivial upper bound for l_{max} increases from $T/2$ to T. Therefore, our algorithm struggles to improve the results when the emphasis is put on arrival time consistency.

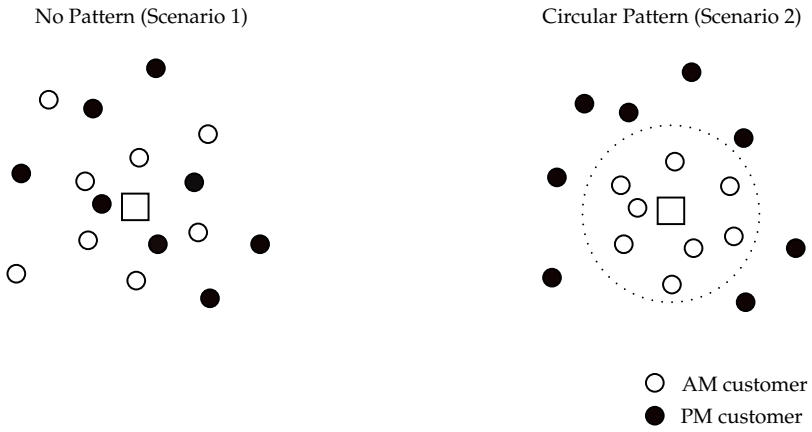


Figure 4.18 Two scenarios: First, there is no correlation between customer location and time window assignment. Second, customers are assigned to time windows in a circular fashion.

4.5 Summary

In this chapter, we presented an extension of the consistent vehicle routing problem (ConVRP). We call it the generalized consistent vehicle routing problem (GenConVRP) as it generalizes the ConVRP by allowing more than one driver per customer, by relaxing the constraint for the maximum arrival time difference, by allowing flexible vehicle departure times, and by incorporating AM/PM time windows for the customers. The GenConVRP is solved by a large neighborhood search algorithm (LNS). Many solution approaches for the ConVRP are based on the template concept in which daily vehicle routes are derived from a set of template routes. Our approach deviates from this principle and generates solutions without using templates. Experiments on different ConVRP variants show that on average the non-template-based LNS performs better than all published algorithms for the ConVRP.

Except for instances where α is very small (which we assume are not interesting in commercial applications), we can summarize that the proposed LNS algorithm performs robustly as it generates high service quality solutions with different input parameters with a minor increase in the variable cost (i.e., travel time) and the fixed cost (i.e., fleet size). The results show that a very strict level of driver consistency with one driver per customer can be obtained without having to increase the fleet size. Yet, with greater flexibility, i.e., by allowing at least two drivers per customer, the objective value can be reduced by up to 6.5% (see Figure 4.9 with $\alpha = 0.999$). The larger the irregularities in the customer demands, the greater the potential savings. A large reduction in the maximum arrival time difference can be obtained with modest increases in travel cost and fleet size. The variation of the vehicle departure times necessary to improve arrival time consistency is slightly affected by

		Data Set					
		$C_{0.5}$		$C_{0.7}$		$C_{0.9}$	
W	α	Circular	No TW	Circular	No TW	Circular	No TW
1	0.5	6.13	14.62	4.02	11.36	3.16	10.66
	0.6	4.84	13.89	2.52	11.29	2.29	10.57
	0.7	4.44	13.38	1.94	11.14	2.53	10.92
	0.8	3.14	13.06	1.12	11.12	1.48	10.29
	0.9	2.71	12.86	0.68	10.89	1.24	10.45
2	0.5	5.71	14.23	3.80	11.31	3.65	10.97
	0.6	4.50	13.08	2.62	10.84	2.67	10.74
	0.7	3.91	13.31	1.96	10.42	2.14	10.53
	0.8	3.44	12.56	1.02	10.09	2.01	10.82
	0.9	2.81	12.18	1.17	9.83	1.28	10.16
3	0.5	5.65	14.02	3.27	11.23	3.04	10.75
	0.6	4.61	13.05	2.52	10.52	2.87	10.72
	0.7	3.85	12.52	2.03	10.58	2.26	10.75
	0.8	3.22	12.41	0.99	9.93	1.99	10.35
	0.9	2.94	12.27	1.23	10.08	1.37	10.45

Table 4.8 Average percentage improvement in the objective value compared to scenario 1 (i.e., time windows are independent of the location of customers). Circular refers to scenario two and No TW means that time windows are ignored.

time windows. Disregarding AM/PM time windows decreases cost by up to 14.62% when demand fluctuations are high and emphasis is put on service consistency.

Chapter 5

The multi-objective generalized consistent vehicle routing problem

5.1 Introduction

Vehicle routing with consistency considerations is a multi-objective process: Companies put large efforts into optimizing vehicle routes in order to decrease cost. At the same time, many companies are willing to increase routing cost to visit each customer at similar times of the day with drivers familiar to them. This is because service consistency increases customer satisfaction and, thus, the lifetime value of customers. Decision makers are faced with the task of choosing between a minimal-cost routing plan that is poorly consistent, a high-cost routing plan with perfect consistency (i.e., visiting each customer at exactly the same time with his favorite driver), or any routing plan in between these extremes.

Several papers examine the trade-off between service consistency and routing cost, but the multi-objective nature of the problem is often oversimplified. In Chapter 3 (Kovacs et al. [107]), for example, we compared the routing cost of solutions in which consistency is ignored to solutions in which consistency is enforced by hard constraints. The same approach is applied in Coelho and Laporte [35], Francis et al. [67], Groër et al. [75], and Spliet [159]. In Chapter 4 (Kovacs et al. [103]), and in Coelho et al. [32], Coelho and Laporte [34], and Smilowitz et al. [158], the multi-objective problem is transformed into a single-objective problem by aggregating routing cost and consistency measurements into a single objective function. This approach involves a priori decisions about the importance of each objective, i.e., service consistency needs to be expressed in monetary gain. Feillet et al. [59] introduce the time-consistent vehicle routing problem with two objectives: minimizing routing cost and improving arrival time consistency.

Multi-objective optimization is a flexible approach for analyzing the trade-off between conflicting objective functions. Without conflicting objectives, the optimization process results in a single solution that is optimal in all objective functions. However, with conflicting objectives, there may exist a large number of relevant solutions. These solutions are not optimal in the sense of single-objective optimization; rather, they are trade-off solutions, i.e., solutions that cannot be improved in one objective without deteriorating another. Among all solutions found, the one that maximizes the utility of the current decision maker will be implemented in the field.

The active interest by researchers and practitioners in multi-objective combinatorial optimization (MOCO) has generated a vast amount of literature. Survey papers are presented, e.g., by Coello Coello [36], Ehrgott and Gandibleux [54], Ehrgott and Wiecek [56], and Steuer et al. [164]. Prominent examples of mathematical programming-based optimizers for multi-objective problems are the ε-constraint method (e.g., Bérubé et al. [14], Ehrgott and Ruzika [55], Haimes et al. [77], Srinivasan and Thompson [163]) and the two-phase method (e.g., Przybylski et al. [138, 139], Visée et al. [181]). Both approaches provide trade-off solutions by repeatedly solving a single-objective problem with changing parameters (e.g., tightness of constraints and coefficients of the objective function). Other algorithms that are based on the iterative scheme are presented, e.g., in Neumayer and Schweigert [125], Ralphs et al. [140], Riera-Ledesma and Salazar-González [143], and Sylva and Crema [168]. Leitner et al. [114] perform a computational study on five iterative solution approaches for the bi-objective prize-collecting Steiner tree problem. Modified branch-and-bound and branch-and-cut algorithms solve the multi-objective problem in a single run (e.g, Jozefowiez et al. [94], Parragh and Tricoire [134], Vincent et al. [180]).

Heuristic multi-objective optimizers are often based on evolutionary algorithms (EAs) (Jones et al. [93]). EAs maintain a pool of solutions (called population) during the search process. Therefore, EAs are particularly suited for finding many trade-off solutions in a single run. An overview of evolutionary solution approaches is presented, e.g., in Coello Coello [36], Fonseca [63], Horn [90], Konak et al. [101], Tamaki et al. [169], Van Veldhuizen [177], Zitzler et al. [191], and Zitzler and Thiele [192]. Multi-objective extensions of single-objective heuristics such as tabu search, variable neighborhood search, ant colony optimization, and simulated annealing are presented, e.g., in Czyżak and Jaszkiewicz [41], Doerner et al. [49], Hapke et al. [78], Jozefowiez et al. [95], Parragh et al. [133], Schilde et al. [150], and Yang et al. [189]. Paquete et al. [130] and Tricoire [174] present heuristic frameworks for MOCO problems that integrate various local search strategies.

In this chapter, we present the multi-objective generalized consistent vehicle routing

problem (MOGenConVRP). The problem is based on the generalized consistent vehicle routing problem (GenConVRP, see Chapter 4 and Kovacs et al. [103]) that aggregates routing cost and arrival time consistency into a single objective function; the number of different drivers per customer is bounded. In the MOGenConVRP, routing cost, arrival time consistency, and driver consistency are independent objectives of the problem. Solving this multi-objective problem in practice is unrealistic: Typically, the long computation time caused by the large search space and the high number of relevant solutions is prohibitive. The aim of our work is to find all trade-off solutions for different test instances; these solution sets enable a thorough analysis of the trade-off between arrival time consistency, driver consistency, and routing cost. Our results should help companies in the routing industry in finding adequate consistency goals to aim for. We devise two exact solution approaches and one heuristic. The exact approaches are based on the ε-constraint method (Haimes et al. [77], Laumanns et al. [112], Srinivasan and Thompson [163]). For both exact approaches, the computation time is reduced by introducing valid inequalities and by exploiting special properties of the problem. Optimal solutions for the MOGenConVRP are important for benchmarking tests that measure the quality of approximation methods. Our heuristic is a combination of the large neighborhood search algorithm (LNS) for the GenConVRP (Chapter 4, Kovacs et al. [103]) and the multi directional local search framework (MDLS, Tricoire [174]). We refer to the heuristic algorithm as multi directional large neighborhood search (MDLNS). The quality of the MDLNS is validated by computing several unary quality measurements for multi-objective solution sets. The computation time of the algorithms is secondary for our analysis. Nevertheless, for practical applications, all algorithms can be speeded up at the cost of solution quality.

5.2 Problem definition

The MOGenConVRP is based on the GenConVRP (see Chapter 4 and Kovacs et al. [103]) and is modeled on a complete directed graph $G = (N^0, A)$. $N^0 = \{0, 1, ..., n\}$ is the set of nodes representing customer locations and the depot 0, N is the set of customers ($N^0 \backslash \{0\}$). Customers are divided into AM customers who can only be visited in the morning ($N^{am} \subseteq N$) and PM customers who require a visit in the afternoon ($N^{pm} \subseteq N, N^{am} \cap N^{pm} = \{\}, N^{am} \cup N^{pm} = N$). Set N^f contains all customers that require at least two visits during the planning period. $A = \{(i, j) \ : \ i, j \in N^0, i \neq j\}$ is the set of arcs. Arcs from customers $\in N^{pm}$ to customers $\in N^{am}$ are omitted from the graph. Each arc $(i, j) \in A$ is associated with travel time t_{ij}. The travel time matrix satisfies the triangle inequality. Customers are visited by

a homogeneous fleet of vehicles in the set $K = \{0, 1, ..., k\}$. The number of vehicles is not restrictive (i.e., $k = n$). Each vehicle has a capacity of Q. Drivers and vehicles share a one-to-one relationship. We use drivers and vehicles interchangeably, depending on the context. Each route starts and ends at the depot. The departure time is flexible, but vehicles must return to the depot before time T. The planning horizon involves $|D|$ days where D is the set of days. On each day $d \in D$, each customer $i \in N$ has demand q_{id} and service time s_{id}. Auxiliary parameters w_{id} are set to 1 if customer i requires service on day d ($q_{id} > 0$) and set to 0, otherwise. Each set of nodes is specified by index d to denote customers that require service on day d (e.g., N_d^0 contains customers that require service on day d and the depot). Additionally, we define set $A_d \subset A$ that contains arcs between customers in N_d; A_d^0 also contains the arcs from and to the depot. The model uses the following binary variables:

$$x_{ijkd} = \begin{cases} 1, \text{if arc } (i,j) \in A_d^0 \text{ is traversed by vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

$$y_{ikd} = \begin{cases} 1, \text{if customer } i \text{ is assigned to vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

$$z_{ik} = \begin{cases} 1, \text{if customer } i \text{ is assigned to vehicle } k, \\ 0, \text{otherwise.} \end{cases}$$

Variables a_{id} denote the arrival time at customer $i \in N_d$ on day d; z_{max} is the maximum number of different drivers that any customer encounters; l_{max} gives the maximum arrival time difference, i.e., largest difference between the latest and the earliest arrival time per customer among all customers. Constraints (5.5) - (5.13) in the MOGenConVRP model are equivalent to constraints (4.2) - (4.10) in the GenConVRP model (Chapter 4), respectively. Constraints (4.11) - (4.14) in the GenConVRP are combined into constraints (5.14) and (5.15).

$$min\ f = (f_1, f_2, f_3) \tag{5.1}$$

$$f_1 = \sum_{d \in D} \sum_{k \in K} \sum_{(i,j) \in A_d^0} t_{ij} x_{ijkd} \tag{5.2}$$

$$f_2 = z_{max} \tag{5.3}$$

$$f_3 = l_{max} \tag{5.4}$$

subject to:

$$y_{0kd} = 1 \qquad \forall\, k \in K, d \in D \tag{5.5}$$

$$\sum_{k \in K} y_{ikd} = 1 \qquad \forall\, i \in N_d, d \in D \tag{5.6}$$

$$\sum_{i \in N_d} q_{id} y_{ikd} \leq Q \qquad \forall\, k \in K, d \in D \tag{5.7}$$

$$\sum_{i \in N_d^0} x_{ijkd} = \sum_{i \in N_d^0} x_{jikd} = y_{jkd} \qquad \forall\, j \in N_d^0, k \in K, d \in D \tag{5.8}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \leq a_{jd} \qquad \forall\, (i,j) \in A_d, k \in K, d \in D \tag{5.9}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \geq a_{jd} \qquad \forall\, (i,j) \in A_d, k \in K, d \in D \tag{5.10}$$

$$z_{ik} \geq y_{ikd} \qquad \forall\, i \in N_d^f, k \in K, d \in D \tag{5.11}$$

$$\sum_{k \in K} z_{ik} \leq z_{max} \qquad \forall\, i \in N^f \tag{5.12}$$

$$(a_{i\alpha} - a_{i\beta}) w_{i\alpha} w_{i\beta} \leq l_{max} \qquad \forall\, i \in N^f, \alpha, \beta \in D \tag{5.13}$$

$$[t_{0i}, min(\frac{T}{2}, T - t_{i0} - s_{id})] \ni a_{id} \qquad \forall\, i \in N_d^{am}, d \in D \tag{5.14}$$

$$[max(t_{0i}, \frac{T}{2}), T - t_{i0} - s_{id})] \ni a_{id} \qquad \forall\, i \in N_d^{pm}, d \in D \tag{5.15}$$

$$x_{ijkd} \in \{0,1\} \qquad \forall\, (i,j) \in A_d^0, k \in K, d \in D \tag{5.16}$$

$$y_{ikd} \in \{0,1\} \qquad \forall\, i \in N_d^0, k \in K, d \in D \tag{5.17}$$

$$z_{ik} \in \{0,1\} \qquad \forall\, i \in N^f, k \in K \tag{5.18}$$

Equalities (5.5) ensure that each route starts from the depot. Constraints (5.6) guarantee that each customer is serviced on each day he requires service. Inequalities (5.7) limit the vehicle capacity to Q. Equalities (5.8) are flow conservation constraints. Inequalities (5.9) and (5.10) set the arrival times at the customers. Vehicle idling to improve time consistency is not allowed (5.10). Inequalities (5.9) also prevent sub-tours. Driver consistency is defined in (5.11) and (5.12). Constraints (5.11) set the z variables and constraints (5.12) set the maximum number of drivers per customer. Constraints (5.13) define the maximum arrival time difference. Finally, constraints (5.14)-(5.18) define the domains of the decision variables. Time window feasibility is ensured by restricting the domains of the arrival time variables; this approach reduces the computation time of the applied integer linear programming optimizer.

The objective (5.1) is to minimize a vector of conflicting objective functions. The vector f is composed of the total travel time (5.2), the maximum number of different drivers

per customer (5.3), and the maximum arrival time difference (5.4). In the MOGenConVRP, there is no solution that optimizes all three objectives simultaneously. Rather, there are several solutions that are optimal in the sense of multi-objective optimization.

With multiple objectives of equal rank, we cannot decide whether or not a solution is better than another for any pair of solutions, i.e., the multi-objective search space is partially ordered. Comparable solutions are ordered by a dominance relation defined as follows:

Solution s^1 dominates solution s^2 ($s^1 \succ s^2$) if and only if

$$\forall\, i \in \{1,2,3\}:\ f_i(s^1) \leq f_i(s^2) \text{ and}$$
$$\exists\, i \in \{1,2,3\}:\ f_i(s^1) < f_i(s^2).$$

A solution s is efficient if and only if there is no other solution s' that dominates s. Set E contains all efficient solutions, $E = \{s\ :\ \nexists\, s' \text{ such that } s' \succ s\}$. (Note: solutions in E are incomparable and equally good for a neutral decision maker.) The Pareto-front is defined as $P = \{f(s) : s \in E\}$. Elements of P are said to be non-dominated points in the objective space. Depending on the context, we use solution either to denote an element of E or an element of P.

Our goal is to generate P in order to study the trade-off between travel cost and service consistency. There are at least as many efficient solutions as non-dominated points, i.e., several solutions might give the same objective vector. For us, it suffices to find only one solution for each point in P.

5.3 Solution approaches

We propose two exact solution approaches (i.e., approaches that find all points in P) based on the ε-constraint method. The ε-constraint method has been applied successfully for solving various multi-objective problems (e.g., Bérubé et al. [14], Kirlik and Sayın [98], Laumanns et al. [112], Srinivasan and Thompson [163]). Leitner et al. [114] show that the ε-constraint method outperforms other mathematical programming-based solution approaches for the bi-objective prize-collecting Steiner tree problem. In Section 5.4, we show that the exact algorithms are able to solve MOGenConVRP instances with up to 10 customers that require service over five days. So, in order to study realistic problem instances, we also devise a metaheuristic algorithm that provides an approximation of P, denoted by P_{approx}. MDLNS integrates the large neighborhood search algorithm for the GenConVRP (Chapter 4, Kovacs et al. [103]) into the multi directional local search framework (Tricoire [174]). The LNS

is the state-of-the-art approach for single-objective consistent vehicle routing problems; the performance of the MDLS is comparable to the best known solution approaches for three different combinatorial optimization problems. In this section, we describe the algorithms in detail and present problem-specific enhancements.

