Stochastic Vehicle Routing

From Theory to Practice

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Dennis Weyland

under the supervision of
Luca Maria Gambardella and Roberto Montemanni

July 2013

Dissertation Committee

Evanthia Papadopoulou	Università della Svizzera Italiana, Switzerland
Fabian Kuhn	Università della Svizzera Italiana, Switzerland
Richard Hartl	University of Vienna, Austria
Arne Løkketangen	Molde University College, Norway

Dissertation accepted on 29 July 2013

Research Advisor

Luca Maria Gambardella

Co-Advisor

Roberto Montemanni

PhD Program Director

Antonio Carzaniga

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Dennis Weyland
Lugano, 29 July 2013

In memory of Arne Løkketangen.

Abstract

In this thesis we discuss practical and theoretical aspects of various stochastic vehicle routing problems. These are combinatorial optimization problems related to the field of transportation and logistics in which input data is (partially) represented in a stochastic way. More in detail, we focus on *two-stage stochastic vehicle routing problems* and in particular on so-called *a priori optimization problems*. The results are divided into a theoretical part and a practical part. In fact, the theoretical results provide a strong motivation for the development and the usage of the methods presented in the practical part.

We begin the theoretical part with a convergence result regarding vehicle routing problems with stochastic demands. This result can be used to give explanations for some phenomena related to these problems which have been reported in literature. We then continue with hardness results for stochastic vehicle routing problems on substantially stochastic instances. Here we show that several stochastic vehicle routing problems remain NP-hard even if they are restricted to instances which differ significantly from non-stochastic instances. Additionally, we give some inapproximability results for these problems restricted to substantially stochastic instances. After that, we focus on a stochastic vehicle routing problem which considers time dependencies in terms of deadlines. We show that various computational tasks related to this problem, including the evaluation of the objective function, are #P-hard even for Euclidean instances. Note that this is a very strong hardness result and it immediately implies that these computational tasks are also NP-hard. We then further investigate the objective function of this problem. Here we demonstrate that the existing approximations for this objective function are not able to guarantee any reasonable worst-case approximation ratio. Finally, we show that it is NP-hard to approximate the objective function of a slightly more general problem within any reasonable worst-case approximation ratio.

In the practical part we develop and apply various methods for the optimization of stochastic vehicle routing problems. Since the theoretical results indicate that it is a great challenge to optimize these problems, we focus mainly

on heuristic methods. We start with the development of strong local search algorithms for one of the most extensively studied stochastic vehicle routing problems. These algorithms use an efficient approximation of the objective function based on Monte Carlo sampling. They are then further used within different heuristics, leading to new state-of-the-art methods for this problem. We then transfer our results to a more intricate stochastic vehicle routing problem. Here we first present an approximation of the objective function using the novel method of quasi-parallel evaluation of samples. Then we again develop strong local search algorithms and use them within more complex heuristics to obtain new state-of-the-art methods. After that we change the scope towards a general framework for the optimization of stochastic vehicle routing problems based on general purpose computing on graphics processing units. Here we are exploiting the massive computational power for parallel computations offered by modern graphics processing units in the context of stochastic vehicle routing problems. More in detail, we propose to use an approximation of the objective function based on Monte Carlo sampling which can be parallelized in an extremely efficient way. The effectiveness of this framework is then demonstrated in a case study. We finish the practical part with an application of our methods to a real world stochastic vehicle routing problem. This problem is part of a project that has been initiated in 2010 by Caritas Suisse. It is still in an early stage, but with our work we were able to successfully support some of the decision processes at this stage.

Acknowledgements

At this point I want to thank everyone who helped me in the last years to make this document possible. Shame on me for those people that I forgot to mention.

First of all, let me give credits to my parents *Ernst-Albert and Maria Weyland* and to my sister *Julia Weyland*. You have constantly supported me for my whole life and there are no words that could acknowledge your endless efforts in a proper way. At the end it is you who made the largest contribution to this document. Thank you for all your care and for affording me the education I got: at home, at school, and at university.

Then I want to acknowledge the support and help from my research advisor *Luca Maria Gambardella*, my co-advisor *Roberto Montemanni* and also my former supervisor *Leonora Bianchi*, all from the Dalle Molle Institute for Artificial Intelligence (IDSIA), Switzerland. Your efforts and encouragements helped me to arrive at this point.

Let me continue to thank my dissertation committee (in alphabetic order): *Richard Hartl* from the University of Vienna, Austria, *Fabian Kuhn* from the Università della Svizzera Italiana (USI), Switzerland, *Arne Løkketangen* from the Molde University College, Norway, and *Evanthia Papadopoulou* from the Università della Svizzera Italiana (USI), Switzerland.

Many thanks are given to the different PhD Program Directors at the Università della Svizzera Italiana (USI), Switzerland, during the last years (in alphabetic order): *Antonio Carzaniga*, *Fabio Crestani*, and *Michele Lanza*.

Let me also thank all my colleagues at the Dalle Molle Institute for Artificial Intelligence (IDSIA), Switzerland, the Scuola Universitaria Professionale della Svizzera Italiana (SUPSI), Switzerland, and the Università della Svizzera Italiana (USI), Switzerland. I had a great time with you guys and you provided me a very nice working environment.

At this point I have to apologize to a very special person in my life, *Michela Papandrea*. I am sorry for all the time I could not spend with you while I was working on this document. Thanks a lot for your patience, your support and your love.

Finally, I would like to give thanks to some persons that contributed to my research in one way or another (again in alphabetic order): *David Adjiashvili*, *Reinhard Bürgy*, *Cassio de Campos*, *Dan Ciresan*, *Patrick Czaplicki*, *Peter Detzner*, *Andrei Duma*, *Michael Felten*, *Alexander and Anna Förster*, *Kail Frank*, *Fred Glover*, *Jan Hofeditz*, *Jan Koutnik*, *Juxi Leitner*, *Jonathan Masci*, *Ueli Meier*, *Nikos Mutsanas*, *David Pritchard*, *Rudolf Scharlau*, *Kaspar Schüpbach*, *Georgios Stamoulis*, *Ola Svensson*, *Alex Tomic*, *Marc Uldry*, *Paul Wojtarowicz*, and *Akira Yokokawa*.

Contents

Chapter 1

Introduction

In this introductory chapter we provide the basic knowledge and the background information that are required in the remaining part of this thesis. We first put our research work into a broader context. We start with a discussion of *optimization under uncertainty* (Diwekar [2008]) and then we gradually confine the classification of our work within the fields of *stochastic combinatorial optimization* (Bianchi et al. [2009]), *stochastic vehicle routing* (Gendreau et al. [1996a]), *two-stage stochastic combinatorial optimization* (Schultz et al. [2008]) and finally *a priori optimization* (Bertsimas et al. [1990]). We continue with a discussion of four stochastic vehicle routing problems that are used throughout the thesis. We motivate these problems, we present related literature and we give formal definitions. After that we finish the introductory part with a brief outlook on the following chapters.