5.3.1 Exact approaches

In the ε-constraint method, the multi-objective problem is transformed into a single-objective problem by optimizing one objective function and restricting the objective value of the remaining functions. By modifying the tightness of the constraints, we can identify different elements of the Pareto-front. The single-objective problem is solved by a procedure that we refer to as $opt(f, \varepsilon, \varepsilon')$; it is defined as follows (Laumanns et al. [112]):

$$lex\ min\ f(s) \quad = (f_1, f_2, f_3) \tag{5.19}$$

subject to:

$$\varepsilon_i \leq f_i(s) < \varepsilon_i' \qquad \forall i \in \{2,3\} \tag{5.20}$$

$$s \in S \tag{5.21}$$

The objective is to minimize the three objective functions in lexicographic order (5.19): f_1 first, f_2 second, and f_3 third. Constraints (5.20) are ε-constraints that bound the objective value of f_2 and f_3. Set S contains all feasible solutions defined by constraints (5.5) - (5.18). Expression (5.21) restricts the search space to solutions from S.

Without lexicographic optimization, the procedure might return solutions that are not efficient. For example, a solution s^1 with minimal travel cost and objective vector $f(s^1) = (100, 3, 21)$ is dominated by solution s^2 with $f(s^2) = (100, 2, 20)$. The computation time for finding s^1 will be wasted, if s^1 and s^2 are both within the same search space. By performing a lexicographic optimization, we can guarantee that the provided solution is always non-dominated for the given search space. The output of the procedure is either an optimal solution for the constrained problem (if a feasible solution exists) or null (if the search space is empty).

In our implementation, $opt(f, \varepsilon, \varepsilon')$ is based on a branch-and-bound algorithm. The lexicographic optimization is performed in two phases: The single-objective problem with f_1 as the only objective is solved in the first phase; let s^1 be the provided solution. In the

second phase, we restrict the f_1 value of the second solution s^2 to the previously obtained value ($f_1(s^2) \leq f_1(s^1)$) and solve the problem with the following objective function:

$$min\ f_2(s^2) + \frac{f_3(s^2)}{UB_3}. \tag{5.22}$$

Parameter UB_3 is an upper bound for f_3 such that $UB_3 > f_3$. (A trivial upper bound for f_3 is half the closure time at the depot $T/2$.) This approach optimizes f_2 and f_3 in lexicographic order since $f_2 \in \mathbb{N}$ and $\frac{f_3(s)}{UB_3} < 1$. Furthermore, the second phase can be solved with minimal effort: Once the first phase solution is found, the second phase can be solved by exploring the unprocessed nodes of the branch-and-bound tree; all other nodes have already been pruned either by infeasibility or by the f_1-bound. The resulting solution is the correct output of $opt(f, \varepsilon, \varepsilon')$.

We propose two algorithms that repeatedly apply $opt(f, \varepsilon, \varepsilon')$ in order to obtain P. The algorithms differ in the scheme the ε-constraints (5.20) are modified. The total computation time of both algorithms depends mainly on the computation time of $opt(f, \varepsilon, \varepsilon')$. The optimization process can be speeded up by aborting the procedure when a certain optimality gap (i.e., difference between the best found solution and the best lower bound) is reached. Yet, this might prevent us from generating the optimal Pareto-front.

Two dimensional adaptive ε-constraint method

Our first exact multi-objective algorithm is called two dimensional (2D) adaptive ε-constraint method. The 2D method is based on the observation that the number of different drivers per customer (f_2) is, typically, low (in our experiments no more than five). Therefore, we can enumerate the bi-objective problem with objectives f_1 and f_3 for all possible bounds on f_2. For any f_2 value, the bi-objective problem is solved by repeatedly executing $opt(f, \varepsilon, \varepsilon')$ with modified ε-constraints.

The outline of the 2D method is given in Algorithm 5. The procedure starts with an empty set of efficient solutions E. In line 1 and 2, we initialize the ε-constraints on f_2 and f_3. In each iteration of the outer loop (line 3-12), the bound on f_2 is tightened by at least one (line 11); variable k memorizes the maximum number of drivers per customer among all solutions for the respective bi-objective problem. For each upper bound on f_2, the inner loop (line 5-10) provides efficient solutions for the bi-objective problem by repeatedly tightening the bound on f_3 (line 8). The algorithm stops as soon as the bi-objective problem is solved for the strictest driver consistency ($f_2 \leq 1$). The output is a set of solutions that contains at least one solution for each point on the Pareto-front.

Algorithm 5 *2D ε − constraint method*

Require: $E = \{\}$
1: $\varepsilon = (0,1)$
2: $\varepsilon' = (\infty,\infty)$
3: **while** $\varepsilon'_2 > \varepsilon_2$ **do**
4: $k = 0$
5: **while** $\varepsilon'_3 > \varepsilon_3$ **do**
6: $s = opt(f,\varepsilon,\varepsilon')$
7: $E = E \cup \{s\}$
8: $\varepsilon'_3 = f_3(s)$
9: $k = max\{k, f_2(s)\}$
10: **end while**
11: $\varepsilon' = (\infty,k)$
12: **end while**
13: **return** E

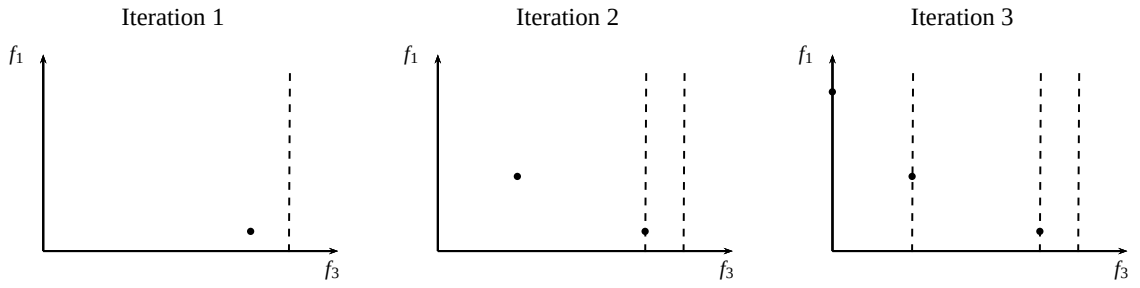


Figure 5.1 Objective space for three iterations of the inner loop (line 5-10) of Algorithm 5, i.e., the bi-objective problem for an arbitrary bound on f_2.

Figure 5.1 illustrates the objective space for three iterations of the inner loop (line 5-10), respectively. The dashed lines denote ε-constraints on f_3 and the dots represent the objective vector of solutions found by $opt(f,\varepsilon,\varepsilon')$. The left figure shows the initial bound on f_3 and the first solution with minimal f_1 value. In the second iteration, the bound on f_3 (i.e., ε'_3) is set to the f_3 value of the solution found previously and the constrained problem is solved. The figure at the right shows a solution with $f_3 = 0$. The inner loop stops because f_3 is optimal. The algorithm proceeds by tightening the bound on f_2 and solving the bi-objective problem again.

Three dimensional adaptive ε-constraint method

The second exact algorithm is referred to as three dimensional (3D) adaptive ε-constraint method. It is based on the framework proposed by Laumanns et al. [112]. The authors present a general algorithm for solving multi-objective problems with an arbitrary number of objectives, m. The worst case time complexity of the algorithm, measured by the calls of $opt(f,\varepsilon,\varepsilon')$, depends on the number of points in P ($|P|$) and the number of objectives: $\mathcal{O}(|P|^{m-1})$.

The basic concept is an $m-1$ dimensional hypergrid that divides the objective space into rectangular subspaces that are parallel to the axes, e.g., into strips in bi-objective problems (see figure at the right of Figure 5.1), (Laumanns et al. [112]). Each cell in the grid represents one constrained search space for which $opt(f,\varepsilon,\varepsilon')$ is applied. The coordinates of the grid are given by solutions that have been found earlier; so, the number of cells in the grid grows to the power of $m-1$ each time a new solution is found. In our case, the objective space is divided with respect to f_2 and f_3. For each efficient solution found during the search, we store the objective values f_2 and f_3 in vectors e_2 and e_3 ($e=(e_2,e_3)$), respectively. Matrix e defines the coordinates of the grid.

The pseudo-code of the algorithm is given in Algorithm 6. The procedure is initialized with an empty set of efficient solutions E, an empty set of already searched subspaces F, and the initial coordinates of the grid e. Variable i is an iteration counter that indicates the index of the cell that is to be searched and variable c is the current number of cells in the grid. Initially, the grid contains one cell that is defined by the lower and upper bounds on f_2 and f_3, respectively. In each iteration of the outer loop (line 3-24), the algorithm provides one new solution. The inner loop (line 4-18) examines each cell for new efficient solutions; in each iteration, the subspaces are explored in increasing number of cell index starting with index 0. The current subspace is provided by function $getConstraints(i,e,E)$ that translates the current cell into ε-constraints (line 8) (details on this function are described in Laumanns et al. [112]). If the current search space has not yet been examined (line 9), we apply $opt(f,\varepsilon,\varepsilon')$. Depending on the outcome, we distinguish two paths: First, $opt(f,\varepsilon,\varepsilon')$ provides a solution that is dominated by a solution in E or there is no feasible solution in the respective cell. Then, we mark the current subspace defined by ε and ε' as searched (line 12) and move on to the next cell (line 17). Second, $opt(f,\varepsilon,\varepsilon')$ provides a new efficient solution. In this case, we leave the inner loop (line 14), add the solution to E (line 19), and mark the subspace that is dominated by the new solution ($[f(s),\varepsilon']$) as searched (line 20). In line 21, we call the function $updateConstraints(f(s),e)$ that inserts the new coordinates of the grid into matrix e (for details on this function see Laumanns et al. [112]). The number

of cells grows with each solution found; c is updated in line 22. The algorithm generates one solution for each point in P, i.e., $|E| = |P|$. This is guaranteed by the definition of the subspaces (inequalities (5.20)) and the lexicographic optimization within each subspace. Finally, we reset the iteration counter to zero in order to start the search in the inner loop with cell 0. The algorithm stops when all cells have been examined (line 5-7); the output is a set of efficient solutions E.

Figure 5.2 illustrates the $f_2 - f_3$ projection of the objective space for three iterations of the outer loop of the 3D method, respectively. For each iteration, the figure at the left shows the initial grid and the figure at the right shows the grid after an efficient solution has been found. In the first iteration, the grid consists of a single cell that contains an efficient solution denoted by a point. The new solution divides the grid into four cells. The numbering of the cells starts with 0 in the lower left corner and increases successively column by column. In the second iteration, we continue the search in cell 0. Cell 3 is hatched because it represents a subspace that is dominated by the solution found in the first iteration. We find a new efficient solution in cell 0. The grid is partitioned and the cells are renumbered. In iteration 3, we again start the search with cell 0. Cell 4 is hatched because it is dominated by the solution found in iteration 2. (Cells 5 and 7 are not dominated because they might contain solutions with a lower f_1 value.) The search in cell 0 is without result, i.e., $opt(f, \varepsilon, \varepsilon')$ returns null. So, cell 0 is marked as searched and we continue with cell 1. Here, we find a new solution, partition the grid, and mark cell 6 as searched.

It is essential to search the cells in ascending order of indices starting with cell 0: Let's assume that in iteration 2 there is a solution s^2 with objective vector $(100, 2, 12)$ in cell 2 and a solution s^1 with objective vector $(100, 2, 10)$ in cell 0. If we started with cell 2, we would partition the grid based on s^2 even though s^1 dominates s^2.

Single objective optimizer

The computation time of our exact approaches is mainly affected by the lexicographic optimizer $opt(f, \varepsilon, \varepsilon')$. Therefore, we put emphasis on strengthening the model in a branch-and-cut fashion. In the following, we describe and evaluate the efficiency of five types of valid inequalities: Capacity cuts and subtour cuts are general inequalities for routing problems (see, e.g., Laporte and Nobert [111], Naddef and Rinaldi [123], Toth and Vigo [173]). Symmetry breaking constraints have been investigated in Coelho and Laporte [35] and Fischetti et al. [60]; we present a modification of these constraints that is valid for the MOGenConVRP. Finally, two time consistency related inequalities are introduced.

Algorithm 6 *3D ε − constraint method*

Require: $E = \{\}, F = \{\}, e_2 = (0, \infty), e_3 = (1, \infty)$

$i = 0$
$c = 1$
loop
  loop
    if $i \geq c$ **then**
      return E
    end if
    $(\varepsilon, \varepsilon') = getConstraints(i, e, E)$
    if $[\varepsilon, \varepsilon'] \not\subset F$ **then**
      $s = opt(f, \varepsilon, \varepsilon')$
      if $s = null$ **or** $\exists s' \in E : s' \succ s$ **then**
        $F = F \cup [\varepsilon, \varepsilon']$
      else
        break
      end if
    end if
    $i = i + 1$
  end loop
  $E = E \cup \{s\}$
  $F = F \cup [f(s), \varepsilon']$
  $updateConstraints(f(s), e)$
  $c = (|E| + 1)^2$
  $i = 0$
end loop

Capacity cuts (CC) and subtour cuts (ST-$|S|$) The model defined by inequalities (5.5) - (5.18) is complete. Nevertheless, adding a limited number of rounded capacity constraints (inequalities (5.23)) and generalized subtour elimination constraints (inequalities (5.24)) can speed up the optimization process.

$$\sum_{i \notin S} \sum_{j \in S} x_{ijkd} \geq \lceil \frac{\sum_{i \in S} q_{id}}{Q} \rceil \qquad \forall S \subseteq N_d, |S| \geq 2, d \in D, k \in K, \tag{5.23}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijkd} \leq |S| - 1 \qquad \forall S \subseteq N_d, |S| \geq 2, d \in D, k \in K. \tag{5.24}$$

Symmetry breaking (SB) Drivers are homogeneous in the MOGenConVRP; therefore, the number of feasible driver-customer assignments grows exponentially with the number of customers. If $z_{max} \leq 1$, there are k^n feasible assignments; k is the number of available drivers and n is the number of customers. Table 5.1 shows all 27 feasible assignments

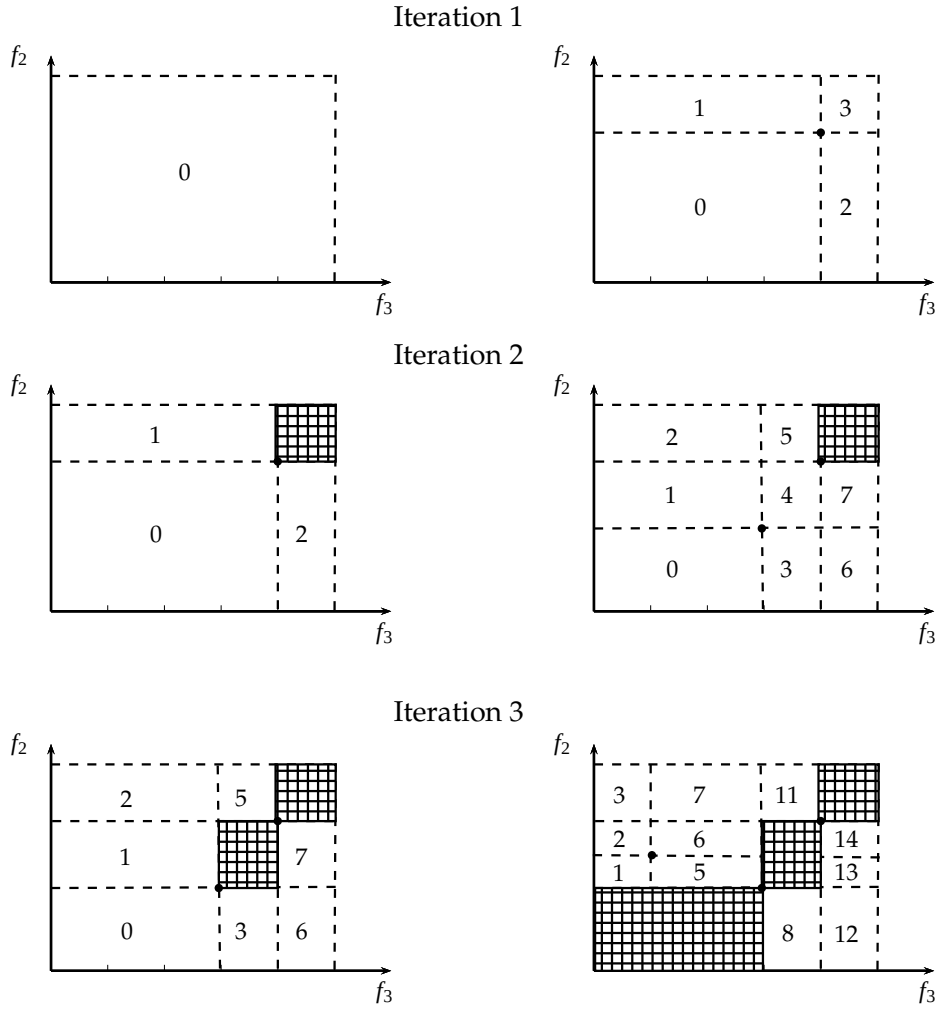


Figure 5.2 $f_2 - f_3$ projection of the objective space for three iterations of Algorithm 6, respectively.

for three drives $\{A,B,C\}$ and three customers $\{1,2,3\}$. Many assignments are equivalent; for example, each assignment from 1 to 6 would yield the same point on the Pareto-front. Motivated by Coelho and Laporte [35] and Fischetti et al. [60], we use symmetry breaking inequalities that reduce the number of feasible assignments significantly:

$$z_{ik} \leq \sum_{j<i} z_{j,k-1} \qquad \forall\, k \in K\backslash\{0\}, i \in N_f, \tag{5.25}$$

$$z_{ik} \leq \sum_{d\in D} y_{ikd} \qquad \forall\, k \in K, i \in N_f. \tag{5.26}$$

Assignment	A	B	C	Assignment	A	B	C
1	1	2	3	16	1,2	3	
2	1	3	2	17	1,2		3
3	3	1	2	18	3	1,2	
4	3	2	1	19	3		1,2
5	2	1	3	20		1,2	3
6	2	3	1	21		3	1,2
7	1	2,3		22	1,3	2	
8	1		2,3	23	1,3		2
9	2,3	1		24		1,3	2
10	2,3		1	25		2	1,3
11		2,3	1	26	2	1,3	
12		1	2,3	27	2		1,3
13	1,2,3						
14		1,2,3					
15			1,2,3				

Table 5.1 Feasible driver-customer assignments if $z_{max} \leq 1$. The set of drivers is $\{A,B,C\}$ and the set of customers is $\{1,2,3\}$.