1.1 Classification of the Research Area

In the last decades *optimization under uncertainty* (Diwekar [2008]; Sahinidis [2004]; Freund [2004]; Gutjahr [2004]) has received increasing attention. This field deals with combinatorial optimization problems that consider uncertainty of the given information directly in the problem definition. In this way real world problems can be modeled in a more realistic way. Since we are confronted with uncertain information in many aspects of our lives, it is not surprising that *optimization under uncertainty* has plenty of applications in numerous areas. Among them are for example the generation of electrical power (Dentcheva and Römisch [1998]; Nowak and Römisch [2000]), the operation of water reservoirs (Cervellera et al. [2006]; Karamouz and Vasiliadis [1992]; Stedinger et al. [1984]), applications related to inventory management (Por-

teus [1990]; Hvattum et al. [2009]; You and Grossmann [2008]; Zheng [1992]; Hariga and Ben-Daya [1999]), portfolio selection (Samuelson [1969]; Inuiguchi and Ramık [2000]; Zhou and Li [2000]; Liu [1999]), facility planning (Chen et al. [2006]; Ermoliev and Leonardi [1982]; Louveaux and Peeters [1992]; Chang et al. [2007]), pollution control (Wong and Somes [1995]; Adar and Griffin [1976]; Horan [2001]), stabilization of mechanisms (Wang et al. [2002, 2010]; Lu et al. [2009]), analysis of biological systems (Frank et al. [2003]; Isukapalli et al. [1998]; Wilkinson [2009]), network design problems (Hoff et al. [2010]), scheduling problems (Almeder and Hartl [2012]) and applications in the field of transportation and logistics (Powell and Topaloglu [2003]; Shu et al. [2005]; Cooper and Leblanc [1977]; Barbarosoğlu and Arda [2004]). While on the one hand such problems based on more realistic models can be used to obtain more meaningful results, these problems are on the other hand usually much harder to solve than the non-stochastic counterparts. Therefore, it is of great importance to develop efficient methods for solving such problems.

There exist different ways in which the uncertainty can be modeled. One possibility is to provide possible values for the input data instead of only a single value, for example in terms of intervals. It is assumed that the real data is among these values, but no further knowledge about the likelihood of the different values is given. This leads to the field of *robust optimization* (Ben-Tal and Nemirovski [2002]; Beyer and Sendhoff [2007]; Ben-Tal et al. [2009]). Here the task is to find solutions with a certain robustness against the uncertainty. Another possibility is to use fuzzy variables to express uncertainty. Problems using such fuzzy variables for the input belong to the field of *fuzzy optimization* (Negoita and Ralescu [1977]; Delgado et al. [1994]; Luhandjula and Gupta [1996]). One different possibility is to represent the uncertainty using stochastic data, for example by means of probability distributions. This field is called *stochastic combinatorial optimization* (Gutjahr [2003]; Hentenryck and Bent [2009]; Immorlica et al. [2004]; Carraway et al. [1989]; Bianchi et al. [2009]). By using probability distributions, we somehow specify possible values for the different input data, like also for *robust optimization*. Here the main difference is that we additionally assume to have knowledge about the likelihood of these different values. In most of the cases the optimization goal is then to optimize a certain stochastic value, for example the expected costs of a solution, with respect to the given probability distributions.

In this thesis we focus on *stochastic combinatorial optimization problems*. More in detail, we focus on *stochastic vehicle routing problems* (Gendreau et al. [1996a]; Stewart and Golden [1983]; Hemmelmayr et al. [2009]; Dror and Trudeau [1986]; Kenyon and Morton [2003]; Yang et al. [2000]; Bertsimas et al.

[1995]; Gendreau et al. [1996b]; Bertsimas [1992]; Bastian and Rinnooy Kan [1992]; Gendreau et al. [1995]; Hvattum et al. [2006]; Secomandi [2001]; Liu and Lai [2004]; Schilde et al. [2011]). These are *stochastic combinatorial optimization problems* arising in the field of transportation and logistics. While in general uncertainty can be modeled in many different ways, in the field of transportation and logistics it is reasonable to model the uncertainty of several aspects in a stochastic way. For example, probability distributions for travel times, for customers' demands and for the presence of customers can be obtained based on historical data.

Many such *stochastic vehicle routing problems* are modeled as so called *two-stage stochastic combinatorial optimization problems* (Schultz et al. [2008]; Carøe and Tind [1998]; Klein Haneveld and van der Vlerk [1999]; Dhamdhere et al. [2005]; Uryasev and Pardalos [2001]). Here the idea is to make an initial decision at the first stage, where the stochastic information is available without knowing the actual realizations of the stochastic events. After the realizations of the stochastic events become known, a second stage decision is taken. This decision is based on the first stage decision and on the realizations of the stochastic events. This framework incorporates many different settings. One extreme case is to omit the first stage completely and to perform the actual optimization just after the realizations of the stochastic events become known. Such an approach is called a *reoptimization approach* (Secomandi and Margot [2009]; Wu et al. [2002]; Böckenhauer et al. [2008]; Delage [2010]). Although the results of this approach are usually of very high quality, it cannot be applied in a lot of situations. The main problem is that this approach requires certain computational resources, in particular computational time, which is usually not available between the realizations of the stochastic events become known and the final decision has to be taken. The other problem is that for some problems the actual realizations of the stochastic events are gradually revealed. While it is still possible to model such a problem as a *two-stage stochastic combinatorial optimization problem*, a *reoptimization approach* cannot be applied here. The other extreme case is to shift the main decision process to the first stage and to use the second stage decision only as a mechanism to assure feasibility of solutions. Since the actual optimization is performed before the realizations of the random events are revealed, this approach is called *a priori optimization* (Bertsimas et al. [1990]; Laporte et al. [1994]; Murat and Paschos [2002]; Miller-Hooks and Mahmassani [2003]; Campbell and Thomas [2008a]). The quality of solutions computed by an *a priori optimization* approach are usually worse compared to a *reoptimization approach*. Nonetheless, in many settings the constraints on the computational resources for the second stage, in particular computation time,

make an *a priori optimization* approach necessary. And as we will see later in this section, there are also other reasons why such an approach might be used. In between these two extreme cases other approaches, weighting the first stage and the second stage in different ways, can be found (Birge and Louveaux [1988]; Barbarosoğlu and Arda [2004]; Huang and Loucks [2000]; Cheung and Chen [1998]).

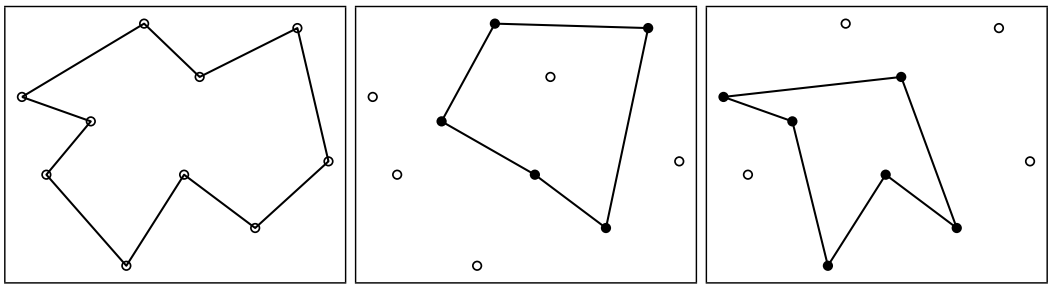
In this work we focus on *a priori optimization*. Although the quality of the final solutions is usually better using a *reoptimization approach*, an *a priori optimization* approach is the method of choice in many different situations. In the context of stochastic vehicle routing there are three main reasons that are in favor of an *a priori optimization* approach. The first one is that the second stage decision requires only a minimum of computational resources. If every day a new solution is required, the computational resources and in particular the computational time are usually a limiting factor. The second reason is that in some situations the realizations of the stochastic events are not revealed at once, but only step by step. Still these problems can be modeled as a *two-stage stochastic combinatorial optimization problems*, but it is not possible to apply a *reoptimization approach*. An example here is the collection of waste (Nuortio et al. [2006]; Maqsood and Huang [2003]). Although it is possible to retrieve probability distributions for the amount of waste that has to be collected, the real amount is only revealed during the collection process. Therefore, it is not possible to apply a *reoptimization approach*, while a proper *a priori optimization* approach can be used. The last reason holds especially for the class of *stochastic vehicle routing problems*. In fact, in the context of *stochastic vehicle routing problems* it is very convenient for the customers to be served periodically at roughly the same time. Additionally, it is convenient for the drivers of the vehicles to follow roughly the same route every time. Obviously, a *reoptimization approach* cannot guarantee these properties in general. On the other hand, many *a priori optimization* approaches maintain this property in a natural way. All in all, *a priori optimization* is the method of choice in many different situations. The usage of *a priori optimization* is very well motivated and has numerous real world applications.

1.2 The Main Optimization Problems used in this Thesis

In this section we discuss those stochastic vehicle routing problems that are the main subjects of investigation in this thesis. Since these problems are used throughout the thesis we decided to examine them already at this point. For all of the four problems presented in this section we give a motivation, we discuss related literature and we present a formal definition. To maintain consistency we refer to these problems with the names commonly used in literature. We begin with the well studied PROBABILISTIC TRAVELING SALESMAN PROBLEM (Jaillet [1985]; Laporte et al. [1994]; Bianchi et al. [2002a]; Bertsimas and Howell [1993]). This problem is a generalization of the famous TRAVELING SALESMAN PROBLEM (Lawler et al. [1985]; Lin [1965]; Held and Karp [1970]; Johnson and McGeoch [1997]) where in addition the presence of customers is modeled in a stochastic way. After that we discuss the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES (Campbell and Thomas [2008b, 2009]; Weyland et al. [2012a,b,d,c]) which is a generalization of the PROBABILISTIC TRAVELING SALESMAN PROBLEM considering time dependencies in terms of deadlines. Then we examine the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS (Bertsimas [1992]; Bianchi et al. [2006]; Dror et al. [1993]; Bastian and Rinnooy Kan [1992]). Here the demands of the customers are modeled in a stochastic way and a single vehicle with an integral capacity is used to satisfy the customers' demands. Finally, we present the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS (Gendreau et al. [1995, 1996b]). This problem is a combination of the PROBABILISTIC TRAVELING SALESMAN PROBLEM and the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS.

1.2.1 The Probabilistic Traveling Salesman Problem

The PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) has been introduced in Jaillet [1985] and is a generalization of the famous TRAVELING SALESMAN PROBLEM (TSP). Like for the TSP, a set of customers and distances between these customers are given. In addition, the presence of the customers is modeled in a stochastic way. Each customer has assigned a value which represents the probability with which this customer is present. Furthermore, the presence of different customers are independent events. This problem belongs to the class of *a priori optimization problems*. Here a solution is a tour visiting all customers exactly once, just as for the TSP. In this context such a solution is called a priori solution



(a) a priori tour (b) one possible a posteriori tour (c) another possible a posteriori tour

Figure 1.1. Example of how a posteriori tours are derived from a given a priori tour for the PTSP. Part (a) shows the given a priori tour. Parts (b) and (c) represent two particular realizations of the random events. Here the filled circles represent the customers that require a visit. These customers are visited in the order specified by the a priori tour, while the other customers are just skipped.

or a priori tour. This reflects that the order of the customers is determined before it is known which customers are present. After the presence of the customers becomes known, a so called a posteriori tour is derived from the a posteriori solution by skipping the customers which are not present. In this way the customers which are present are visited in the order given by the a priori tour. This process is illustrated in figure 1.1. Now the optimization goal is to find an a priori tour of minimum expected length over the a posteriori tours with respect to the given probabilities.

Most of the literature for the PTSP deals with heuristics for this problem. To our knowledge the only work which proposes an exact method is Laporte et al. [1994]. Here a formulation of the PTSP as an integer stochastic program is introduced. A branch-and-cut algorithm is then used to solve instances of up to 50 customers to optimality. This work was published almost 20 years ago. Although the computational power has increased a lot during the last two decades, the size of the instances which can be solved to optimality has not been significantly improved. In Bertsimas and Howell [1993] some theoretical properties for the PTSP are presented. Among them are improved bounds and asymptotic relations. This work also contains a comparison between the *a priori optimization* approach and the *reoptimization approach*. Additionally, some simple heuristics are analyzed. To tackle the PTSP many different metaheuristics have been proposed. Different *ant colony optimization* approaches are presented in Bianchi et al. [2002a,b]; Branke and Guntsch [2003]; Gutjahr [2004]; Branke

and Guntsch [2004]. A hybrid scatter search approach for the PTSP is discussed in Liu [2007] and an improved local search strategy for this approach is given in Liu [2008a]. A heuristic based on the aggregation of customers into clusters is proposed in Campbell [2006]. A memetic algorithm for the PTSP is suggested in Liu [2008b]. The generation of initial solutions is discussed in the context of genetic algorithms in Liu [2010]. In Marinakis et al. [2008]; Marinakis and Marinaki [2009, 2010] the authors present a greedy randomized adaptive search procedure, a hybrid honey bees mating optimization algorithm and a hybrid multi-swarm particle swarm optimization algorithm. Methods based on an approximation of the objective function using Monte Carlo sampling are discussed in Balaprakash et al. [2009b,a, 2010]; Birattari et al. [2008b]. In Birattari et al. [2008b] a local search algorithm using such an approximation of the objective function is proposed. A hybrid ant colony optimization approach based on this local search is then presented in Balaprakash et al. [2009b]. Further improvements for the local search algorithm are introduced in Balaprakash et al. [2009a] and metaheuristics based on this local search are finally presented in Balaprakash et al. [2010].