Inequalities (5.26) restrict the z_{ik} variables; inequalities (5.25) allow a customer-driver assignment (i,k) only if driver $k-1$ visits a customer with a smaller index than i. In our example, the inequalities reduce the number of feasible assignments from 27 to 5, i.e., to assignments 1, 7, 13, 16, and 22.

Inequalities (5.25) are invalid if $z_{max} \geq 2$. In this case, the number of feasible driver-customer assignments is $\prod_{i=0}^{z_{max}-1}(k-i)^n$ because each customer can be assigned to different drivers on different days. A slight modification of inequalities (5.25) is valid regardless of the number of different drivers per customer:

$$z_{ik} \leq \sum_{j \leq i} z_{j,k-1} \qquad \forall\, k \in K \backslash \{0\}, i \in N_f. \tag{5.27}$$

Proposition: Inequalities (5.27) are valid for the MOGenConVRP.
Proof: Let $N_k \in \mathscr{P}(N)$ be the set of customers assigned to driver $k \in K$ ($\mathscr{P}(N)$ is the power set of all customers) and i_k be the customer with smallest index in set N_k. By sorting the i_k's in non-descending order, we achieve either $i_{k-1} < i_k$ or $i_{k-1} = i_k \; \forall\, k \in K \backslash \{0\}$. □

Time consistency-related inequalities 1 (TC1) For a given bound on l_{max} (that is imposed by the ε-constraint method), L, we can add inequalities to the model that exclude solutions with poor arrival time consistency. Time consistency-related inequalities TC1 (5.28) state that if customer i is visited before customer j on the same route on one day,

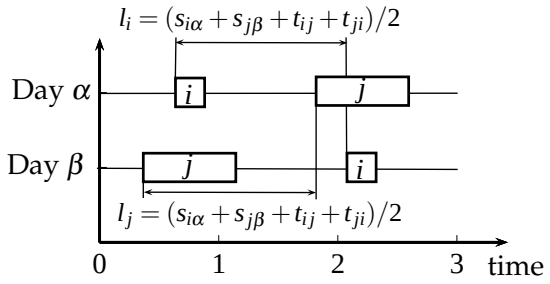


Figure 5.3 Time consistency-related inequalities (TC1).

then it is infeasible to assign customer j before customer i on the same route on an another day if the arrival time consistency constraint would be violated. The reasoning behind these inequalities is illustrated in Figure 5.3. The arrival time difference at customer i (l_i) is $s_{j\beta} + t_{ji}$ plus the departure time from the depot $a_{0\alpha}$[1]; the arrival time difference at customer j (l_j) is $s_{i\alpha} + t_{ij} - a_{0\alpha}$. In an optimal solution l_i is equal to l_j; otherwise, the maximum arrival time difference could be reduced by leveling the arrival time differences at the two customers. So, $s_{j\beta} + t_{ji} + a_{0\alpha} = s_{i\alpha} + t_{ij} - a_{0\alpha}$. This leads to $a_{0\alpha} = (s_{i\alpha} - s_{j\beta} + t_{ij} - t_{ji})/2$ and, therefore, to $l_i = l_j = (s_{i\alpha} + s_{j\beta} + t_{ij} + t_{ji})/2$.

$$x_{ij\delta\alpha} + x_{ji\gamma\beta} \leq 1 \quad \forall\, (i,j) \in A_d^0 : (s_{i\alpha} + s_{j\beta} + t_{ij} + t_{ji})/2 > L, \alpha, \beta \in D, \delta, \gamma \in K. \quad (5.28)$$

Time consistency-related inequalities 2 (TC2) Time consistency-related inequalities TC2 (5.29) prevent the assignment of a customer l between two customers i and j if i and j are scheduled one after the other on another day and if the assignment of l would violate the constraint on the maximum arrival time difference. Figure 5.4 illustrates the reasoning: The arrival time difference at customer i is $l_i = a_{0\alpha}$; the arrival time difference at customer j is $l_j = t_{il} + t_{lj} + s_{l\beta} - t_{ij} - a_{0\alpha}$. Again, we set $l_i = l_j$, i.e., $2a_{0\alpha} = t_{il} + t_{lj} + s_{l\beta} - t_{ij}$. Finally, we obtain $l_i = l_j = (t_{il} + t_{lj} + s_{l\beta} - t_{ij})/2$.

$$x_{ij\delta\alpha} + x_{il\gamma\beta} + x_{lj\gamma\beta} \leq 2 \quad \forall\, (i,j),(i,l),(l,j) \in A_d^0 : (t_{il} + s_{l\beta} + t_{lj} - t_{ij})/2 > L, \quad \alpha, \beta \in D, \delta, \gamma \in K. \quad (5.29)$$

[1] We consider only two routes on two different days α and β. Therefore, we can assume that the vehicle departure time is fixed on day β and flexible on day α.

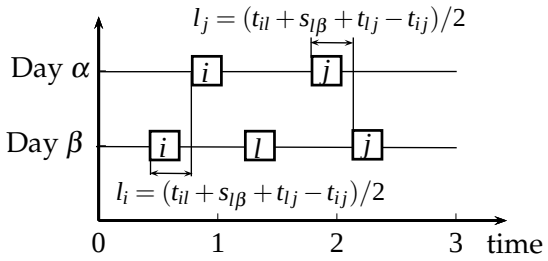


Figure 5.4 Time consistency-related inequalities (TC2).

Algorithm 7 *MDLNS*

Require: set of efficient solutions E
1: **for all** unprocessed solutions in E **do**
2: $s = getUnprocessedSolution(E)$
3: **for all** $i \in \{1,2,3\}$ **do**
4: $applyLNS(s, f_i, E)$
5: **end for**
6: $setProcessed(s)$
7: **end for**
8: **return** E

5.3.2 Heuristic approach

Our heuristic approach for approximating the Pareto-front is based on the MDLS framework for general multi-objective problems (Tricoire [174]). For a given solution, MDLS applies different local search strategies in order to identify new efficient solutions. Each objective value in the objective vector is improved by a specialized search strategy. As mentioned above, for the MOGenConVRP, we combine the MDLS framework with the LNS algorithm proposed in Chapter 4 (Kovacs et al. [103]). The resulting algorithm is referred to as MDLNS.

Algorithm 7 gives the outline of the MDLNS. Starting with an initial set of efficient solutions E, we iteratively select an unprocessed solution (line 2) and perform a local search for each objective function (line 3-5). New efficient solutions are added to E; processed solutions are marked in line 6. The algorithm stops as soon as all solutions have been processed.

Set E is initialized by using the construction heuristics proposed in Kovacs et al. [103]: Two solutions are generated by a travel-time oriented heuristic; one by ignoring driver consistency and one by restricting driver consistency to one driver per customer. One solution is generated by a time consistency oriented heuristic. Among the three solutions, we keep

only efficient ones. Function $applyLNS(s, f_i, E)$, applies the LNS algorithm for a given number of iterations on solution s with the goal of improving objective function f_i. The LNS was devised for the GenConVRP; it integrates arrival time consistency into the objective function and achieves driver consistency by restricting the feasible driver-to-customer assignments. In the MOGenConVRP, for each $i \in \{1,2,3\}$, the objective function f_i is a weighted average of the total travel time and the maximum arrival time difference; the number of different drivers per customer is bounded. The f_1 and f_3 values are weighted by α_i and $(1-\alpha_i)$, respectively:

$$f_i = \alpha_i f_1 + (1-\alpha_i) f_3. \tag{5.30}$$

We set $\alpha_1 = \frac{1}{1+\frac{\delta}{UB_3}}$ when we optimize f_1 and $\alpha_3 = \frac{1}{1+\frac{UB_1}{\delta}}$ when we optimize f_3; parameter δ is set to 10^{-3} and UB_i is an upper bound for the respective objective function. The objective values of f_1 and f_3 are equally important if $\alpha = \frac{1}{1+\frac{f_1}{f_3}}$; therefore, by using α_1 and α_3, we perform a lexicographic optimization, respectively. In both cases, f_2 is bounded by $f_2(s)+1$ in order to exclude solutions with large f_2 values. Parameter α_2 is set to $(\alpha_1+\alpha_3)/2$ when optimizing f_2 (α_1 and α_3 are calculated as defined above); additionally, f_2 is bounded by $f_2(s)-1$.

Each solution found during the search is checked whether or not it is dominated by another solution in E; if not, it is added to E. Solutions that are dominated by the new solution are removed from E. This approach is time consuming; yet, we aim for finding the best possible approximation of the Pareto-front. Adding solutions to E is performed efficiently by using the data structure illustrated in Table 5.2. For each f_2 value, we maintain a sorted list that contains all solutions with the respective f_2 value. Sorting solutions in ascending order of f_1 will automatically sort the solutions in descending order of f_3. This results from the definition of an efficient solution: a solution can be improved in one objective only by deteriorating another. In the worst case, the computation time for finding the potential insertion position of a new solution s increases logarithmically with the number of solutions in E with the same f_2 value. Solutions to the left of s and solutions with a smaller f_2 value than s have a chance of dominating s; solutions to the right of s and solutions with a larger f_2 value than s might be dominated by s.

The data structure relies on the property that $f_2 \in \mathbb{N}$ and on the assumption that the domain of f_2 is small. A similar approach for managing E is proposed in Tricoire [174] for the bi-objective case; a general approach for maintaining efficient solutions for an arbitrary number of objectives is presented in Habenicht [76] and further examined in Mostaghim

	(f_1, f_3) Pairs			
f_2	1	2	3	4
1	(189,80)	(190,71)	(191,44)	(192,40)
2	(178,77)	(179,73)	(181,70)	
3	(179,55)	(180,52)	(181,49)	(196,19)

Table 5.2 Data structure for managing the set of efficient solutions.

and Teich [120] and Sun and Steuer [166].

The computation time of the MDLNS can be reduced by applying $applyLNS(s, f, E)$ only to a subset of E, by checking only selected solutions for dominance (e.g., best found solutions or current incumbent solutions), and by reducing the number of LNS iterations.

5.4 Computational results

Solutions for different test instances are obtained by applying the described algorithms. In this section, we examine the results and analyze the trade-off between routing cost and service consistency. All algorithms are implemented in C++ and run on Intel Xeon X5550 computers with 2.67GHz. Mixed integer linear programs (i.e, calls to $opt(f, \varepsilon, \varepsilon')$) are solved by IBM's CPLEX 12.5. Approximate solution sets are obtained by merging the solutions of three independent MDLNS runs (non-efficient solutions are removed). The LNS algorithm embedded in the MDLNS performs robustly with regard to solution quality (Chapter 4, Kovacs et al. [103]); the stochastic variations are further mitigated in the multi-objective problem: MDLNS explores the solution space extensively and we check for each generated solution whether or not it is efficient. The parameters of the LNS are set as proposed by Kovacs et al. [103].

5.4.1 Data sets

The test instances for the MOGenConVRP are taken from the GenConVRP (Chapter 4, Kovacs et al. [103]). The data sets are named $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, with respect to the service frequencies that are set to 50%, 70%, and 90%, respectively. Service frequency is a measure for the variation in the demand of the customers. A service frequency of 100% means that each customer is visited each day. In this case, we could achieve perfect service consistency by executing the same routing plan on each day of the planning horizon. With decreasing service frequency, there is more variation in the demand and it becomes more difficult to provide consistent service. Each data set consists of 12 instances with 50 to 199 customers

and a planning horizon of five days. Additionally, we consider data set C_{small} and C_{smallE} with 10 and 5 instances, respectively. Set C_{small} contains instances with 10 to 12 customers that are visited over a planning horizon of three days. In set C_{smallE}, we extend the instances with ten customers to a planning horizon of five days.

5.4.2 Results for small instances

Experiments on data set C_{small} are mainly performed for benchmarking tests. In this section, we examine the results to evaluate the efficiency of the single objective optimizer. Furthermore, we compare the 2D method and the 3D method, and evaluate the performance (i.e., solution quality and computation time) of the MDLNS.

Evaluating the single objective optimizer

In Section 5.3.1, we proposed several valid inequalities for the single-objective model. The efficiency of these inequalities is tested by optimizing only the total travel time (f_1) of the instances in C_{small} and by bounding the maximum number of different drivers per customer (f_2) and the maximum arrival time difference (f_3). Each instance is run with three different driver consistency configurations ($z_{max} \leq Z$, $Z \in \{1,2,3\}$) and four different arrival time consistency configurations ($l_{max} \leq l \min_{(i,j)\in A^0}\{t_{ij}\}$, $l \in \{0,1,2,3\}$). The total number of test instances is 120, i.e., ten instances times (3×4) configurations.

In Table 5.3, we present the results for different combinations of valid inequalities. The first column gives the configuration of the optimizer. Configuration "Plain" refers to the model specified by inequalities (5.5) - (5.18). The remaining rows refer to the plain model extended by the respective inequalities: SB are symmetry breaking inequalities, $ST2$ and $ST3$ are subtour elimination inequalities with $|S| = 2$ and $|S| = 3$, respectively, $TC1$ and $TC2$ are time consistency-related inequalities, CC are capacity cuts, and "All" means that all inequalities are added to the model. Capacity cuts are added only if there are not more than 100 combinations of choosing $|S|$ customers out of N_d (i.e., $\binom{N_d}{|S|} \leq 100$) and only for subsets S that require at least two vehicles (i.e., $\frac{\sum_{i\in S} q_{id}}{Q} > 1$). The second column gives the number of instances that were solved to proven optimality within 24 hours. The remaining columns show the best, average, and worst speed up that is obtained compared to the plain model. The speed up values are averaged over the 83 instances that were solved by each configuration.

Each configuration is helpful in increasing the number of instances solved. However, the computation time might increase by 12753% (from 82.75 seconds to 10636.6 seconds)

		Speed up (%)		
Configuration	Solved	Best	Average	Worst
Plain	95	-	-	-
TC2+SB	96	99.86	-587.74	-12753.53
ST2	96	99.81	-34.75	-3430.45
TC1	97	99.69	-63.55	-2971.62
ST3	97	99.09	14.46	-808.43
TC2+SB+ST3	98	99.87	-448.31	-12425.50
CC	99	99.38	-3.95	-2283.39
All	101	99.96	-515.24	-14582.65
TC2+ST3	103	98.97	-129.24	-3164.06
SB	103	99.89	-120.06	-3845.46
TC2	104	98.15	-231.78	-5785.63
SB+ST3	104	99.90	18.71	-793.68

Table 5.3 Efficiency of valid inequalities for the single-objective problem on 120 test instances in C_{small}. The configurations of the optimizer are sorted in ascending order of the number of instances solved within 24 hours.

in the worst case (see configuration TC2+SB). Even though configuration TC2+SB performs poorly, configurations TC2 and SB achieve good results separately. This indicates that combining different inequalities might deteriorate the performance significantly. The most efficient configuration is SB+ST3: we can solve 104 out of 120 instances with an average speed up of more than 18%. This configuration is used in the lexicographic optimizer $opt(f, \varepsilon, \varepsilon')$.

Comparing the 2D method and the 3D method

In the MOGenConVRP, we assume that the number of vehicles is not restrictive. This assumption is necessary for achieving the highest consistency level for any problem instance: in the extreme case, we could assign each customer to one driver exclusively that visits the customer at exactly the same time of the day. However, in most solutions only a small fraction of the fleet is active; several customers are consolidated on each route in order to reduce travel cost. We define a reasonable fleet size by generating a solution with $l_{max} = 0$, $z_{max} = 1$, and a total travel time that is not worse than 1.2 times the optimal travel time. The fleet size is set to the number of vehicles used in the obtained solution. The computation time of $opt(f, \varepsilon, \varepsilon')$ is significantly reduced by this preprocessing approach; however, it might prevent us from finding optimal solutions. Figure 5.5 illustrates an example: Four customers require service over several periods. The demand is constant over the entire planning period, i.e., repeating the same routing plan each day results in perfect service consistency. Customers 1 and 2 have a demand of $1 - \delta$ and customers 3 and 4 have a demand of δ;

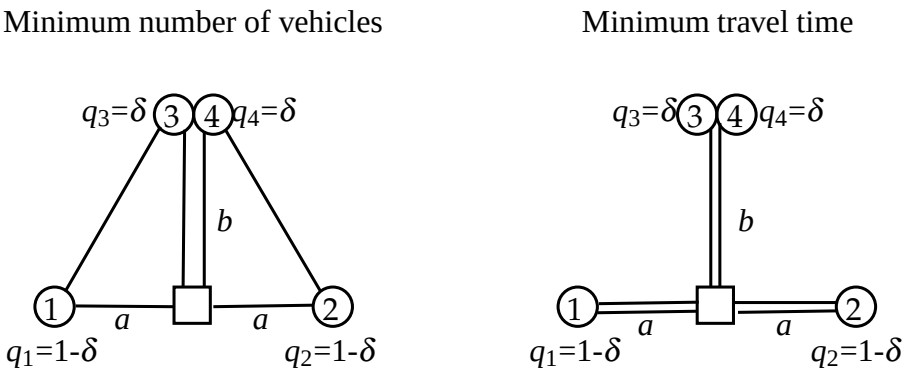


Figure 5.5 Preprocessing approach might prevent finding optimal solutions. The demand (q) is given for each customer; δ is a small number < 0.5. Each vehicle has a capacity of one unit.