Before we give a formal definition of this problem, we would like to present an expression for the costs of an a priori tour. Let V be the set of n customers, let $d : V \times V \to \mathbb{R}^+$ be a function representing the distances between these customers and let $p : V \to [0,1]$ be a function representing the probabilities of the customers' presence. Given a permutation $\tau : \langle n \rangle \to V$ of the customers, which represents an a priori tour, the expected costs over the a posteriori tours with respect to the given probabilities can be computed according to Jaillet [1985] as

$$\begin{aligned} f_{\text{ptsp}}(\tau) &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}(\text{cost generated by the edge from } \tau_i \text{ to } \tau_j) \\ &\quad + \sum_{i=1}^{n} \sum_{j=1}^{i-1} \mathbb{E}(\text{cost generated by the edge from } \tau_i \text{ to } \tau_j) \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} d(\tau_i, \tau_j) p(\tau_i) p(\tau_j) \prod_{k=i+1}^{j-1} (1 - p(\tau_k)) \\ &\quad + \sum_{i=1}^{n} \sum_{j=1}^{i-1} d(\tau_i, \tau_j) p(\tau_i) p(\tau_j) \prod_{k=i+1}^{n} (1 - p(\tau_k)) \prod_{k=1}^{j-1} (1 - p(\tau_k)). \end{aligned}$$

Due to linearity of expectation the expected costs of the a posteriori tours is the sum of the expected costs generated by the edges between any pair of

customers. The expected cost generated by the edge between two customers is the distance between these customers multiplied by the probability that these customers are next to each other in an a posteriori solution. Customers τ_i and τ_j are next to each other in an a posteriori solution if both customers are present and if all customers between τ_i and τ_j are not present. Since the indices of the sums and products in the given formula range over at most n values, the costs of an a priori solution can be trivially computed in $\mathcal{O}(n^3)$ arithmetic operations. Using a specific order for the summations, the computational time can be reduced to $\mathcal{O}(n^2)$ (Jaillet [1985]). Using this expression for the costs of an a priori tour, we are now able to define the PTSP formally.

Problem 1 (PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP))**.** *Given a set V of size n, a function $d : V \times V \to \mathbb{R}^+$ and a function $p : V \to [0,1]$, the problem is to compute a permutation $\tau^\star : \langle n \rangle \to V$, such that $f_{ptsp}(\tau^\star) \leq f_{ptsp}(\tau)$ for any permutation $\tau : \langle n \rangle \to V$.*

1.2.2 The Probabilistic Traveling Salesman Problem with Deadlines

In Campbell and Thomas [2008b] the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES (PTSPD) has been introduced as a generalization of the PROBABILISTIC TRAVELING SALESMAN PROBLEM where time dependencies are modeled in terms of deadlines. More in detail, four different variants were proposed: three recourse models and one chance constrained model. In this thesis we will concentrate on the two very similar variants called PTSPD RECOURSE I with fixed penalties and PTSPD RECOURSE I with proportional penalties. We will formally introduce the PTSPD RECOURSE I with fixed penalties and refer to this problem simply as the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. Nonetheless, all our results can also be generalized with similar proofs to the other models. As for the PTSP we model the presence of the customers in a stochastic way. Additionally, each customer has assigned a deadline and a penalty value. We are then interested in an a priori solution with minimum expected costs over the a posteriori solutions. For this problem the costs of the a posteriori solutions are the travel times plus penalties for deadlines which are violated. For each violated deadline a fixed penalty, dependent on the customer, incurs. Note that for the variant called PTSPD RECOURSE I with proportional penalties the penalties for missed deadlines are proportional to the delay. This is the only difference to the model with fixed penalties.

Since the PTSPD has been introduced recently in 2008, not a lot of publications are dealing with this problem so far. In Campbell and Thomas [2008b] the problem is introduced and all the four variants are formally defined. Some theoretical properties are derived and some artificial special cases of the problem are discussed. The only other publication regarding the PTSPD is Campbell and Thomas [2009]. Here approximations for the objective function are introduced. These approximations are then compared with the exact evaluation of the objective function using a simple local search algorithm. It has been shown that the approximations can be used within the local search in combination with the exact evaluation of the objective function to obtain solutions of competitive quality, while the computational time could be reduced significantly. Although the computational complexity of the PTSPD objective function was not known, the authors stated that one of the main challenges for the PTSPD is the computationally demanding objective function. We show in this work that the objective function is in fact hard to compute from a computational complexity point of view. Additionally, we also show how to address this challenge and how to obtain high quality solutions within a reasonable computational time.

The formal definition of the PTSPD is similar to that of the PTSP. With V we refer to the set of n customers. As for the PTSP we have given distances between the locations which are represented by a function $d : V \times V \to \mathbb{R}^+$ and probabilities for the customers' presence which are represented by a function $p : V \to [0,1]$. Since we are using time dependencies, the routes require a fixed starting point. This starting point is a special element $v_1 \in V$ for which we set $p(v_1) = 1$. In this context the starting point is usually called the depot. The deadlines for the different customers are now modeled using a function $t : V \to \mathbb{R}^+$ and the penalty values for the different customers are modeled using a function $h : V \to \mathbb{R}^+$. To keep the mathematical formulation as simple as possible we also define these values for the depot v_1, although we will meet the deadline at the depot in any case, since we start the tour there. An a priori solution can now be represented by a permutation $\tau : \langle n \rangle \to V$ with $\tau_1 = v_1$.

As we did in the previous section we will first give a mathematical expression for the costs of an a priori tour. Let $\tau : \langle n \rangle \to V$ with $\tau_1 = v_1$ be an a priori solution. For all $v \in V$ let A_v be a random variable indicating the arrival time at customer v. Since the travel times of the a posteriori solutions are identical to those for the PTSP, the costs of τ can be expressed as

$$f_{\text{ptspd}}(\tau) \;=\; f_{\text{ptsp}}(\tau) + \sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} \geq t(\tau_i))\, h(\tau_i).$$

The first part of the costs represents the expected travel times over the a posteriori solutions. The second part represents the penalties for missed deadlines. While the first part of the costs can be computed in polynomial time, we will see later in this work that this is very unlikely for the second part of the costs. With this expression for the costs of an a prior tour, we define the PTSPD in the following way.

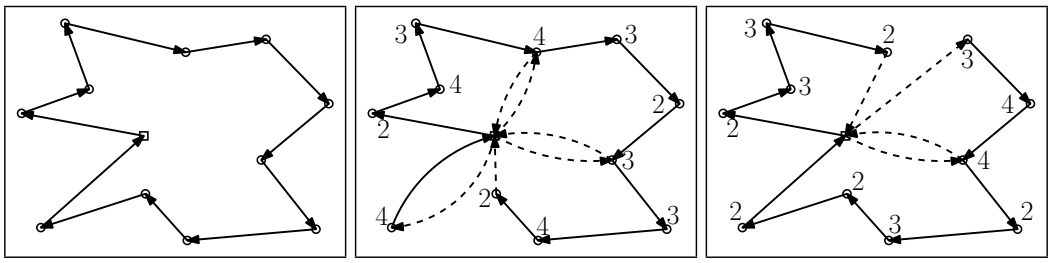
Problem 2 (PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES (PTSPD)). *Given a set V of size n with a special element $v_1 \in V$, a function $d : V \times V \to \mathbb{R}^+$, a function $p : V \to [0,1]$, a function $t : V \to \mathbb{R}^+$ and a function $h : V \to \mathbb{R}^+$, the problem is to compute a permutation $\tau^\star : \langle n \rangle \to V$ with $\tau^\star_1 = v_1$, such that $f_{ptspd}(\tau^\star) \leq f_{ptspd}(\tau)$ for any permutation $\tau : \langle n \rangle \to V$ with $\tau_1 = v_1$.*

Note that in some situations we use a different identifier for the depot to allow for easier notations. The definition of the problem changes accordingly, but it should be clear in the different contexts.

1.2.3 The Vehicle Routing Problem with Stochastic Demands

Like the PROBABILISTIC TRAVELING SALESMAN PROBLEM, the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS (VRPSD, Bertsimas [1992]) can be seen as a special case of the TRAVELING SALESMAN PROBLEM. In contrast to the TRAVELING SALESMAN PROBLEM, we use a vehicle of a fixed capacity to deliver identical and integral goods from a depot to the different customers. In some situations this problem is also used to model a collection process, where the goods are collected at the customers and transported to the depot. The customers' demands are modeled in a stochastic way and the sum of all the demands usually exceeds the vehicle capacity by a multiple. Therefore, the vehicle has to visit the depot frequently to load the goods. To serve the customers the vehicle starts fully loaded at the depot. It then visits the customers in a certain order and delivers the required amount of goods. If the vehicle runs out of goods while a customer is served, it returns to the depot, refills the goods and continues at that customer. If the vehicle runs out of goods just after a customer has been served, it returns to the depot, refills the goods and continues at the next customer. Note that this restocking strategy is called the basic restocking strategy. After all customers have been processed, the vehicle returns to the depot.

Like the PTSP and the PTSPD, this problem belongs to the class of *a priori optimization problems*. As for the other problems the task is to find an a priori solution such that the expected costs of the a posteriori solutions with respect to the given demand distributions is minimized. An a priori solution for this



(a) a priori tour (b) one possible a posteriori tour (c) another possible a posteriori tour

Figure 1.2. Example of how a posteriori tours are derived from a given a priori tour for the VRPSD. The vehicle capacity in this example is 10 and the depot is visualized by the square. Part (a) shows the given a priori tour. Parts (b) and (c) represent two particular realizations of the random events. Here the numbers denote the demands of the customers. The customers are served in the order specified by the a priori solution and the vehicle is refilled only if it gets empty. Note that if a vehicle gets empty after fully serving a customer, it is refilled at the depot and proceeds with the next customer. This happens between the last two customers in (b) and between the fourth and fifth customer in (c).

problem is a tour starting at the depot and visiting all customers exactly once. The costs for an a posteriori solution are just the total travel times. Note that here restocking actions are influencing the travel times. Figure 1.2 illustrates the relation between the a priori solution and the a posteriori solution in the context of the VRPSD.

In Bertsimas [1992] closed-form expressions and algorithms for the VRPSD objective function are given. Different interesting bounds are derived and a comparison with the corresponding *reoptimization approach* is performed. Additionally, the worst-case behavior of some simple heuristics is analyzed. Bastian and Rinnooy Kan [1992] introduces modifications of existing models and shows that under some assumptions the VRPSD exhibits the structure of the TIME-DEPENDENT TRAVELING SALESMAN PROBLEM (Lucena [1990]; Gouveia and Voß [1995]; Vander Wiel and Sahinidis [1996]). In Dror et al. [1993] a chance-constrained model and three recourse models of the VRPSD are introduced. It is shown that the chance-constrained model can be solved to optimality and that the recourse models can be solved by optimizing multiple instances of the TSP. Hjorring and Holt [1999] introduces new optimality cuts for the VRPSD. The problem is then solved using the integer L-shaped method with an approxima-

tion of the restocking costs. The integer L-shaped method is also used in Laporte et al. [2002] to solve instances of size up to 100 to optimality. In Christiansen and Lysgaard [2007] a formulation of the VRPSD as a set partitioning problem is introduced and promising results are reported. A local branching method in combination with Monte Carlo sampling is used in Rei et al. [2010]. Metaheuristics are analyzed in Bianchi et al. [2004, 2006]; Chepuri and Homem-de Mello [2005]. While the performance of different metaheuristics are compared in Bianchi et al. [2004, 2006], Chepuri and Homem-de Mello [2005] deals with an algorithm based on the cross-entropy method. Variants with multiple goods are discussed in Mendoza et al. [2010, 2011]. A robust optimization approach is given in Sungur et al. [2008]. Finally, the problem is discussed from a reoptimization point of view in Novoa and Storer [2009]; Secomandi and Margot [2009].

We now focus on the VRPSD objective function and present the formal problem definition shortly after. Let V be the set of n customers including the depot $v_1 \in V$. The distances between the customers are again modeled using a function $d : V \times V \to \mathbb{R}^+$. Let $Q \in \mathbb{N}$ be the capacity of the vehicle. The demand distributions can be modeled by a function $g : V \times \langle Q \rangle \to \mathbb{R}^+$ with $\sum_{i=1}^{Q} g(v,i) = 1$ for all $v \in V$. An a prior solution can now be simply represented by a permutation $\tau : \langle n \rangle \to V$ with $\tau_1 = v_1$. For a given solution let A_v be a random variable describing the amount of goods in the vehicle just before customer $v \in V$ is processed and let D_v be a random variable describing the demand of customer v according to the given demand distribution. With $\tau_{n+1} = \tau_1$ and $D_{\tau_1} = 0$ the expected costs of τ can then be expressed as

$$\begin{aligned} f_{\text{vrpsd}}(\tau) \;=\; & \sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} > D_{\tau_i})\, d(\tau_i, \tau_{i+1}) \\ & + \sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} = D_{\tau_i})\,(d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\ & + \sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} < D_{\tau_i})\,(d(\tau_i, \tau_1) + d(\tau_1, \tau_i) + d(\tau_i, \tau_{i+1})). \end{aligned}$$

The first case corresponds to the situation in which the vehicle still contains goods after serving a customer. In that case the vehicle continues to the next customer. The second case corresponds to the situation in which the vehicle is empty after serving a customer. In this situation the vehicle returns to the depot for a restocking action and continues the tour at the next customer. The last

case corresponds to the situation in which a customer cannot be fully served. A restocking action is necessary and the vehicle travels to the depot and back to the customer. The customer is then fully served and the vehicle continues to the next customer. Note that the objective function can be computed in a runtime of $\mathcal{O}(nQ^2)$ with a dynamic programming approach Bertsimas [1992]. That means for a fixed value of Q, which is common for this kind of problem, and even for a value of Q which is polynomially in the input size, the objective function can be computed in polynomial time. Using the mathematical expression for the objective function, we are able to state the problem as follows.