δ is a small number $0 < \delta < 0.5$. The vehicle capacity is one. The travel times are given next to the edges. The figure at the left shows a solution with two vehicles (i.e., 0-1-3-0 and 0-2-4-0); the figure at the right shows a solution with three vehicles (i.e., 0-1-0, 0-2-0, and 0-3-4-0). For each $b > 0$, the solution with two vehicles has longer travel time per day ($2a + 2b + 2\sqrt{a^2 + b^2}$) than the solution with three vehicles ($4a + 2b$). In Table 5.4, we compare the 2D method to the 3D method on data set C_{small}. The second column gives the computation time in minutes for the preprocessing phase, $CPU_{pp}(min)$. The third column shows the resulting decrease in the fleet size $|K|$. The fourth column is the number of points on the Pareto-front $|P|$. For both solution approaches, we report the number of calls to the lexicographic optimizer $opt(f, \varepsilon, \varepsilon')$ and the total computation time in minutes, $CPU(min)$. The last column shows the improvement in the computation time of the 3D method over the 2D method.

On average, the preprocessing phase described above reduces the fleet size by 73.5% in one minute. The size of the model is significantly influenced by the number of vehicles; yet, our approach might generate solutions that are not optimal with regard to travel cost. In the 2D method, the lexicographic optimizer is called only 20.5 times on average (the average number of points on the Pareto-front is 20.1). The 3D method requires 24.7 calls; nevertheless, the 3D method is 5.6% faster on average. The subproblems are smaller in the 3D method and, therefore, each call is executed more quickly. The advantage of the 3D method seems to increase with the number of customers: the 3D method is 5.44% slower on average than the 2D method with 10 customers (first five instances) but 16.65% faster with 12 customers (last five instances).

Instances	$CPU_{pp}(min)$	$Imp_K(\%)$	$\|P\|$	2D method		3D method		$Imp_{CPU}(\%)$
				calls $opt()$	CPU(min)	calls $opt()$	$CPU(min)$	
convrp_10_test_1.vrp	0.01	70	16	17	5.63	19	5.41	3.87
convrp_10_test_2.vrp	0.09	80	20	20	7.86	24	8.39	-6.72
convrp_10_test_3.vrp	0.10	70	10	10	19.64	11	21.76	-10.80
convrp_10_test_4.vrp	0.23	70	10	11	51.83	13	54.32	-4.81
convrp_10_test_5.vrp	0.02	70	18	20	19.75	22	21.48	-8.76
convrp_12_test_1.vrp	5.90	75	21	21	224.73	26	260.82	-16.06
convrp_12_test_2.vrp	0.07	75	25	25	265.35	32	178.11	32.88
convrp_12_test_3.vrp	3.69	75	34	34	1900.50	45	1266.44	33.36
convrp_12_test_4.vrp	0.46	75	26	26	953.66	31	679.51	28.75
convrp_12_test_5.vrp	0.02	75	21	21	50.43	24	48.25	4.33
Average	1.06	73.50	20.10	20.50	349.94	24.70	254.45	5.60

Table 5.4 Comparison of 2D method and 3D method on data set C_{small}.

Evaluating the performance of the MDLNS

The performance of multi-objective optimizers is defined by the quality of the generated set of solutions and the required computation time. With a single objective, the quality of an optimizer is proportional to the achieved objective value: the lower the objective value (in minimization problems), the better the algorithm. However, in multi-objective optimization, we need to evaluate sets of objective vectors. Zitzler et al. [191] list three features of high quality multi-objective optimizers: the distance between P_{approx} and P is small, the points in P_{approx} are distributed evenly, and the range of each objective value (i.e., for each objective function, the difference between the best and worst objective value in P_{approx}, respectively) is large. Unary quality measurements assign each approximation set a single value that reflects a certain quality feature (Zitzler et al. [193]). Typically, heuristics are evaluated by several quality measurements. (Binary quality measurements assign a value to each pair of approximation sets and are used for comparing heuristics by pairs.) Papers that examine quantitative approaches for evaluating multi-objective optimizers are presented, e.g., by Knowles and Corne [99], Sarker and Coello Coello [148], Van Veldhuizen and Lamont [176], and Zitzler et al. [193].

We evaluate the quality of the MDLNS by applying five unary quality measurements. Each measurement has drawbacks that prevent us from using it as the only quality measurement. A meaningful evaluation of the algorithm is possible only if all measurements are considered at the same time.

Number of points on the Pareto-front Providing more efficient solutions means giving the decision maker more choices (Schott [154], Van Veldhuizen [177]). The cardinality of the Pareto-front ($|P_{approx}|$) gives the number of alternative solutions generated by the

algorithm. However, this measurement ignores the quality of the approximation: a single solution might dominate the entire approximation set. Additionally, decision makers might be overwhelmed by an (unnecessarily) large number of solutions.

Error ratio The error ratio (*ER*) (Van Veldhuizen [177]) gives the share of points in P_{approx} that are not in P:

$$ER = \frac{\sum_{p \in P_{approx}} e_p}{|P_{approx}|}. \tag{5.31}$$

For each point $p \in P_{approx}$, e_p is 0 if $p \in P$ and 1, otherwise. Lower ER values indicate better approximations; yet, the value may be misleading. Consider, for example, two approximation sets: one with a single solution that is $\in P$ and one that contains many alternative solutions $\in P$ and one solution $\notin P$. The first set has an ER value of zero and would, therefore, be preferred to the second set with $ER > 0$.

Generational distance The generational distance (*GD*) gives the distance between P_{approx} and P (Rudolph [146], Van Veldhuizen [177], Van Veldhuizen and Lamont [178, 179]):

$$GD = \frac{\sqrt{\sum_{p \in P_{approx}} d_p^2}}{|P_{approx}|}; \tag{5.32}$$

d_p is the Euclidean distance from a point p to the closest point $q \in P$ ($d_p = min_{q \in P} \sqrt{(p-q)(p-q)}$). A generational distance of zero indicates that all points in P_{approx} are also in P; the larger *GD*, the larger the distance between the approximation set and the optimal Pareto-front. Similar to the error ratio, *GD* may favor algorithms that generate one solution $\in P$ over algorithms that find many solutions $\in P$ but also solutions $\notin P$.

Unary ε-indicator The unary ε-indicator (I_ε) defines by how much an approximation set is worse then the optimal set with regard to all objective functions (Zitzler et al. [193]):

$$I_\varepsilon = \max_{q \in P} \min_{p \in P_{approx}} \max_{i \in \{1,2,3\}} \frac{p_i}{q_i}. \tag{5.33}$$

For each point $p \in P_{approx}$ and each point $q \in P$, we have $p_i \leq I_\varepsilon q_i \; \forall \; i = \{1, ..., |p|\}$; $I_\varepsilon = 1$ indicates that all points in P_{approx} are also in P. The ε-indicator is a worst case quality measurement; so, the quality evaluation might be based on a single poor solution in P_{approx}.

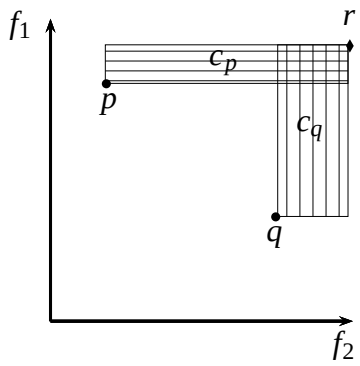


Figure 5.6 Example for the hypervolume in the two-dimensional objective space (i.e., hyperarea).

Hypervolume Hypervolume (HV) is the size of the bounded objective space that is dominated by a set of solutions P:

$$HV = \{\bigcup_{p} c_p | p \in P\}. \tag{5.34}$$

Each point $p \in P$ covers a space c_p where c_p is the volume of the cuboid that is defined by p and a reference point r ($r_i \geq p_i \; \forall \; p \in P, i = \{1, ..., |p|\}$). An example of a hypervolume in the two-dimensional objective space is given in Figure 5.6: p and q are elements of P, r is a reference point, and c_p and c_q are areas dominated by p and q, respectively. The union of c_p and c_q is the hyperarea. The HV measurement is sensitive to the choice of the reference point as it may affect the quality evaluation (Knowles and Corne [99]). For easier comparability, we report the gap between the HV of the optimal set P (HV_P) and the HV value of the approximation set P_{approx} ($HV_{P_{approx}}$):

$$GapHV = \frac{100(HV_P - HV_{P_{approx}})}{HV_P} \tag{5.35}$$

Computing HV in a three-dimensional objective space is nontrivial. We apply version 1.3 of the recursive, dimension-sweep algorithm for computing the hypervolume proposed and implemented by Fonseca et al. [62, 64].

In Table 5.5 and Table 5.6, we evaluate the quality of the MDLNS on data sets C_{small} and C_{smallE}, respectively. Solution sets are aggregated over three runs. The second column gives the number of points in P. The remaining columns show the five unary quality indicators for

	Opt.	$3*10^4$ iterations					$4*10^4$ iterations					$5*10^4$ iterations				
Instances	$\|P\|$	$\|P\|$	ER	GD	I_ε	$GapHV$	$\|P\|$	ER	GD	I_ε	$GapHV$	$\|P\|$	ER	GD	I_ε	$GapHV$
c_10_1	16	16	0.063	0.006	1.001	0.000	16	0.125	0.006	1.001	0.000	17	0.118	0.006	1.001	0.000
c_10_2	20	19	0.105	0.000	1.004	0.000	18	0.111	0.000	1.005	0.000	18	0.111	0.000	1.005	0.000
c_10_3	10	11	0.182	0.092	1.008	0.001	11	0.182	0.092	1.008	0.001	11	0.182	0.092	1.008	0.001
c_10_4	10	13	0.538	0.240	1.024	2.073	12	0.500	0.260	1.024	2.073	13	0.538	0.240	1.024	2.073
c_10_5	17	19	0.421	0.146	1.012	0.537	20	0.450	0.131	1.025	0.613	19	0.421	0.137	1.025	0.788
c_12_1	21	26	0.308	0.387	1.076	0.280	25	0.320	0.392	1.076	0.445	29	0.414	0.377	1.049	0.298
c_12_2	25	28	0.429	0.170	1.062	1.093	29	0.414	0.194	1.052	1.586	26	0.385	0.162	1.052	1.001
c_12_3	34	33	0.061	0.054	1.027	0.002	34	0.088	0.052	1.027	0.002	33	0.091	0.054	1.027	0.002
c_12_4	26	28	0.179	0.040	1.009	0.142	27	0.111	0.039	1.007	0.006	26	0.115	0.002	1.007	0.001
c_12_5	21	20	0.250	0.132	1.027	0.367	19	0.105	0.118	1.021	0.044	20	0.200	0.127	1.021	0.071
Average	20.0	21.3	0.253	0.127	1.025	0.449	21.1	0.241	0.128	1.025	0.477	21.2	0.257	0.120	1.022	0.424

Table 5.5 Quality evaluation of the MDLNS on data set C_{small}; the names of the instances are abbreviated.

	Opt.	$3*10^4$ iterations					$4*10^4$ iterations					$5*10^4$ iterations				
Instances	$\|P\|$	$\|P\|$	ER	GD	I_ε	$GapHV$	$\|P\|$	ER	GD	I_ε	$GapHV$	$\|P\|$	ER	GD	I_ε	$GapHV$
cN_10_1	44	46	0.413	0.241	1.086	1.441	54	0.481	0.308	1.069	1.287	44	0.386	0.196	1.086	1.721
cN_10_2	35	38	0.395	0.105	1.165	0.189	37	0.459	0.194	1.133	0.623	35	0.429	0.123	1.155	0.556
cN_10_3	28	31	0.290	0.361	1.089	0.865	27	0.185	0.265	1.089	0.365	26	0.154	0.162	1.079	0.252
cN_10_4	37	35	0.114	0.165	1.071	0.180	39	0.282	0.227	1.046	0.101	42	0.429	0.166	1.072	0.511
cN_10_5	37	46	0.543	0.164	1.035	0.926	48	0.583	0.180	1.057	1.603	46	0.565	0.185	1.068	0.934
Average	36.2	39.2	0.351	0.207	1.089	0.720	41.0	0.398	0.235	1.079	0.796	38.6	0.393	0.167	1.092	0.795

Table 5.6 Quality evaluation of the MDLNS on data set C_{smallE}; the names of the instances are abbreviated.

the MDLNS with 30, 40, and 50 thousand iterations per LNS run. The number of iterations seems to have a low influence on the solution quality. Starting from different solutions, the MDLNS explores the solution space by optimizing several objective functions. Therefore, the algorithm performs robustly regardless of the chosen parameters. In both data sets, each MDLNS configuration provides at least as many points on average as there are points on the Pareto-front. In C_{small}, on average, 24.1% to 25.7% of the solutions are not element of P; in C_{smallE} the average ER value is between 35.1% and 39.8%. The average generational distance is between 0.120 and 0.128 in C_{small} and between 0.167 and 0.235 in C_{smallE}. The average unary ε-indicator shows that the approximation set is between 2.2% and 2.5% worse than the optimal set in C_{small} and less than 9.2% worse in C_{smallE}. The $GapHV$ indicator shows that the MDLNS provides approximation sets that, on average, have only 0.424% to 0.477% smaller hypervolumes than P in C_{small} and between 0.720% and 0.796% smaller hypervolumes in C_{smallE}. For each instance, the reference point for computing the hypervolume is (r_1, r_2, r_3) where r_1 is the longest total travel time, r_2 is the maximum number of drivers per customer, and r_3 is the maximum arrival time difference among all solutions in the examined approximation sets. Given these results, we can conclude that the quality of the MDLNS approach is satisfactory for performing a meaningful trade-off analysis.

		MDLNS		
	3D method	30k	40k	50k
convrp_10_test_1	5.41	0.61	0.86	1.09
convrp_10_test_2	8.39	1.07	1.42	1.65
convrp_10_test_3	21.76	0.59	0.80	0.98
convrp_10_test_4	54.32	0.65	0.82	1.03
convrp_10_test_5	21.48	1.13	1.47	1.75
convrp_12_test_1	260.82	1.50	2.11	2.62
convrp_12_test_2	178.11	2.29	3.29	3.30
convrp_12_test_3	1266.44	2.00	2.77	3.29
convrp_12_test_4	679.51	1.82	2.26	2.70
convrp_12_test_5	48.25	1.08	1.42	1.69
Average	254.45	1.27	1.72	2.01

Table 5.7 Runtime evaluation of the MDLNS on data set C_{small}. The reported average computation time is given in minutes per run.

		MDLNS		
	3D method	30k	40k	50k
convrpNew_10_1_5	836.47	3.56	5.63	5.76
convrpNew_10_2_5	3953.57	3.21	3.99	6.73
convrpNew_10_3_5	2434.82	2.44	3.05	3.62
convrpNew_10_4_5	1540.03	2.80	4.32	5.34
convrpNew_10_5_5	427.33	5.17	5.18	6.86
Average	1838.44	3.43	4.43	5.66

Table 5.8 Runtime evaluation of the MDLNS on data set C_{smallE}. The 3D method is run on eight threads; the reported time is the wall-clock time in minutes per run.

The computation time of the MDLNS is examined in Table 5.7 for data set C_{small} and in Table 5.8 for data set C_{smallE}. As a reference point, we report the computation time of the 3D method in the second column; the remaining columns show the average computation time for the MDLNS with 30, 40, and 50 thousand iterations per LNS run, respectively. On data set C_{small} (Table 5.7), the MDLNS requires not more than 2.01 minutes on average. This result is noticeable when compared to the 3D method that requires more than four hours. For solving the instances in C_{smallE} (Table 5.8), we run the 3D method in parallel on eight threads; the reported time is the wall-clock time. MDLNS runs on a single thread. The 3D method requires more than 30 hours while MDLNS provides results between 3.34 and 5.66 minutes on average. This result indicates that applying exact solution approaches in practice is unrealistic.

5.4.3 Results for large instances (data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$)

The trade-off analysis between total travel time, driver consistency, and arrival time consistency is performed on data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$. The results are obtained by applying the MDLNS with 40 thousand iterations per LNS run. Solution sets are aggregated over three runs. An example of the output of the algorithm is shown at the left of Figure 5.7. The horizontal axis of the diagram shows the maximum arrival time difference and the vertical axis shows the total travel time. The maximum number of drivers per customer is distinguished by colors; each f_2 value is associated with a different color.

Each point p with a lower p_3 value than $0.01 \max_{p \in P}\{p_3\}$ is excluded from the approximation set[2]. In instance Christofides_4_5_0.7, for example, the travel time increases by 332% in the solution with $f_3 = 0$ compared to the solution with the second best f_3 value. Solutions with very small f_3 values and very large f_1 values would distort the analysis and are, therefore, treated as outliers. We think this approach would also be justified from a managerial perspective, because aiming for perfect arrival time consistency is unreasonable.

The magnitude of the objective values varies instance-by-instance. In order to make general statements, we transform the Pareto-front P into an improvement front T that shows the percentage improvement in one objective as a function of the percentage improvement in another objective:

$$T = \{(a,b,c) : a = \frac{p_1^N - p_1}{p_1^N}, b = p_2, c = \frac{p_3^N - p_3}{p_3^N} \; \forall \; p \in P\}. \tag{5.36}$$

Each point on the Pareto-front is transformed into a triple (a,b,c): a and c are the improvements in f_1 and f_3, respectively, compared to the Nadir-point (p_1^N, p_3^N) ($p_1^N = \max_{p \in P}\{p_1\}$ and $p_3^N = \max_{p \in P}\{p_3\}$); b is the number of drivers per customer in the respective solution. The right figure in Figure 5.7 shows the improvement front associated with the Pareto-front at the left. By putting the absolute values into perspective, we can aggregate several results in order to conduct a meaningful analysis.

Cost of arrival time consistency

The correlation between total travel time and maximum arrival time difference is unclear when examining single results. For example, in the figure at the left of Figure 5.8, we see a partly convex relationship, i.e., we have to give up much travel time in order to improve

[2] Each point on the Pareto-front P represents a solutions in the set of efficient solutions E, i.e., for each point $p = (p_1, p_2, p_3)$, there exists a solution $s \in E$ such that $p = f(s)$.