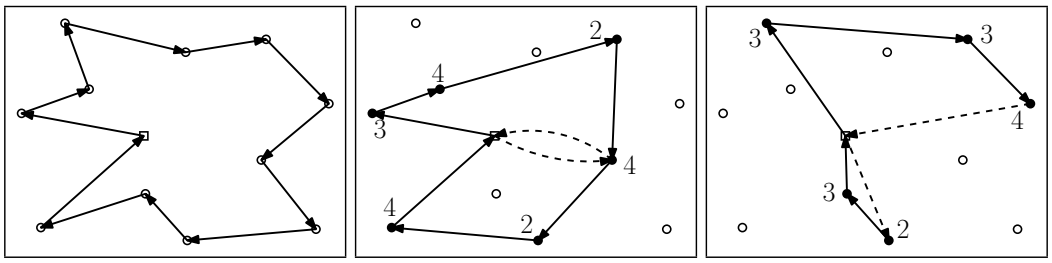
Problem 3 (Vehicle Routing Problem with Stochastic Demands (VRPSD))**.** *Given a set V of size n with a special element $v_1 \in V$, a function $d : V \times V \to \mathbb{R}^+$, a value $Q \in \mathbb{N}$ and a function $g : V \times \langle Q \rangle \to \mathbb{R}^+$ with $\sum_{i=1}^{Q} g(v,i) = 1$ for all $v \in V$, the problem is to compute a permutation $\tau^\star : \langle n \rangle \to V$ with $\tau_1^\star = v_1$, such that $f_{vrpsd}(\tau^\star) \leq f_{vrpsd}(\tau)$ for any permutation $\tau : \langle n \rangle \to V$ with $\tau_1 = v_1$.*

1.2.4 The Vehicle Routing Problem with Stochastic Demands and Customers

The Vehicle Routing Problem with Stochastic Demands and Customers (VRPSDC, Gendreau et al. [1995]) is a combination of the Probabilistic Traveling Salesman Problem and the Vehicle Routing Problem with Stochastic Demands. In addition to the formulation of the VRPSD, the presence of the customers is modeled in a stochastic way. Depending on the context we can also see this problem as a variant of the Vehicle Routing Problem with Stochastic Demands where we formally allow a demand of zero and customers to be skipped in this case. The relation between the a priori solution and the a posteriori solution for the VRPSDC is shown in figure 1.3. In the following we will omit an informal description of the problem, since the combination of the PTSP and the VRPSD is straightforward.

The literature for the VRPSDC is quite rare. A formulation of the VRPSDC as a stochastic integer program is given in Gendreau et al. [1995]. Here the integer L-shaped method is used to solve the problem to optimality. A tabu search heuristic for tackling larger instances is presented in Gendreau et al. [1996b]. Finally, a new solution approach including numerical results is given in FuCe et al. [2005].

For the formal definition of the problem we use the same notations as before. We have given a set V of n customers including the depot $v_1 \in V$. A function $d : V \times V \to \mathbb{R}^+$ is representing the distances between the customers. The capacity of the vehicle is $Q \in \mathbb{N}$ and the customers' demands are modeled by



(a) a priori tour (b) one possible a posteriori tour (c) another possible a posteriori tour

Figure 1.3. Example of how a posteriori tours are derived from a given a priori tour for the VRPSDC. The vehicle capacity in this example is 10 and the depot is visualized by the square. Part (a) shows the given a priori tour. Parts (b) and (c) represent two particular realizations of the random events. Here the filled circles represent the customers that require a visit and the numbers denote the demands of these customers. The customers that require a visit are served in the order specified by the a priori solution and the vehicle is refilled only if it gets empty. Note that if a vehicle gets empty after fully serving a customer, it is refilled at the depot and proceeds with the next customer. This happens between the third and fourth customer in (c).

a function $g : V \times \langle Q \rangle \to \mathbb{R}^+$ with $\sum_{i=1}^{Q} g(v,i) = 1$ for all $v \in V$. Moreover, a function $p : V \to [0,1]$ represents the probabilities for the presence of the different customers. As usual, a solution is represented by a permutation $\tau : \langle n \rangle \to V$ with $\tau_1 = v_1$.

Now let $\tau : \langle n \rangle \to V$ be such a solution. A_v is a random variable describing the amount of goods in the vehicle just before customer $v \in V$ is processed and D_v is a random variable describing the demand of customer v according to the given demand distribution. Additionally, N_v is a random variable indicating the next customer after v (with respect to τ) which requires to be visited. With $\tau_{n+1} = \tau_1$ and $D_{\tau_1} = 0$ the expected costs for τ are then

$$
\begin{aligned}
f_{\text{vrpsdc}}(\tau) = &\sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} > D_{\tau_i}) \sum_{j=1}^{n} \mathbb{P}(N_{\tau_i} = \tau_j)\, d(\tau_i, \tau_j) \\
&+ \sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} = D_{\tau_i}) \left(d(\tau_i, \tau_1) + \sum_{j=1}^{n} \mathbb{P}(N_{\tau_i} = \tau_j)\, d(\tau_1, \tau_j) \right) \\
&+ \sum_{i=1}^{n} \mathbb{P}(A_{\tau_i} < D_{\tau_i}) \left(d(\tau_i, \tau_1) + d(\tau_1, \tau_i) + \sum_{j=1}^{n} \mathbb{P}(N_{\tau_i} = \tau_j)\, d(\tau_i, \tau_j) \right).
\end{aligned}
$$

With this expression the problem can be stated as follows.

Problem 4 (Vehicle Routing Problem with Stochastic Demands and Customers (VRPSDC))**.** *Given a set V of size n with a special element $v_1 \in V$, a function $d : V \times V \to \mathbb{R}^+$, a value $Q \in \mathbb{N}$, a function $g : V \times \langle Q \rangle \to \mathbb{R}^+$ with $\sum_{i=1}^{Q} g(v,i) = 1$ for all $v \in V$ and a function $p : V \to [0,1]$, the problem is to compute a permutation $\tau^\star : \langle n \rangle \to V$ with $\tau^\star_1 = v_1$, such that $f_{vrpsdc}(\tau^\star) \leq f_{vrpsdc}(\tau)$ for any permutation $\tau : \langle n \rangle \to V$ $\tau_1 = v_1$.*

1.3 Outline

In this section we give a brief overview about the content of the subsequent chapters and the main results that are presented in this thesis.

All in all, the thesis is partitioned into a theoretical part and a practical part. The theoretical part consists of chapters 2 to 4. The results presented in these chapters are mostly about the complexity of computational tasks related to stochastic vehicle routing problems. Hardness results for the optimization problems, that are presented in this chapter, motivate the usage of heuristics for tackling these problems. Additionally, hardness results for the objective functions of some stochastic vehicle routing problems motivate the usage of efficient approximations for these objective functions. The practical part consists of chapters 5 to 8 and is motivated and partially based on the theoretical results. In chapters 5 and 6 we present heuristics for the PTSP and the PTSPD using approximations of the objective functions. Comprehensive computational studies reveal the efficiency of these heuristics with respect to existing approaches. In chapter 7 we present a metaheuristic framework for solving stochastic combinatorial optimization problems using graphics processing units. A case study on the PTSPD shows that major runtime improvements can be obtained in this way.

Finally, chapter 8 deals with a real world stochastic vehicle routing problem. We introduce a model for this problem and present an efficient heuristic. This problem is part of a project that has been initiated in 2010 by Caritas Suisse. The project is still in an early stage, but with our work we were able to successfully support some of the decision processes at this stage.

Note that this thesis is based on Weyland et al. [2009a,b, 2011a,b, 2012a,b,c,d, 2013a,b]. All these publications have been written by the author of this thesis under guidance and supervision of the corresponding co-authors. Weyland et al. [2012b,c, 2013b] are journal articles, Weyland et al. [2011a, 2012a] are journal articles which are currently under review and Weyland et al. [2009a,b, 2011b, 2012d, 2013a] are conference papers. At the beginning of the different sections we always refer to the corresponding publications.

Chapter 2

Convergence Results for Vehicle Routing Problems with Stochastic Demands

In this chapter we give convergence results for vehicle routing problems where demands are modeled in a stochastic way, like for example the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS (Bertsimas [1992]) and the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS (Gendreau et al. [1995, 1996b]). The results presented here are based on the publication Weyland et al. [2013a]. These results are interesting for two reasons. On the one hand, they have an immediate impact for practical applications of vehicle routing problems with stochastic demands. We will discuss the corresponding implications at the end of this chapter. On the other hand, they are used in our analyses of the computational complexity of stochastic vehicle routing problems on substantially stochastic instances in chapter 3.

The remainder of this chapter is organized in the following way. We will first introduce a Markov chain model (Revuz [2005]) for vehicle routing problems with stochastic demands, which describes the amount of goods in the vehicle while the different customers are processed. Then we continue with the convergence results itself. Here we show that under mild conditions the distribution of the goods in the vehicle converges to the uniform distribution. This gives us a sort of asymptotic equivalence of the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS and the TRAVELING SALESMAN PROBLEM (Lin [1965]; Johnson and McGeoch [1997]) as well as the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS and the PROBABILISTIC TRAVELING SALESMAN PROBLEM (Jaillet [1985]). For practical applications it is very interesting to identify the

speed with which the distribution of goods in the vehicle converges to the uniform distribution. We investigate this convergence speed for binomial demand distributions. After that we finish this chapter with an extensive discussion of our results.

2.1 A Markov Chain Model

In this section we introduce a Markov chain model (Revuz [2005]) that describes the amount of goods in the vehicle, which is fundamental for the remaining part of this chapter, and in particular for the proofs in the following two sections. After that we prove some properties for this model using the basic restocking strategy. For more details about Markov chains we refer to Revuz [2005].

Given the amount of goods in the vehicle before processing a certain customer, the amount of goods after the customer has been processed depends only on the demand distribution of this customer (and in the case of the VRPSDC additionally on the probability that this customer requires a visit) and is independent of the demands of the other customers. Thus it is clear that we can model the amount of goods in the vehicle for a given route with a time-discrete Markov chain, starting with a distribution representing a full vehicle, and using transition matrices that are based on the demand distributions of the customers and on the restocking strategy.

The different states represent the different amounts of goods in the vehicle. If the amount of goods is bounded from above by the vehicle capacity Q, and if we observe the amount of goods after a possible restocking action has been performed, it is sufficient to have different states for the amount levels $1, 2, \ldots, Q$, because a minimum requirement for a restocking strategy should be to perform a restocking action, if the vehicle is empty after processing a customer. For being able to use modular arithmetic in a straightforward way, we use in the following $1, 2, \ldots, Q-1$ for amount levels of $1, 2, \ldots, Q-1$ and 0 for an amount of Q.

That means the distribution of the amount of goods can be represented by a column vector of size Q, whose elements are all non-negative and sum up to 1. Such a vector is called stochastic (Latouche and Ramaswami [1987]). Furthermore, the transition matrices are of size $Q \times Q$ and depend only on the demand distributions of the customers (and in the case of the VRPSDC additionally on the probability that the customer requires a visit), and on the restocking strategy that is used. The entry in row i and column j represents the probability that we reach state i from state j in one step. In other words, this is the probability that we have an amount of i goods after processing the customer and performing a

possible restocking action, if the amount has been j goods before processing the customer. By definition the transition matrices are stochastic in its columns, i.e. that the elements in each column sum up to 1. Note that the transition matrices for the VRPSDC are convex combinations of the corresponding transition matrices for the VRPSD and the identity matrix, since with a certain probability a customer is skipped and the amount of goods does not change, and otherwise the customer is served exactly as in the VRPSD.

In this chapter we focus on the basic restocking strategy which has been used in the formal definitions of the VRPSD and the VRPSDC in chapter 1. This means that we perform a restocking action if we run out of goods while processing a customer, or if the vehicle is empty after processing a customer. Therefore, the probability to reach state i from state j is the same as the probability to reach state $(i+k) \bmod Q$ from state $(j+k) \bmod Q$, $\forall k \in \{1,2,\ldots,Q-1\}$. These observations lead to two additional properties for the transition matrices. Two successive rows only differ in a cyclic shift of one position, in particular for each row the following row is shifted cyclic one position to the right. The other property, which follows directly from this one for square matrices, is that the transition matrices are also stochastic in their rows, i.e. that the elements in each row sum up to 1.

Before we finish this section, we give a formal overview about the properties mentioned above, using the following definitions. For the vectors and matrices we use indices starting at 0 to allow the use of modular arithmetic in a straightforward way.

Definition 1 (Column stochastic matrix)**.** *A $m \times n$ matrix $A = (a_{ij})$ is called stochastic in its columns, if the following two properties hold:*

1. $\forall i \in \{0,1,\ldots,m-1\},\ \ \forall j \in \{0,1,\ldots,n-1\} :\ \ a_{ij} \geq 0$
2. $\forall j \in \{0,1,\ldots,n-1\} :\ \ \sum_{i=0}^{m-1} a_{ij} = 1.$

Definition 2 (Row stochastic matrix)**.** *A $m \times n$ matrix $A = (a_{ij})$ is called stochastic in its rows, if the transposed matrix A^T is stochastic in its columns.*

Definition 3 (Doubly stochastic matrix)**.** *A matrix A is called doubly stochastic, if it is stochastic in its columns and stochastic in its rows.*

Definition 4 (Cyclic matrix)**.** *A $m \times m$ matrix $A = (a_{ij})$ is called cyclic, if the following holds:*

$$\forall i \in \{0,1,\ldots,m-1\},\ \ \forall j \in \{0,1,\ldots,m-1\} :\ \ a_{ij} = a_{(i+1) \bmod m,(j+1) \bmod m}.$$

Now we are able to give a formal summary of the properties of the transition matrices in the following lemma.

Lemma 1. *Using the Markov chain model introduced at the beginning of this section for the VRPSD and the VRPSDC with the basic restocking strategy, the transition matrices are square matrices, doubly stochastic and cyclic.*

Another important fact is that column stochastic matrices, row stochastic matrices, doubly stochastic matrices and cyclic matrices are all closed under multiplication. A proof for the last statement is given in appendix A, the other statements are well known results (Latouche and Ramaswami [1987]) and easy to verify.