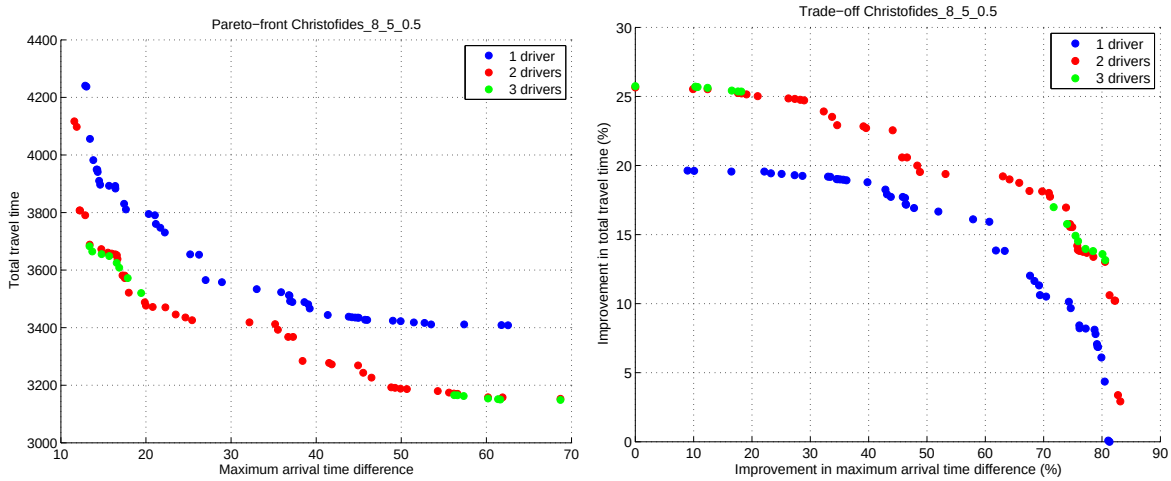


Figure 5.7 Example of a Pareto-front and the associated improvement front.

arrival time consistency a little. In the figure at the right, the maximum arrival time difference seems to improve gradually with increasing travel time. We examine the trade-off that is to be expected by aggregating the results of several instances as follows: The hull of the $f_1 - f_3$ projection of the improvement front is defined by a set of line segments that are parallel to the f_3-axis and bound the improvement in f_1 from above. Let $T^{f_1-f_3}$ denote the subset of T that contains only points that are non-dominated in the $f_1 - f_3$ plane. The points in $T^{f_1-f_3}$ are sorted in ascending order of the improvements in f_3; the line segment between each pair of neighboring solutions (p^i, p^{i+1}) is defined by the improvement in f_1 of point p^{i+1}. By quantizing the improvement in f_3 and averaging the hull over several instances, we can aggregate the results of several instances and observe the correlation between the maximum arrival time difference and the total travel time that is to be expected. An example of a quantized hull is given at the left of Figure 5.9; the average improvement curve for data sets with different service frequency is given at the right.

By ignoring driver consistency, we obtain a relaxed improvement curve between cost and arrival time consistency. For examining a strict scenario, we use the same principle as above but the hull is defined only for points in $\{T \,|\, f_2 = 1\}$. An example of this hull is given at the right of Figure 5.10; the left figure shows the average improvement curves for different service frequencies when $f_2 = 1$. The findings are summarized in Table 5.9 and 5.10. The tables show by how much the total travel time increases compared to the solution with minimal travel cost when the maximum arrival time difference is decreased in the relaxed scenario and in the strict scenario, respectively. The first column is the improvement in l_{max}; columns 2–4 are the percentage increase in the total travel time (TT) for data sets $C_{0.5}$, $C_{0.7}$,

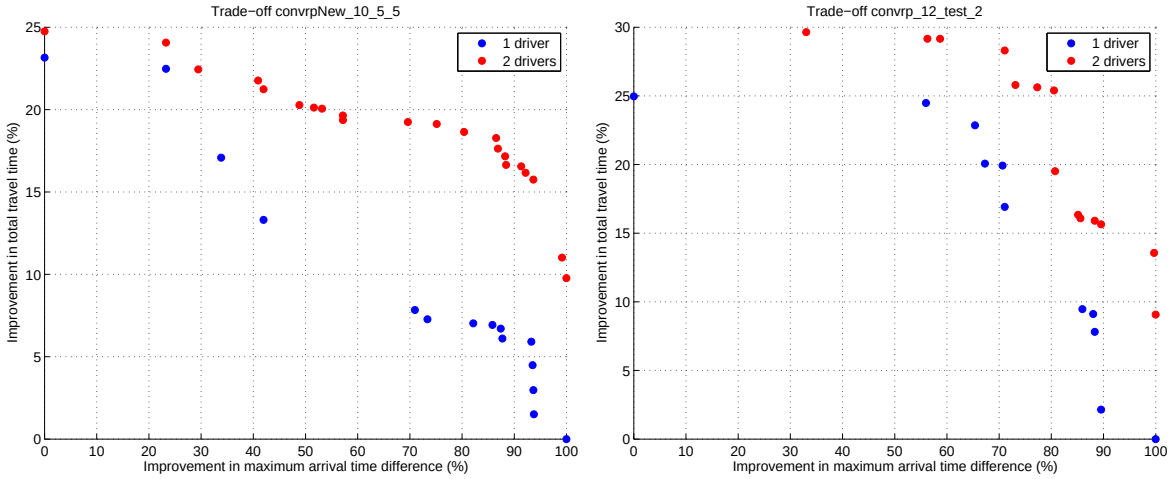


Figure 5.8 Partly convex relationship between travel time and arrival time consistency at the left and stepwise relationship at the right.

	Increase in TT (%)			
Imp l_{max} (%)	$C_{0.5}$	$C_{0.7}$	$C_{0.9}$	Average
10	0.09	0.08	0.14	0.10
30	0.53	0.55	0.39	0.49
50	1.86	1.77	1.11	1.58
70	5.09	3.99	2.46	3.84
90	17.92	21.42	11.86	17.07

Table 5.9 Cost of arrival time consistency in percent when driver consistency is ignored for data sets with 50%, 70%, and 90% service frequency (i.e., data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$).

and $C_{0.9}$, respectively. The last column gives the average increase in the travel time. The results are similar in both tables: we can improve arrival time consistency by 50% at the cost of increasing travel time by less than 1.58%, on average. Decreasing l_{max} by 70% is achieved at 3.84% higher cost when driver consistency is ignored and at 2.43% higher cost when driver consistency is optimal. Decreasing l_{max} further significantly increases travel time: the travel time increases by 17.07% in the relaxed scenario and by 14.81% in the strict scenario.

Cost of driver consistency

We can use the hull of the improvement front for examining the cost of driver consistency. In Table 5.11, we compare the relaxed scenario to the strict scenario. For each data set and for different levels of arrival time consistency, we give the average increase in travel time in

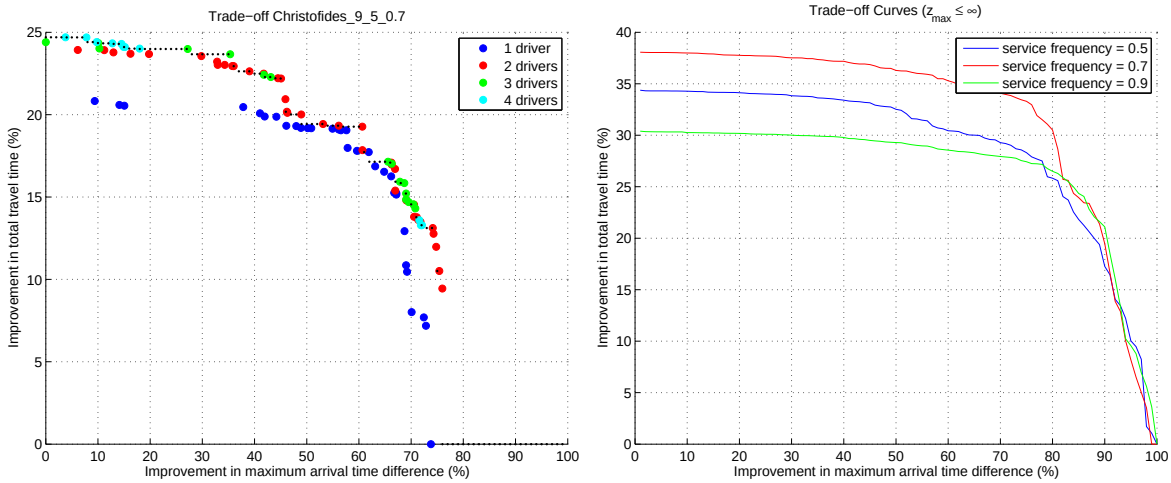


Figure 5.9 Hull of instance Christofides_9_5_0.7 when driver consistency is ignored (i.e., in the relaxed scenario) at the left and the average improvement curves for different service frequencies at the right.

	Increase in TT (%)			
Imp l_{max} (%)	$C_{0.5}$	$C_{0.7}$	$C_{0.9}$	Average
10	0.02	0.00	0.01	0.01
30	0.06	0.04	0.02	0.04
50	0.72	0.38	0.03	0.38
70	3.97	2.24	1.07	2.43
90	15.73	18.27	10.42	14.81

Table 5.10 Cost of arrival time consistency in percent when driver consistency is fixed to one driver per customer for data sets with 50%, 70%, and 90% service frequency (i.e., data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$).

the strict scenario ($z_{max} = 1$) and in the relaxed scenario ($z_{max} \leq \infty$), respectively, compared to the solution with minimal travel time (i.e., the solution that ignores arrival time and driver consistency). Column Diff gives the difference between the strict scenario and the relaxed scenario. The comparison shows the cost of perfect driver consistency for different levels of arrival time consistency.

The larger the maximum arrival time difference, the higher the cost of driver consistency: the average increase in travel time caused by servicing each customer by a single driver is 4.61%, 3.88%, and 2.08% for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively, when l_{max} is improved by 10%. At the highest level of arrival time consistency (improving l_{max} by 90%), the increase in travel time decreases to 2.49%, 0.81%, and 0.77% in $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively. This result suggests that arrival time consistency and driver consistency can be

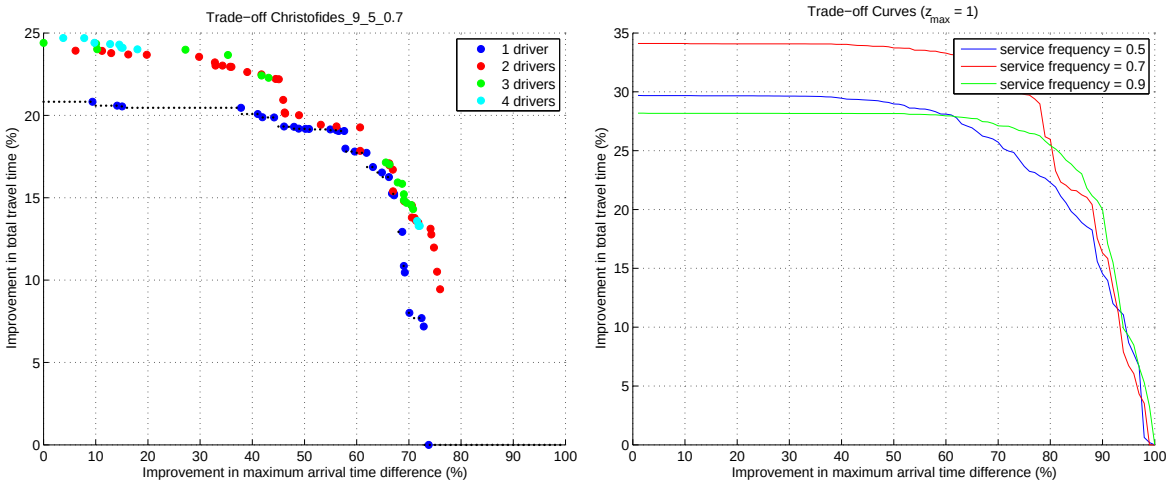


Figure 5.10 Hull of instance Christofides_9_5_0.7 when driver consistency is fixed to one driver per customer (i.e., in the strict scenario) at the left and the average improvement curves for different service frequencies at the right.

	Increase in TT (%)								
	$C_{0.5}$			$C_{0.7}$			$C_{0.9}$		
Imp l_{max} (%)	$z_{max}=1$	$z_{max}\leq\infty$	$Diff$	$z_{max}=1$	$z_{max}\leq\infty$	Diff	$z_{max}=1$	$z_{max}\leq\infty$	Diff
10	4.70	0.09	4.61	3.96	0.08	3.88	2.22	0.14	2.08
30	4.74	0.53	4.20	3.99	0.55	3.44	2.23	0.39	1.84
50	5.40	1.86	3.55	4.34	1.77	2.57	2.24	1.11	1.13
70	8.65	5.09	3.56	6.20	3.99	2.21	3.28	2.46	0.83
90	20.41	17.92	2.49	22.22	21.42	0.81	12.63	11.86	0.77

Table 5.11 Cost of arrival time consistency in percent when driver consistency is fixed to one driver per customer compared to solutions in which driver consistency is ignored; $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$ are data sets with 50%, 70%, and 90% service frequency.

optimized simultaneously.

The cost of driver consistency is further examined by defining a variant of the unary ε-indicator (see Section 5.4.2):

$$I_{\varepsilon}^{xQ} = \text{x-quantile} \min_{q \in P^{Z+1}} \max_{p \in P^{Z}} \max_{i \in \{1,3\}} \frac{p_i}{q_i}. \tag{5.37}$$

Sets P^Z and P^{Z+1} are subsets of the Pareto-front that contain only points with $p_2 = Z$ and $p_2 = Z+1$, respectively. The modified ε-indicator gives the value for which x% of the solutions become less than I_{ε}^{xQ} times worse if z_{max} is reduced from $Z+1$ to Z.

Figures 5.11, 5.12, and 5.13 show the average I_{ε}^{xQ} values for $x = 25\%$, 50%, and 75%,

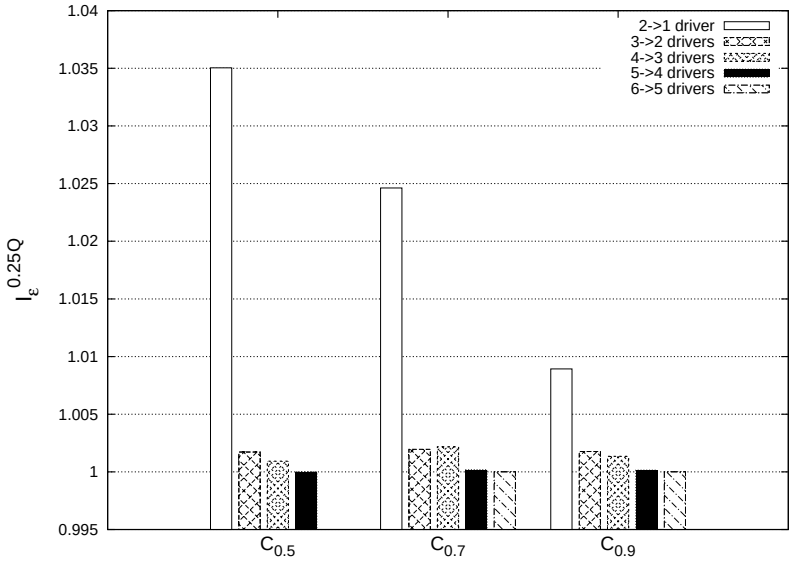


Figure 5.11 First quartile of the average increase in cost for improving driver consistency. A value of $I_\varepsilon^{0.25Q} = 1$ indicates that the solutions are not deteriorating when the maximum number of different drivers per customer is reduced. Results are given for different levels of driver consistency and different service frequencies.

respectively. In each figure, we show by how much the solutions deteriorate for different Z values and for different service frequencies (i.e., for data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$).

The pattern in all figures is similar: The solutions deteriorate most if the number of drivers per customer is reduced from two drivers to one driver. The difference between solutions is minor when $Z > 1$; in this case, the solutions do not deteriorate by more than 0.9%. The effect of driver consistency decreases with increasing service frequency, i.e., decreasing fluctuations in the demand. A quarter of the solutions with $z_{max} = 1$ in data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$ deteriorates by less than 3.5%, 2.5%, and 0.9%, respectively, compared to solutions with $z_{max} = 2$ (Figure 5.11); half of the solutions deteriorate by less than 5.1%, 3.7%, and 1.5% in data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively (Figure 5.12). In three quarters of the solutions, the cost of achieving perfect driver consistency increases by less than 6.7%, 5%, and 2.3% in in data sets $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$, respectively (Figure 5.13).

Effect of strict driver consistency on arrival time consistency

In this section, we examine the effect of enforcing perfect driver consistency on arrival time consistency when the focus is on travel cost. Many companies, e.g., in the small package shipping industry, perform a districting strategy in order to reduce the operational complex-

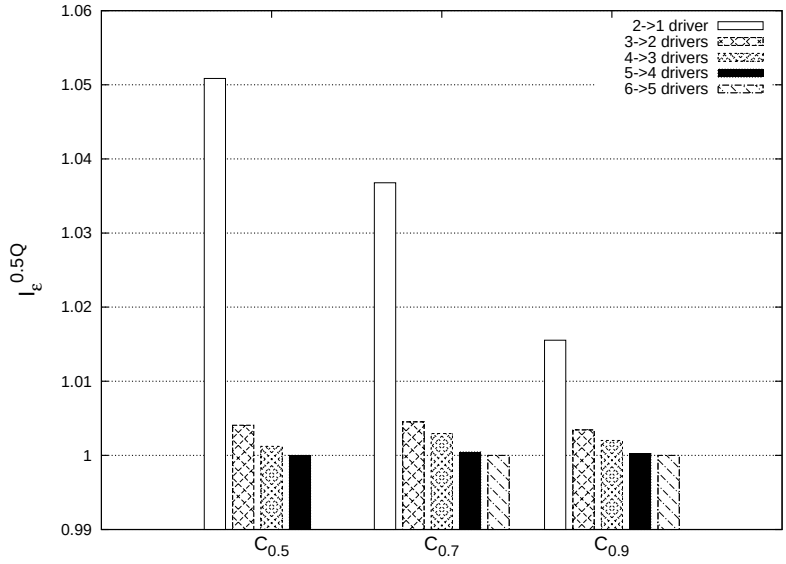


Figure 5.12 Median of the average increase in cost for improving driver consistency. A value of $I_\varepsilon^{0.5Q} = 1$ indicates that the solutions are not deteriorating when the maximum number of different drivers per customer is reduced. Results are given for different levels of driver consistency and different service frequencies.

ity (see Section 2.4.2). Here, the service territory is partitioned into smaller areas called districts and each driver is assigned to one district. Districts are designed in a tactical phase and routes are planned in an operational phase. Typically, the focus in both phases is on travel cost. Each customer experiences perfect driver consistency but the effect of districting on arrival time consistency is unclear. The intuition is that it is easier to achieve arrival time consistency for each district separately than for the entire service territory. An example is illustrated in the left figure of Figure 5.14: By restricting the maximum number of different drivers to one (which is equivalent to assigning each driver to one district) and optimizing the total travel time, we implicitly improve arrival time consistency by 81% compared to the solution without districting when travel cost is the primary objective. However, in the figure at the right, we see the opposite: with districting, we have a 32% worse arrival time consistency (and longer travel time), i.e., improving arrival time consistency and reducing the number of different drivers per customer are conflicting objectives.

In Table 5.12, we present aggregated results that show the effect of strict driver consistency on arrival time consistency. We report the number of instances in which the arrival time difference is larger with districting (i.e., with emphasis on cost and $z_{max} = 1$) than without (# Worst l_{max}) and the average improvement in l_{max}. (Each l_{max} value refers to the

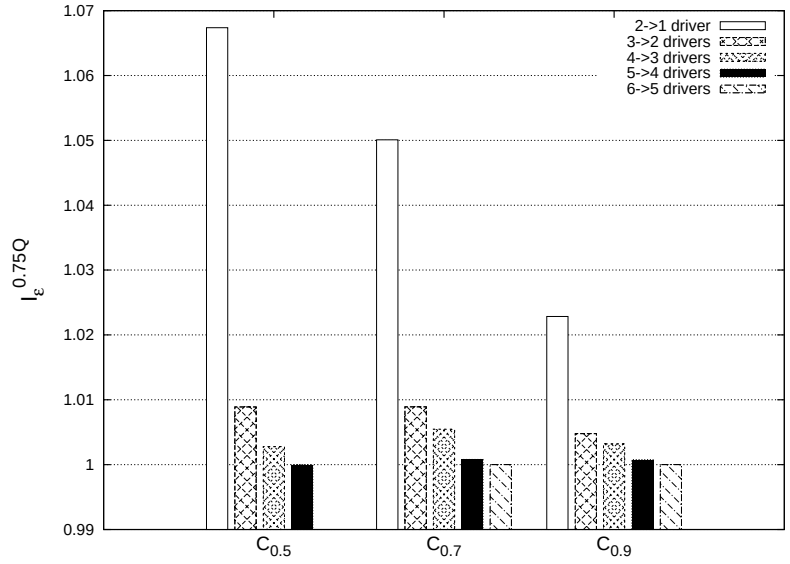


Figure 5.13 Third quartile of the average increase in cost for improving driver consistency. A value of $I_\varepsilon^{0.75Q} = 1$ indicates that the solutions are not deteriorating when the maximum number of different drivers per customer is reduced. Results are given for different levels of driver consistency and different service frequencies.

solution in P_{approx} with minimal total travel time.) Additionally, we use another variant of the unary ε-indicator to measure by how much the districting solution, denoted by p, is worse compared to solutions in which each customer may be visited with an arbitrary number of different drivers (denoted by $P^{Z\leq\infty}$):

$$I'_\varepsilon = \min_{q\in P^{Z\leq\infty}} \max_{i\in\{1,3\}} \frac{p_i}{q_i}. \tag{5.38}$$

The effect of districting depends primarily on the number of days in the planning horizon. The planning horizon involves three days in data set C_{small} and five days in the remaining data sets (i.e., C_{smallE}, $C_{0.5}$, $C_{0.7}$, and $C_{0.9}$). In C_{small}, the arrival time difference increases in 6 out of 10 instances; nevertheless, l_{max} decreases by 13.34% on average; in terms of the I'_ε indicator, the closest efficient solution without districting is 1.145 times better, on average, than the solution with districting. In data set C_{smallE}, l_{max} improves in each of the five instances due to strict driver consistency; the average improvement is 26%. The solutions are, on average, 4.8% worse compared to solutions without districting, i.e., $I'_\varepsilon = 1.048$. A positive impact of districting on arrival time consistency is observed in the large instances. The arrival time difference increases in only one out of 36 instances and

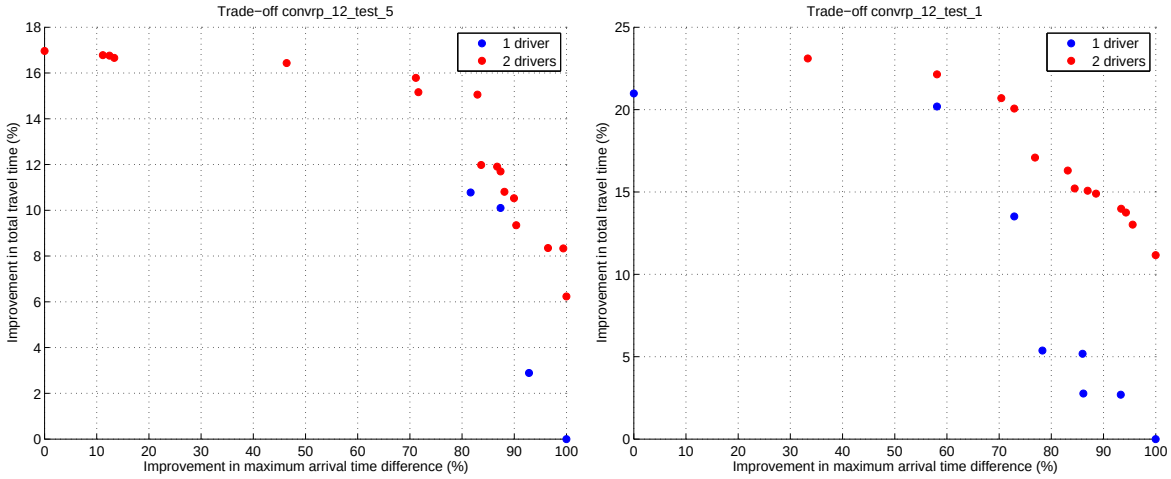


Figure 5.14 Two examples of the effect of districting on arrival time consistency when cost is the primary objective: Districting improves arrival time consistency implicitly in the left figure. In the right figure, arrival time consistency is worse with districting than without.

	C_{small}	C_{smallE}	$C_{0.5}$	$C_{0.7}$	$C_{0.9}$
# Worst l_{max}	6/10	0/5	1/12	0/12	0/12
Average Imp. l_{max} (%)	13.34	26.02	37.54	35.33	54.25
I'_ε	1.145	1.048	1.052	1.049	1.014

Table 5.12 Effect of strict driver consistency on arrival time consistency when travel cost is the primary objective. # Worst l_{max} is the number of instances in which l_{max} is larger with districting than without, Average Imp. l_{max} is the average improvement in l_{max} compared to the solution with minimal travel time and without districting, and I'_ε gives the factor by how much solutions deteriorate because of districting.

the average improvement is between 35.33% and 54.25%. The cost of districting, i.e., I'_ε, decreases with larger service frequency from 1.052 in data set $C_{0.5}$ to 1.014 in data sets $C_{0.9}$.

5.5 Summary

Multi-objective optimization is a suitable approach for investigating the trade-off between conflicting objectives. In this chapter, we introduced the multi-objective generalized consistent vehicle routing problem (MOGenConVRP); the objective functions minimize the total travel cost and improve driver consistency (i.e., the maximum number of different drivers a customer encounters) and arrival time consistency (i.e., the maximum difference between the earliest and the latest arrival time). Based on different unary quality indicators, we can

summarize that the proposed multi directional large neighborhood search (MDLNS) is an eligible algorithm for solving the MOGenConVRP. Compared to the proposed exact solution approaches, the heuristic provides good approximation sets in short amount of time. MDLNS is specialized in finding the best possible approximation of the optimal Pareto-front: we checked each solution generated during the search process whether or not it is dominated and we apply the LNS for a large number of iterations. The average computation time on the large instances (data set *C*) is 23.2 hours (the average number of elements in *P* is 88.5). Nevertheless, the MDLNS is applicable in the field subject to slight modifications, e.g., by checking only selected solutions for efficiency and by reducing the number of LNS and MDLNS iterations, respectively.

The trade-off between travel time, driver consistency, and arrival time difference depends significantly on the fluctuations in the demand: the lower the service frequency, the more costly it is to achieve service consistency. Adequate levels of arrival time consistency can be provided with modest increase in travel time. Compared to the minimal cost routing plan, the maximum arrival time difference can be reduced by 50% at the cost of 1.58% longer travel time, on average; improving arrival time consistency by 70% costs 3.84%. In many cases, the level of arrival time consistency affects the cost of driver consistency: Visiting each customer with the same driver when arrival time consistency is improved by 10% increases travel time between 2.08% and 4.61% compared to the solution with minimal travel time. Achieving perfect driver consistency when l_{max} is decreased by 90% increases travel time between 0.77% and 2.49%. Perfect driver consistency is significantly more expensive than visiting each customer with two different drivers: the routing cost increases by up to 5.1% in 50% of the solutions; visiting each customer by two drivers instead of three drivers increases cost by less than 0.9%. A strict driver consistency, as it is enforced, e.g., in districting applications, improves arrival time consistency as a side benefit. The average maximum arrival time difference decreases between 35.33% and 54.25% as a consequence of allowing only one driver per customer when travel cost is the primary objective.

Chapter 6

Conclusion

The positive impact of customer-first business strategies on profit was recognized in the marketing literature a long time ago. In vehicle routing, these strategies have rarely been studied compared to the traditional business strategies that focus on reducing cost. In routing, customer satisfaction is improved by providing consistent service over time. In numerous applications, customers want to be visited at similar times of the day by a familiar driver (e.g., small package shipping, home health care, and deliveries to restaurants). Service consistency was already addressed in the 1970's; yet, only in the last couple of years have authors investigated measurements that quantify the consistency of multi-period routing plans. In this book, we described early approaches that acknowledge service consistency as a side benefit. These approaches have been revisited in recent publications that target service consistency explicitly. We grouped these approaches into three categories: a priori approaches that derive daily routes from a set of preplanned routes, districting approaches that assign each customer to a fixed service territory serviced by a single driver, and approaches that ignore fluctuations in the demand but accept shortages and inventories at the customers.

Service consistency can be achieved in three dimensions: First, reducing variations in the arrival times at the customers. Second, visiting customers in regular time intervals with similar delivery quantities. Third, visiting customers with the same driver repeatedly (for consistency from the customer's point of view) or assigning drivers to the same customers or regions as often as possible in order to increase their familiarity (for consistency from the driver's point of view). We reviewed the relevant literature and classified recent papers according to the considered dimensions of consistency. For each dimension, we have described how the problems have been modeled and solved.

In response to the demand for higher service consistency, the consistent vehicle routing problem (ConVRP) combines traditional vehicle routing constraints with the requirements

for service consistency. The problem is motivated by applications in the small package shipping industry, in which customer satisfaction can be increased by providing driver and time consistent service. Based on the ConVRP, we proposed three model variants and performed extensive computational experiments.

In the ConVRP, vehicles have to leave the depot at time zero. In the first model variant, we allowed for a delay in the departure times. This realistic relaxation mitigated the conflict between total travel time and time consistency, i.e., while total cost increases sharply if the maximum arrival time difference constraint becomes very tight, this effect can be almost neutralized by adapting the starting times of the routes. Second, we presented an extension of the ConVRP that we call the generalized consistent vehicle routing problem (GenConVRP). It generalizes the ConVRP by allowing more than one driver per customer, by relaxing the constraint for the maximum arrival time difference, by allowing flexible vehicle departure times, and by incorporating AM/PM time windows for the customers. Third, we introduced the multi-objective generalized consistent vehicle routing problem (MOGenConVRP) that extends the GenConVRP; the objective functions minimize the total travel cost and improve driver consistency (i.e., the maximum number of different drivers a customer encounters) and arrival time consistency (i.e., the maximum difference between the earliest and the latest arrival time). Multi-objective optimization proved to be a suitable approach for investigating the trade-off between conflicting objectives.

For each ConVRP variant, we devised specialized solution algorithms. The ConVRP was solved by a solution approach called template-based adaptive large neighborhood search (TALNS). It embeds the principle of deriving a multi-day routing plan from a set of template routes into the adaptive large neighborhood search framework. Experimental comparisons with other algorithms proved the competitiveness of the TALNS in terms of solution quality and computation time. The TALNS provides slightly better results than the best competitor while being about eight times faster. The GenConVRP was solved by a large neighborhood search (LNS) algorithm. Many solution approaches for the ConVRP are based on the template concept. LNS deviates from this principle and generates solutions without using templates. Experiments on different ConVRP variants showed that, on average, the non-template-based LNS performs better than all published algorithms for the ConVRP. For the MOGenConVRP, we proposed two exact solution approaches and one heuristic. The exact approaches are based on the ε-constraint framework. The heuristic is a combination of the LNS for the generalized consistent vehicle routing problem and the multi directional local search framework; we refer to our heuristic as multi direction large neighborhood search (MDLNS). Solving the MOGenConVRP is challenging; therefore, we applied the exact so-

lution approaches only on small problem instances. The quality of the MDLNS is evaluated by comparing the set of approximate solutions to the set of efficient solutions. Based on five multi-objective quality indicators, we showed that the MDLNS provides good solution sets in a short amount of time.

We generated challenging test instances and analyzed the consequences of varying emphasis on time and driver consistency. The results highlighted interesting effects on the trade-off between service consistency and routing cost. High levels of time consistency can be provided with small increases in travel cost. A 70% better arrival time consistency is achieved by increasing travel time by not more than 3.84%, on average. The variation of the vehicle departure times necessary to improve arrival time consistency is modest. In terms of driver consistency, a one-to-one driver-to-customer assignment might be too restrictive for commercial applications. An important managerial implication comes from the observation that allowing more than one driver per customer can reduce the routing cost by up to 6.5%, on average. However, allowing more than two drivers per customer produces negligible cost savings. The average cost of visiting each customer with a single driver depends on the level of arrival time consistency: the larger the maximum arrival time difference, the higher the average cost of driver consistency, i.e., arrival time consistency and driver consistency tend to be non-conflicting. The results also show that arrival time consistency improves as a side effect of visiting each customer with a single driver. We also found that both, driver and time consistency, have a small impact on the fleet size, i.e., providing higher consistency requires only a minor increase in the staff size. Disregarding AM/PM time windows decreases cost by up to 14.62% when demand fluctuations are high and emphasis is put on service consistency. In general, the cost of providing consistent customer service depends on the level at which demand is fluctuating: the stronger the fluctuation, the higher the cost of providing service consistency.

There are many directions for future research with regard to modeling and solving routing problems in which consistency is important. Service quality is a tool for attracting and keeping profitable customers. A profitable customer generates long-term revenues that exceed the long-term cost of servicing him. Unprofitable customers can be turned into profitable ones by either increasing prices or reducing the effort of servicing them (Niraj et al. [128]). This service-quality-dependent and customer-specific pricing mechanism has not yet been addressed in the context of consistency in vehicle routing. Customer-specific pricing in home delivery services (e.g., drug and grocery delivery) is examined, e.g., in Agatz et al. [1, 2] and Campbell and Savelsbergh [21, 22]. Here, the vendor decides which customer orders to accept or reject and sets the delivery fee of accepted orders in real time in

order to maximize expected profit. The cost of a delivery depends on the customer's address and the required (and guaranteed) service time window, but also on the addresses and service time windows of already accepted customers. With regard to arrival time consistency, future work could incorporate stochastic travel times into the models. The challenge here is to design robust a priori routes that guarantee stable arrival times at the customer locations despite unforeseeable events such as traffic jams. There are few articles that apply mathematical programming approaches to problems with consistency goals. The development of exact solution approaches would contribute to the investigation of how input parameters, such as emphasis on consistency and demand fluctuations, affect routing cost. Another challenge for future research is trying to prove the optimality of Algorithm 3 for adjusting the vehicle departure times or demonstrate a counter-example.

Bibliography

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

[2] Agatz, N., Campbell, A. M., Fleischmann, M., van Nunen, J., and Savelsbergh, M. (2013). Revenue management opportunities for internet retailers. *Journal of Revenue & Pricing Management*, 12(2):128–138.

[3] Ageev, A. A. and Pyatkin, A. V. (2008). A 2-approximation algorithm for the metric 2-peripatetic salesman problem. In Kaklamanis, C. and Skutella, M., editors, *Approximation and Online Algorithms*, volume 4927 of *Lecture Notes in Computer Science*, pages 103–115. Springer Berlin Heidelberg.

[4] Beasley, J. E. (1984). Fixed routes. *The Journal of the Operational Research Society*, 35(1):49–55.

[5] Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., and Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13(6):4–23.

[6] Beltrami, E. J. and Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94.

[7] Bennett Milburn, A. (2012). Operations research applications in home healthcare. In Hall, R., editor, *Handbook of Healthcare System Scheduling*, volume 168 of *International Series in Operations Research & Management Science*, pages 281–302. Springer US.

[8] Benton, W. and Rossetti, M. D. (1992). The vehicle scheduling problem with intermittent customer demands. *Computers & Operations Research*, 19(6):521 – 531.

[9] Bertazzi, L., Paletta, G., and Speranza, M. G. (2002). Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132.

[10] Bertazzi, L., Savelsbergh, M., and Speranza, M. (2008). Inventory routing. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 49–72. Springer US.

[11] Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866 – 2890.

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

[13] Bertsimas, D. J., Jaillet, P., and Odoni, A. R. (1990). A priori optimization. *Operations Research*, 38(6):1019–1033.

[14] Bérubé, J.-F., Gendreau, M., and Potvin, J.-Y. (2009). An exact ε-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194(1):39–50.

[15] Bianchi, L., Gambardella, L., and Dorigo, M. (2002). An ant colony optimization approach to the probabilistic traveling salesman problem. In Guervós, J., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., and Fernández-Villacañas, J.-L., editors, *Parallel Problem Solving from Nature — PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 883–892. Springer Berlin Heidelberg.

[16] Borsani, V., Matta, A., Beschi, G., and Sommaruga, F. (2006). A home care scheduling model for human resources. In *Service Systems and Service Management, 2006 International Conference on*, volume 1, pages 449–454.

[17] Braun, M. L. and Buhmann, J. M. (2002). The noisy euclidean traveling salesman problem and learning. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 1, pages 251 – 258. MIT Press, Cambridge, MA.

[18] Bredström, D. and Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19 – 31.

[19] Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M. (1998). The inventory routing problem. In Crainic, T. and Laporte, G., editors, *Fleet Management and Logistics*, pages 95–113. Springer US.

[20] Campbell, A. and Thomas, B. (2008a). Challenges and advances in a priori routing. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 123–142. Springer US.

[21] Campbell, A. M. and Savelsbergh, M. (2006). Incentive schemes for attended home delivery services. *Transportation Science*, 40(3):327–341.

[22] Campbell, A. M. and Savelsbergh, M. W. P. (2005). Decision support for consumer direct grocery initiatives. *Transportation Science*, 39(3):313–327.

[23] Campbell, A. M. and Thomas, B. W. (2008b). Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42(1):1–21.

[24] Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, 63(1):2–15.

[25] Canadian Transportation & Logistics (2004). Award-winning suppliers special: Inside mackinnon transport. http://www.ctl.ca/news/award-winning-suppliers-special-inside-mackinnon-transport/1000018391/. Retrieved: October 4, 2013.

[26] Carlsson, J. G. (2012). Dividing a territory among several vehicles. *INFORMS Journal on Computing*, 24(4):565–577.

[27] Carlsson, J. G. and Delage, E. (2013). Robust partitioning for stochastic multivehicle routing. *Operations Research*, 61(3):727–744.

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

[29] Christofides, N. (1971). Fixed routes and areas for delivery operations. *International Journal of Physical Distribution & Logistics Management*, 1(2):87–92.

[30] Christofides, N. and Beasley, J. E. (1984). The period routing problem. *Networks*, 14(2):237–256.

[31] Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *OR*, 20(3):309–318.

[32] Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24(1):270 – 287.

[33] Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*. Forthcoming.

[34] Coelho, L. C. and Laporte, G. (2013a). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, 51(23-24):7156–7169.

[35] Coelho, L. C. and Laporte, G. (2013b). The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558 – 565.

[36] Coello Coello, C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308.

[37] Cordeau, J., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409.

[38] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001). VRP with time windows. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[39] Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46.

[40] Coyte, P. and McKeever, P. (2001). Home care in Canada: Passing the buck. *Canadian Journal of Nursing Research*, 33(2):11–26.

[41] Czyżak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47.

[42] Daganzo, C. F. (1984a). The distance traveled to visit *N* points with a maximum of *C* stops per vehicle: An analytic model and an application. *Transportation Science*, 18(4):331–350.

[43] Daganzo, C. F. (1984b). The length of tours in zones of different shapes. *Transportation Research Part B: Methodological*, 18(2):135 – 145.

[44] Day, J. M., Wright, P. D., Schoenherr, T., Venkataramanan, M., and Gaudette, K. (2009). Improving routing and scheduling decisions at a distributor of industrial gasses. *Omega*, 37(1):227 – 237.

[45] Dayarian, I., Crainic, T. G., Gendreau, M., and Rei, W. (2013a). An adaptive large neighborhood search heuristic for a multi-period vehicle routing problem. Technical Report CIRRELT-2013-60, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT).

[46] Dayarian, I., Crainic, T. G., Gendreau, M., and Rei, W. (2013b). A branch-and-price approach for a multi-period vehicle routing problem. Technical Report CIRRELT-2013-60, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT).

[47] De Kort, J. B. J. M. (1991). Lower bounds for symmetric *K*-peripatetic salesman problems. *Optimization*, 22(1):113–122.

[48] De Kort, J. B. J. M. (1993). A branch and bound algorithm for symmetric 2-peripatetic salesman problems. *European Journal of Operational Research*, 70(2):229 – 243.

[49] Doerner, K., Gutjahr, W., Hartl, R., Strauss, C., and Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1-4):79–99.

[50] Dohn, A., Kolind, E., and Clausen, J. (2009). The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4):1145 – 1157.

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

[52] Duchenne, E., Laporte, G., and Semet, F. (2005). Branch-and-cut algorithms for the undirected *m*-peripatetic salesman problem. *European Journal of Operational Research*, 162(3):700 – 712.

[53] Duchenne, E., Laporte, G., and Semet, F. (2007). The undirected *m*-peripatetic salesman problem: Polyhedral results and new algorithms. *Operations Research*, 55(5):949–965.

[54] Ehrgott, M. and Gandibleux, X. (2002). Multiobjective combinatorial optimization — theory, methodology, and applications. In Ehrgott, M. and Gandibleux, X., editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, volume 52 of *International Series in Operations Research & Management Science*, pages 369–444. Springer US.

[55] Ehrgott, M. and Ruzika, S. (2008). Improved ϵ-constraint method for multiobjective programming. *Journal of Optimization Theory and Applications*, 138(3):375–396.

[56] Ehrgott, M. and Wiecek, M. (2005). Mutiobjective programming. In Figueira, J., Greco, S., and Ehrgott, M., editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 667–708. Springer New York.

[57] Erera, A. L., Savelsbergh, M., and Uyar, E. (2009). Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. *Networks*, 54(4):270–283.

[58] Eveborn, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care – an operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962 – 976.

[59] Feillet, D., Garaix, T., Lehuédé, F., Péton, O., and Quadri, D. (2014). A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks*, 63(3):211–224.

[60] Fischetti, M., González, J. J. S., and Toth, P. (1995). Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*, pages 169 – 173.

[61] Fleischmann, B. and Paraschis, J. N. (1988). Solving a large scale districting problem: a case report. *Computers & Operations Research*, 15(6):521 – 533.

[62] Fonseca, C., Paquete, L., and López-Ibáñez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In *IEEE Congress on Evolutionary Computation, 2006*, pages 1157–1163.

[63] Fonseca, C. M. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16.

[64] Fonseca, C. M., López-Ibáñez, M., Paquete, L., and Guerreiro, A. P. (2010). Computation of the hypervolume indicator. http://iridia.ulb.ac.be/∼manuel/hypervolume. Retrieved: 8. May, 2014.

[65] Fornell, C. (1992). A national customer satisfaction barometer: The swedish experience. *Journal of Marketing*, 56(1):6–21.

[66] Francis, P., Smilowitz, K., and Tzur, M. (2006). The period vehicle routing problem with service choice. *Transportation Science*, 40(4):439–454.

[67] Francis, P., Smilowitz, K., and Tzur, M. (2007). Flexibility and complexity in periodic distribution problems. *Naval Research Logistics*, 54(2):136–150.

[68] Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 73–102. Springer US.

[69] Frey, R.-V., Bayón, T., and Totzek, D. (2013). How customer satisfaction affects employee satisfaction and retention in a professional services context. *Journal of Service Research*, 16(4):503–517.

[70] Gaudioso, M. and Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, 26(2):86–92.

[71] Gehring, H. and Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In Miettinen, K., Mäkelä, M., and Toivanen, J., editors, *Proceedings of EUROGEN99: Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pages 57–64. University of Jyväskylä.

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

[73] Gendreau, M., Laporte, G., and Séguin, R. (1996b). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477.

[74] Golden, B. L. and Stewart, W. (1978). Vehicle routing with probabilistic demands. In Hogben, D. and Gife, D. W., editors, *Computer Science and Statistics: Tenth Annual Symposium on the Interface, NBS Special Publication*, volume 503, pages 252–259.

[75] Groër, C., Golden, B., and Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643.

[76] Habenicht, W. (1983). Quad trees, a datastructure for discrete vector optimization problems. In Hansen, P., editor, *Essays and Surveys on Multiple Criteria Decision Making*, volume 209 of *Lecture Notes in Economics and Mathematical Systems*, pages 136–145. Springer Berlin Heidelberg.

[77] Haimes, Y., Lasdon, L. S., and Wismer, D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-1(3):296–297.

[78] Hapke, M., Jaszkiewicz, A., and Słowiński, R. (2000). Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. *Journal of Heuristics*, 6(3):329–345.

[79] Hardy, W. E. (1980). Vehicle routing efficiency – a comparison of districting analysis and the Clarke-Wright method. *American Journal of Agricultural Economics*, 62(3):534–536.

[80] Haughton, M. A. (1998). The performance of route modification and demand stabilization strategies in stochastic vehicle routing. *Transportation Research Part B: Methodological*, 32(8):551 – 566.

[81] Haughton, M. A. (2000). Quantifying the benefits of route reoptimisation under stochastic customer demands. *The Journal of the Operational Research Society*, 51(3):320–332.

[82] Haughton, M. A. (2002). Measuring and managing the learning requirements of route reoptimization on delivery vehicle drivers. *Journal of Business Logistics*, 23(2):45–66.

[83] Haughton, M. A. (2007). Assigning delivery routes to drivers under variable customer demands. *Transportation Research Part E: Logistics and Transportation Review*, 43(2):157 – 172.

[84] Haughton, M. A. (2008). The efficacy of exclusive territory assignments to delivery vehicle drivers. *European Journal of Operational Research*, 184(1):24 – 38.

[85] Haugland, D., Ho, S. C., and Laporte, G. (2007). Designing delivery districts for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 180(3):997 – 1010.

[86] Hemmelmayr, V., Doerner, K. F., Hartl, R. F., and Savelsbergh, M. W. (2009). Delivery strategies for blood products supplies. *OR Spectrum*, 31(4):707–725.

[87] Heskett, J., Jones, T., Loveman, G., Sasser, W. J., and Schlesinger, L. (1994). Putting the service-profit chain to work. *Harvard Business Review*, 72(2):164–174.

[88] Hollander, M., , Chappell, N., Havens, B., Mcwilliam, C., and Miller, J. A. (2002). *Study Of The Costs And Outcomes Of Home Care And Residential Long Term Care Services, A Report Prepared for the Health Transition Fund, Health Canada*. National Evaluation of the Cost-Effectiveness of Home Care.

[89] Homburg, C., Koschate, N., and Hoyer, W. D. (2005). Do satisfied customers really pay more? A study of the relationship between customer satisfaction and willingness to pay. *Journal of Marketing*, 69(2):84–96.

[90] Horn, J. (1997). Multicriterion decision making. In Back, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, volume 97. IOP Publishing Ltd.

[91] Ioachim, I., Desrosiers, J., Soumis, F., and Bélanger, N. (1999). Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119(1):75 – 90.

[92] Jaillet, P. (1988). A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936.

[93] Jones, D., Mirrazavi, S., and Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1 – 9.

[94] Jozefowiez, N., Laporte, G., and Semet, F. (2012). A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing*, 24(4):554–564.

[95] Jozefowiez, N., Semet, F., and Talbi, E.-G. (2002). Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In Guervós, J. J. M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., and Fernández-Villacañas, J.-L., editors, *Parallel Problem Solving from Nature — PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 271–280. Springer Berlin Heidelberg.

[96] Keith, R. J. (1960). The marketing revolution. *Journal of Marketing*, 24(3):35–38.

[97] Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

[98] Kirlik, G. and Sayın, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479 – 488.

[99] Knowles, J. and Corne, D. (2002). On metrics for comparing nondominated sets. In *Proceedings of the 2002 Congress on Evolutionary Computation*, volume 1, pages 711–716.

[100] Kohli, A. K. and Jaworski, B. J. (1990). Market orientation: The construct, research propositions, and managerial implications. *Journal of Marketing*, 54(2):1–18.

[101] Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992 – 1007.

[102] Kotler, P. (2003). *Marketing management*, volume 11. Pearson Education, Inc., Upper Saddle River, New Jersey.

[103] Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2014a). The generalized consistent vehicle routing problem. *Transportation Science*. Forthcoming.

[104] Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2014b). Vehicle routing problems in which consistency considerations are important: A survey. *Networks*. Forthcoming.

[105] Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600.

[106] Kovacs, A. A., Parragh, S. N., and Hartl, R. F. (2014c). The multi-objective generalized consistent vehicle routing problem. Technical report, Department of Business Administration, University of Vienna.

[107] Kovacs, A. A., Parragh, S. N., and Hartl, R. F. (2014d). A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks*, 63(1):60–81.

[108] Krarup, J. (1975). The peripatetic salesman and some related unsolved problems. In Roy, B., editor, *Combinatorial Programming: Methods and Applications*, volume 19 of *NATO Advanced Study Institutes Series*, pages 173–178. Springer Netherlands.

[109] Kruskal, J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

[110] Kunkel, M. and Schwind, M. (2012). Vehicle routing with driver learning for real world CEP problems. In *45th Hawaii International Conference on System Science (HICSS)*, pages 1315–1322.

[111] Laporte, G. and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, 31:147–184.

[112] Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932 – 942.

[113] Lei, H., Laporte, G., and Guo, B. (2012). Districting for routing with stochastic customers. *EURO Journal on Transportation and Logistics*, 1(1-2):67–85.

[114] Leitner, M., Ljubić, I., and Sinnl, M. (2013). The bi-objective prize-collecting steiner tree problem. Technical report, University of Vienna, Austria.

[115] Li, F., Golden, B., and Wasil, E. (2006). The noisy euclidean traveling salesman problem: A computational analysis. In Alt, F., Fu, M., and Golden, B., editors, *Perspectives in Operations Research*, volume 36 of *Operations Research/Computer Science Interfaces Series*, pages 247–270. Springer US.

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

[117] Marlin, P. G. (1981). Application of the transportation model to a large-scale "districting" problem. *Computers & Operations Research*, 8(2):83 – 96.

[118] Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., and Vitry, G. (2014). Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41(0):196 – 207.

[119] Morlok, E. K., Nitzberg, B. F., Balasubramaniam, K., and Sand, M. L. (2000). The parcel service industry in the U.S.: Its size and role in commerce. Technical report, Systems Engineering Department, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia.

[120] Mostaghim, S. and Teich, J. (2005). Quad-trees: A data structure for storing pareto sets in multiobjective evolutionary algorithms with elitism. In Abraham, A., Jain, L., and Goldberg, R., editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 81–104. Springer London.

[121] Muyldermans, L., Cattrysse, D., and Oudheusden, D. V. (2003). District design for arc-routing applications. *Journal of the Operational Research Society*, 54(11):1209 – 1221.

[122] Muyldermans, L., Cattrysse, D., Oudheusden, D. V., and Lotan, T. (2002). Districting for salt spreading operations. *European Journal of Operational Research*, 139(3):521 – 532.

[123] Naddef, D. and Rinaldi, G. (2001). Branch-and-cut algorithms for the capacitated VRP. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 1–26. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[124] Narver, J. C. and Slater, S. F. (1990). The effect of a market orientation on business profitability. *Journal of Marketing*, 54(4):20–35.

[125] Neumayer, P. and Schweigert, D. (1994). Three algorithms for bicriteria integer linear programs. *Operations-Research-Spektrum*, 16(4):267–276.

[126] Ngueveu, S., Prins, C., and Wolfler Calvo, R. (2010a). Lower and upper bounds for the *m*-peripatetic vehicle routing problem. *4OR*, 8(4):387–406.

[127] Ngueveu, S. U., Prins, C., and Wolfler Calvo, R. (2010b). A hybrid tabu search for the *m*-peripatetic vehicle routing problem. In Maniezzo, V., Stützle, T., and Voß, S., editors, *Matheuristics*, volume 10 of *Annals of Information Systems*, pages 253–266. Springer US.

[128] Niraj, R., Gupta, M., and Narasimhan, C. (2001). Customer profitability in a supply chain. *Journal of Marketing*, 65(3):1–16.

[129] Nowak, M., Hewitt, M., and Nataraj, N. (2013). Planning strategies for home health care delivery. Technical report, Loyola University Chicago, Information Systems and Operations Management.

[130] Paquete, L., Chiarandini, M., and Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'Kindt, V., editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–199. Springer Berlin Heidelberg.

[131] Paquette, J., Bellavance, F., Cordeau, J.-F., and Laporte, G. (2012). Measuring quality of service in dial-a-ride operations: the case of a Canadian city. *Transportation*, 39(3):539–564.

[132] Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2010). Demand responsive transportation. In Cochran, J. J., Cox, L. A., Keskinocak, P., Kharoufeh, J. P., and Smith, J. C., editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc.

[133] Parragh, S. N., Doerner, K. F., Hartl, R. F., and Gandibleux, X. (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, 54(4):227–242.

[134] Parragh, S. N. and Tricoire, F. (2014). Branch-and-bound for bi-objective optimization. Technical report, Department of Business Administration, University of Vienna, Austria.

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

[136] Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, volume 146, pages 399–419. Springer, New York.

[137] Potvin, J. and Rousseau, J. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340.

[138] Przybylski, A., Gandibleux, X., and Ehrgott, M. (2008). Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2):509 – 533.

[139] Przybylski, A., Gandibleux, X., and Ehrgott, M. (2010). A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149 – 165.

[140] Ralphs, T., Saltzman, M., and Wiecek, M. (2006). An improved algorithm for solving biobjective integer programs. *Annals of Operations Research*, 147(1):43–70.

[141] Reichheld, F. F. and Sasser, W. E. (1990). Zero defections: quality comes to services. *Harvard business review*, 68(5):105–111.

[142] Repoussis, P., Stavropoulou, F., Tarantilis, C., Gounaris, C., and Floudas, C. (2012). Collaborative template-based tabu search and branch-and-cut methods for the consistent vehicle routing problem. ODYSSEUS 2012, 5th International Workshop on Freight Transportation and Logistics, May 21-25, Mykonos, Greece.

[143] Riera-Ledesma, J. and Salazar-González, J. J. (2005). The biobjective travelling purchaser problem. *European Journal of Operational Research*, 160(3):599 – 613. Decision Analysis and Artificial Intelligence.

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

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

[146] Rudolph, G. (1998). On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *The 1998 IEEE International Conference on Evolutionary Computation Proceedings*, pages 511–516.

[147] Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, 9(1):1–17.

[148] Sarker, R. and Coello Coello, C. A. (2002). Assessment methodologies for multiobjective evolutionary algorithms. In *Evolutionary Optimization*, volume 48 of *International Series in Operations Research & Management Science*, pages 177–195. Springer US.

[149] Savelsbergh, M. W. P. and Goetschalckx, M. (1995). A comparison of the efficiency of fixed versus variable vehicle routes. *Journal of Business Logistics*, 16(1):163.

[150] Schilde, M., Doerner, K., Hartl, R., and Kiechle, G. (2009). Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201.

[151] Schneider, M., Doppstadt, C., Stenger, A., and Schwind, M. (2010). Ant colony optimization for a stochastic vehicle routing problem with driver learning. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8.

[152] Schneider, M., Stenger, A., Schwahn, F., and Vigo, D. (2014). Territory-based vehicle routing in the presence of time window constraints. *Transportation Science*. Forthcoming.

[153] Schneider, U., Österle, A., Schober, D., and Schober, C. (2006). Die Kosten der Pflege in Österreich. Technical report, Institute for Social Policy, Vienna University of Economics and Business.

[154] Schott, J. R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, USA.

[155] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171.

[156] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520, pages 417–431. Springer, Berlin Heidelberg.

[157] Smilowitz, K., Nowak, M., and Jiang, T. (2009). Workforce management in periodic delivery operations. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208-3119, U.S.A. Working Paper No. 09-004.

[158] Smilowitz, K., Nowak, M., and Jiang, T. (2013). Workforce management in periodic delivery operations. *Transportation Science*, 47(2):214–230.

[159] Spliet, R. (2013). *Vehicle Routing with Uncertain Demand*. PhD thesis, Erasmus University Rotterdam.

[160] Spliet, R. and Desaulniers, G. (2012). The discrete time window assignment vehicle routing problem. Technical Report G–2012–81, Les Cahiers du GERAD, Montreal, Canada.

[161] Spliet, R. and Gabor, A. F. (2012). The time window assignment vehicle routing problem. Technical Report EI2012_07, Econometric Institute, Erasmus School of Economics (ESE), Rotterdam, The Netherlands.

[162] Spliet, R., Gabor, A. F., and Dekker, R. (2014). The vehicle rescheduling problem. *Computers & Operations Research*, 43(1):129 – 136.

[163] Srinivasan, V. and Thompson, G. L. (1976). Algorithms for minimizing total cost, bottleneck time and bottleneck shipment in transportation problems. *Naval Research Logistics Quarterly*, 23(4):567–595.

[164] Steuer, R. E., Gardiner, L. R., and Gray, J. (1996). A bibliographic survey of the activities and international nature of multiple criteria decision making. *Journal of Multi-Criteria Decision Analysis*, 5(3):195–217.

[165] Storbacka, K., Strandvik, T., and Grönroos, C. (1994). Managing customer relationships for profit: the dynamics of relationship quality. *International journal of service industry management*, 5(5):21–38.

[166] Sun, M. and Steuer, R. E. (1996). Quad-trees and linear lists for identifying nondominated criterion vectors. *INFORMS Journal on Computing*, 8(4):367–375.

[167] Sungur, I., Ren, Y., Ordóñez, F., Dessouky, M., and Zhong, H. (2010). A model and algorithm for the courier delivery problem with uncertainty. *Transportation Science*, 44(2):193–205.

[168] Sylva, J. and Crema, A. (2004). A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46 – 55.

[169] Tamaki, H., Kita, H., and Kobayashi, S. (1996). Multi-objective optimization by genetic algorithms: a review. In Fukuda, T. and Furuhashi, T., editors, *Proceedings of IEEE International Conference on Evolutionary Computation, 1996*, pages 517–522.

[170] Tarantilis, C., Stavropoulou, F., and Repoussis, P. (2012). A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications*, 39(4):4233 – 4239.

[171] The Wall Street Journal (2013). A new threat to UPS and FedEx. http://online.wsj.com/news/article_email/SB10001424052702304773104579266682206635994-lMyQjAxMTAzMDIwMDEyNDAyWj. Retrieved: December 20, 2013.

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

[173] Toth, P. and Vigo, D. (2001). An overview of vehicle routing problems. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 1–26. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[174] Tricoire, F. (2012). Multi-directional local search. *Computers & Operations Research*, 39(12):3089 – 3101.

[175] Van Oudheusden, D., Cattrysse, D., and Lotan, T. (1999). On the importance of districting and its potential impact on routing. In *World Transport Research: Selected Proceedings of the 8th World Conference on Transport Research*, number Volume 2, pages 521–531. Elsevier Science Publishing Company, New York, NY 10159 USA.

[176] Van Veldhuizen, D. and Lamont, G. (2000). On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 204–211 vol.1.

[177] Van Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. PhD thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, USA.

[178] Van Veldhuizen, D. A. and Lamont, G. B. (1998). Evolutionary computation and convergence to a pareto front. In Koza, J. R. e. a., editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228. University of Wisconsin, Madison, USA.

[179] Van Veldhuizen, D. A. and Lamont, G. B. (1999). Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, New York, NY, USA.

[180] Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., and Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research*, 40(1):498 – 509.

[181] Visée, M., Teghem, J., Pirlot, M., and Ulungu, E. (1998). Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem. *Journal of Global Optimization*, 12(2):139–155.

[182] Waters, C. D. J. (1989). Vehicle-scheduling problems with uncertainty and omitted customers. *The Journal of the Operational Research Society*, 40(12):1099–1108.

[183] Wolfler Calvo, R. and Cordone, R. (2003). A heuristic approach to the overnight security service problem. *Computers & Operations Research*, 30(9):1269 – 1287.

[184] Wong, K. and Beasley, J. (1984). Vehicle routing using fixed delivery areas. *Omega*, 12(6):591 – 600.

[185] Wong, R. (2008). Vehicle routing for small package delivery and pickup services. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 475–485. Springer, New York.

[186] Woodward, C., Abelson, J., Tedford, S., and Hutchison, B. (2004). What is important to continuity in home care? Perspectives of key stakeholders. *Social Science & Medicine*, 58(1):177 – 192.

[187] Yan, S., Wang, S.-S., and Chang, Y.-H. (2014). Cash transportation vehicle routing and scheduling under stochastic travel times. *Engineering Optimization*, 46(3):289–307.

[188] Yan, S., Wang, S.-S., and Wu, M.-W. (2012). A model with a solution algorithm for the cash transportation vehicle routing and scheduling problem. *Computers & Industrial Engineering*, 63(2):464 – 473.

[189] Yang, Y., Wu, J., Sun, X., Wu, J., and Zheng, C. (2013). A niched pareto tabu search for multi-objective optimal design of groundwater remediation systems. *Journal of Hydrology*, 490(1):56 – 73.

[190] Zhong, H., Hall, R. W., and Dessouky, M. (2007). Territory planning and vehicle dispatching with driver learning. *Transportation Science*, 41(1):74–89.

[191] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

[192] Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms — a comparative case study. In Eiben, A., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin Heidelberg.

[193] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.

Abstract

In vehicle routing, customer satisfaction is often a result of consistent service. Customers appreciate service at regular times of the day provided by the same driver each time. Additionally, drivers become more familiar with their tasks if they visit the same customers and service regions repeatedly. In this thesis, we first survey literature that addresses service consistency in vehicle routing. We present early solution approaches, starting from the 1970's, that focus on reducing the operational complexity resulting from planning and executing new routes each day. One side benefit of these approaches is service consistency; therefore, many recent solution approaches devised for improving customer satisfaction are based on previous achievements. We classify the literature according to three consistency features: arrival time consistency, person-oriented consistency, and delivery consistency. For each feature, we survey different modeling concepts and measurements, demonstrate solution approaches, and examine the cost of improving service consistency.

The consistent vehicle routing problem (ConVRP) combines traditional vehicle routing constraints with the requirements for service consistency. Based on the ConVRP, we introduce three new models in order to examine the trade-off between service consistency and routing cost. First, we present a relaxed variant of the original ConVRP where the departure times from the depot can be delayed to adjust the arrival times at customers. Flexible departure times considerably improve the solution quality under tight consistency requirements. The ConVRP takes customer satisfaction into account by assigning one driver to a customer and by bounding the variation in the arrival times over a given planning horizon. These requirements may be too restrictive in some applications. In our second ConVRP variant, each customer is visited by a limited number of drivers and the variation in the arrival times is penalized in the objective function. The vehicle departure times may be adjusted to obtain stable arrival times. Additionally, customers are associated with AM/PM time windows. The problem is referred to as the generalized ConVRP (GenConVRP). Third, we present the multi-objective GenConVRP (MOGenConVRP) with three independent objective functions: travel cost, arrival time consistency, and driver consistency. Multi-objective

optimization is a flexible approach for examining the trade-off between conflicting objective functions.

For each problem variant, we devise specialized solution algorithms. The most prominent approach for solving routing problems with consistency considerations is the template method. A set of artificial routes (referred to as template routes) is generated in advance and the pre-planned routes are used as a basis for the daily routes. We present a fast solution method called template-based adaptive large neighborhood search. Compared to other template-based heuristics, the developed algorithm is highly competitive on the available benchmark instances. Additionally, we present a solution approach that does not use template routes to generate routing plans. The new approach is based on a large neighborhood search (LNS) that is applied to the entire solution. Arrival time consistency is improved by a simple 2-opt operator that reverses parts of particular routes. The proposed algorithm performs well on different variants of the ConVRP; it outperforms template-based approaches in terms of travel cost and time consistency. For the MOGenConVRP, we embed the LNS into the multi directional local search framework. Small instances are solved to optimality by two algorithms that are based on the ε-constraint method. Furthermore, we propose a greedy algorithm for adjusting the vehicle departure times from the depot such that the variation in the arrival times at the customers is minimized.

We perform experiments on the benchmark instances for the ConVRP and on new instances generated for the generalized problem. Decreasing variations in the arrival times, in order to improve customer satisfaction, is recommended to service providers because arrival time consistency can be improved significantly with modest increases in travel cost and fleet size. The variation of the vehicle departure times necessary to improve arrival time consistency is minor. The average cost of visiting each customer with a separate driver depends on the level of arrival time consistency: The larger the maximum arrival time difference the higher the average cost of driver consistency, i.e., arrival time consistency and driver consistency tend to be non-conflicting. Remarkable cost savings can be obtained by allowing more than one driver per customer. In general, we can summarize that the larger the variations in the customer demands, the higher the cost of service consistency.

Zusammenfassung

In der Tourenplanung wird Servicekonsistenz oft als Mittel zur Steigerung der Kundenzufriedenheit eingesetzt. Ein Service, das jeden Tag zur gleichen Zeit stattfindet und von einem bekannten Fahrer durchgeführt wird, wird von Kunden hoch geschätzt. Zusätzlich arbeiten Fahrer effizienter, wenn sie immer wieder die gleichen Kunden in den gleichen Servicezonen besuchen. Diese Dissertation beginnt mit einem Literaturüberblick über Arbeiten, die sich mit Servicekonsistenz in der Tourenplanung beschäftigen. Ausgehend von den 1970ern präsentieren wir frühe Lösungsansätze, die zum Ziel haben die Komplexität, die durch tägliche Neuplanung von Touren entsteht, zu reduzieren. Ein positiver Nebeneffekt dieser Ansätze ist, dass sie die Konsistenz von Tourenplänen fördern. Aus diesem Grund basieren viele moderne Lösungsansätze für die Verbesserung der Servicekonsistenz auf älteren Ideen. Wir klassifizieren Arbeiten anhand von drei Konsistenzmerkmalen: Konsistenz in den Besuchszeiten, Fahrerkonsistenz und Konsistenz in der Liefermenge. Für jedes Merkmal untersuchen wir unterschiedliche Modellierungskonzepte, präsentieren Lösungsansätze und ermessen die Kosten, die durch höhere Konsistenz anfallen.

Das Consistent Vehicle Routing Problem (ConVRP) verbindet die traditionellen Anforderungen der Tourenplanung mit dem Bedarf nach Servicekonsistenz. Ausgehend von dem ConVRP präsentieren wir drei neue Modelle, anhand derer wir den Zielkonflikt zwischen Servicekonsistenz und Fahrtkosten untersuchen. Als Erstes präsentieren wir eine gelockerte Variante des ConVRPs, bei der die Abfahrtszeiten vom Depot frei gewählt werden dürfen, um die Ankunftszeiten bei den Kunden anzupassen. Flexible Abfahrtszeiten tragen wesentlich zu einer besseren Lösung bei, wenn die Anforderung an die zeitliche Konsistenz hoch ist. Im ConVRP wird Kundenzufriedenheit dadurch erreicht, dass ein Kunde von maximal einem Fahrer besucht wird und die maximale Abweichung zwischen den Besuchszeiten beschränkt ist. In unserer zweiten ConVRP-Variante darf jeder Kunde von mehreren Fahrern besucht werden und die Abweichung in den Besuchszeiten wird in der Zielfunktion pönalisiert. Abfahrtszeiten vom Depot sind flexibel, um die zeitliche Konsistenz zu verbessern. Außerdem dürfen Kunden jeweils nur am Vormittag oder nur am Nachmittag besucht wer-

den. Diese ConVRP-Variante wird Generalized ConVRP (GenConVRP) genannt. Drittens präsentieren wir das Multi-Objective GenConVRP (MOGenConVRP) mit drei eigenständigen Zielfunktionen: Fahrtkosten, zeitliche Konsistenz und Fahrerkonsistenz. Mehrzieloptimierung ist ein eleganter Ansatz, um den Konflikt zwischen unterschiedlichen Zielfunktionen zu untersuchen.

Für jede Problemvariante entwickeln wir spezialisierte Algorithmen. Der bekannteste Ansatz, um hohe Servicekonsistenz in der Tourenplanung zu erreichen, ist die Schablonen-Methode. Dabei wird im Vorfeld ein künstlicher Tourenplan generiert (Schablone genannt) und die im Voraus geplanten Touren werden als Ausgangspunkt für die täglichen Pläne verwendet. Wir präsentieren eine schnelle Schablonen-basierte Lösungsmethode, die Template-Based Adaptive Large Neighborhood Search (TALNS) genannt wird. Im Vergleich zu anderen Schablonen-basierten Heuristiken ist unser Algorithmus äußerst kompetitiv. Zusätzlich präsentieren wir einen Lösungsansatz, der ohne Schablone auskommt. Der neue Ansatz basiert auf dem Large Neighborhood Search (LNS), der auf den gesamten Tourenplan angewendet wird. Die Konsistenz in den Besuchszeiten wird durch einen einfachen 2-Opt Operator verbessert, der Teile von bestimmten Routen umkehrt. Der LNS Algorithmus eignet sich gut für unterschiedliche Varianten des ConVRPs, und die Lösungen übertreffen die Lösungen von Schablonen-basierten Ansätzen im Bezug auf Fahrtkosten und Konsistenz in den Besuchszeiten. Für das MOGenConVRP kombinieren wir den LNS Algorithmus mit dem Multi Directional Local Search Framework. Kleine Testinstanzen lösen wir optimal mit Hilfe von zwei auf der ε-Constraint Methode basierenden Ansätzen.

Für unsere Experimente verwenden wir Benchmark-Instanzen für das ConVRP und neue Instanzen für das generische Problem. Es ist empfehlenswert Kundenzufriedenheit durch geringe Abweichungen in den Ankunftszeiten zu erhöhen, da die Konsistenz in den Besuchszeiten bereits mit minimalem Kostenaufwand und mit minimaler Vergrößerung der Fahrzeugflotte deutlich verbessert werden kann. Die Variation in den Abfahrtszeiten vom Depot, die notwendig ist um die zeitliche Konsistenz zu verbessern, ist minimal. Die durchschnittlichen Kosten, um jeden Kunden mit einem Fahrer zu besuchen, hängen von der Konsistenz in den Besuchszeiten ab: Je höher die Abweichung in den Besuchszeiten, desto höher sind die durchschnittlichen Kosten für Fahrerkonsistenz. D.h. Fahrerkonsistenz und zeitliche Konsistenz sind in vielen Fällen konfliktfreie Zielfunktionen. Die Kosten können erheblich gesenkt werden, wenn jeder Kunde von mehr als einem Fahrer besucht werden darf. Allgemein gilt, dass die Kosten für Servicekonsistenz mit der Schwankung in der täglichen Nachfrage steigen.

Curriculum vitae

Personal data

Name	Attila A. Kovacs
Date of birth	September 14, 1983
Citizenship	Austria

Education

since 2010	PhD Program in Logistics and Operations Management, University of Vienna, Austria.
since 2010	Bachelor Program in Computer Science, University of Vienna, Austria.
2003–2009	Master Program in International Business Administration, University of Vienna, Austria.
09/2008-02/2009	Erasmus Student Exchange Program, Universidad de Las Palmas de Gran Canaria, Spain.

Experience

since 2010	Research Assistant, Chair of Production and Operations Management, University of Vienna, Austria.
02/2012-05/2012	Guest Researcher, University of Maryland, College Park, MD, USA.
2009-2010	Teaching Assistant, Chair of Production and Operations Management, University of Vienna, Austria.

Publications

2014 Kovacs, A.A., Hartl, R.F., Parragh, S.N., The multi-objective generalized consistent vehicle routing problem. Working paper.

2014 Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., Vehicle routing problems in which consistency considerations are important: A survey. Networks, forthcoming.

2014 Kovacs, A.A., Parragh, S.N., Hartl, R.F., A template based adaptive large neighborhood search for the consistent vehicle routing problem. Networks, 63(1): 60–81.

2014 Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., The generalized consistent vehicle routing problem. Transportation Science, forthcoming.

2013 Golden, B.L., Kovacs, A.A., Wasil, E.A., Vehicle routing applications in disaster relief. In P. Toth and D. Vigo (eds.), Vehicle routing: Problems, methods, and applications; MOS-SIAM Series on Optimization, SIAM, Philadelphia, forthcoming.

2012 Kovacs, A.A., Parragh, S.N., Doerner, K.F., Hartl, R.F., Adaptive large neighborhood search for service technician routing and scheduling problems. Journal of Scheduling 15 (5), 579-600.

Presentations

2013 Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., The generalized consistent vehicle routing problem. MIC 2013, 5 – 8 August, Singapore.

2011 Kovacs, A.A., Parragh, S.N., Hartl, R.F., A template based adaptive large neighborhood search for the consistent vehicle routing problem. MIC 2011, 25 – 28 July, Udine, Italy.

2011 Kovacs, A.A., Parragh, S.N., Hartl, R.F., A template based adaptive large neighborhood search for the consistent vehicle routing problem, ROUTE 2011, 31 May – 3 June, Sitges, Spain.

Awards

2010 Thesis Award of the Faculty of Business, Economics and Statistics, University of Vienna.

Computer skills

Languages	C++, Java, Python
Operating Systems	Ubuntu, Microsoft Windows
Tools	Eclipse, Latex, Libre Office, Matlab, Maxima, MS Office

Languages

German	Native speaker
Hungarian	Native speaker
English	Advanced
Spanish	Intermediate