2.2 Convergence Results

We show in this section that there is under some mild conditions an asymptotic equivalence between the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS and the TRAVELING SALESMAN PROBLEM, as well as between the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS and the PROBABILISTIC TRAVELING SALESMAN PROBLEM.

One might think that it is important for the VRPSD and the VRPSDC to optimize the (expected) length of the route as well as to optimize the route in a way, such that those customers which cause a restocking action with high probability, are located close to the depot. We show that the distribution of the amount of goods in the vehicle converges to the uniform distribution under certain conditions. In this case the probability that restocking is required at a certain customer depends mainly on the distribution of its demand and only slightly on the particular shape of the tour. In other words, if the amount of goods in the vehicle is close to the uniform distribution, it is sufficient to optimize the (expected) length of the tour.

Let us assume for the moment that the vehicle is loaded in a probabilistic way, such that the amount of goods is initially distributed according to the uniform distribution. Then the VRPSD and the TSP, and the VRPSDC and the PTSP are equivalent, if the basic restocking strategy is used. These equivalences can be verified easily using the Markov chain model introduced in the previous section. Together with the convergence of the amount of goods in the vehicle towards the uniform distribution, this observation is the basic idea behind our analyses.

We now start with a more general mathematical result. After that we show under which conditions this result can be used to derive concrete results for the VRPSD and for the VRPSDC.

Theorem 1. *Let $(A_n)_{n\in\mathbb{N}}$ be a series of doubly stochastic $m \times m$ matrices, whose entries are all bounded from below by a constant $L > 0$. Then $\lim_{n\to\infty} \prod_{i=1}^{n} A_i = \frac{1}{m} J$, where J is a matrix of ones.*

Proof. This theorem is trivially true for $m = 1$, so we can assume $m > 1$ for the following proof. Since the product of doubly stochastic matrices is a doubly stochastic matrix and for any doubly stochastic matrix $A = (a_{ij})$ we have $\sum_{j=0}^{m-1} a_{ij} = 1 \ \ \forall i \in \{0, 1, \ldots, m-1\}$ and $a_{ij} \geq 0 \ \ \forall i, j \in \{0, 1, \ldots, m-1\}$, it is sufficient to show that the entries in each row converge to the same value. We prove this by showing that the difference between the smallest and the largest entry in each row converges to 0.

We show by induction that the difference between the smallest and the largest entry in each row of $\prod_{i=1}^{n} A_i$ is bounded from above by $(1-2L)^n$. It is clear that this expression is always non-negative, because L is trivially bounded from above by $1/2$.

For $n = 1$ this is clearly the case. The smallest entry is bounded from below by L and since we have doubly stochastic matrices, the largest entry is bounded from above by $1 - L$. The difference of the the largest entry and the smallest entry can be at most $(1-L)-L = 1-2L = (1-2L)^1$. Now suppose the statement holds for $n \in \mathbb{N}$. Here we use $A^{(n)} = \prod_{i=1}^{n} A_i$ as an abbreviation for the product of the first n matrices. Let $i \in \{0, 1, \ldots, m-1\}$ be a fixed row index and let $l_i(n)$ and $u_i(n)$ be the smallest, respectively the largest entry in the i-th row of $A^{(n)}$. The entries in the i-th row of $A^{(n+1)}$ are convex combinations of the entries in the i-th row of $A^{(n)}$, where the corresponding coefficients of each of these convex combinations is a column of A_{n+1}. Since the entries of A_{n+1} are bounded from below by L we have

$$u_i(n+1) \leq (1-L)u_i(n) + Ll_i(n) = u_i(n) - L(u_i(n) - l_i(n)) \text{ and}$$

$$l_i(n+1) \geq (1-L)l_i(n) + Lu_i(n) = l_i(n) + L(u_i(n) - l_i(n)).$$

The difference of these values is

$$\begin{aligned} u_i(n+1) - l_i(n+1) &\leq u_i(n) - L(u_i(n) - l_i(n)) - l_i(n) - L(u_i(n) - l_i(n)) \\ &= u_i(n) - l_i(n) - 2L(u_i(n) - l_i(n)) \\ &= (1-2L)(u_i(n) - l_i(n)) \\ &\leq (1-2L)^{n+1}. \end{aligned}$$

In the last step we used the induction hypothesis and since $\lim_{n\to\infty}(1-2L)^n = 0$ we finished the proof. □

In the previous section we have seen that we can model the distribution of the amount of goods in the vehicle with a Markov chain. It is possible to identify the matrices A_i of theorem 1 with transition matrices of that Markov chain. Using the basic restocking strategy these transition matrices are doubly stochastic, but unfortunately it is a very strong assumption that their entries are all bounded from below by a constant $L > 0$. In this case each customer has a strictly positive probability for each possible demand and this assumption is obviously too strong for practical applications. In the remaining part of this section we show that we can use theorem 1 also under more moderate conditions.

The main idea here is to partition the matrices into blocks of consecutive matrices, such that the product of all the matrices within a block fulfills the conditions of theorem 1. Due to the associative law the product of all the matrices can be obtained by first calculating the products of the matrices within each block and then multiplying the resulting products. In this way we are able to state theorem 1 using much weaker assumptions on the matrices used. These weaker assumptions are comprised in the following definition and as we will see later, it is not even required that these assumptions hold for all of the matrices.

Definition 5 (greatest common divisor property, gcd property)**.** *A cyclic $m \times m$ matrix $A = (a_{ij})$ with only non-negative entries and with strictly positive entries $a_{i_1 0}, a_{i_2 0}, \ldots, a_{i_l 0}$, $i_1 < i_2 < \ldots < i_l$ in the first column fulfills the greatest common divisor property (short: gcd property), if* $\gcd(i_l - i_1, i_l - i_2, \ldots, i_l - i_{l-1}, m) = 1$.

Instead of using the differences relative to i_l we could have also used differences relative to any other of the elements i_k, $k \in \{1, 2, \ldots, l\}$. The definition is also invariant with respect to the chosen column. We prove both invariances in appendix A.

At first glance this definition looks quite technical, but it is usually fulfilled by a lot of the transition matrices for practical instances of the VRPSD and the

VRPSDC. In particular, the greatest common divisor property is fulfilled for a transition matrix, if the corresponding customer has two different demands of amount k and $k+1$ with strictly positive probabilities. For the VRPSDC we have always a strictly positive entry at the first position in the first column (unless the probability for visiting this customer is exactly 1) and therefore the condition is even slightly weaker in this case. We discuss this issue more in detail in section 2.4. Here we continue with deriving a convergence result as in theorem 1 under more mild assumptions comprised by the gcd property.

We start with proving the following important lemma using the definition of the gcd property and some modular arithmetic.

Lemma 2. *Let A be a $m \times m$ matrix with only non-negative entries and with k_i strictly positive entries in row i, $i \in \{0,1,\ldots,m-1\}$, and let B be a cyclic $m \times m$ matrix with only non-negative entries fulfilling the greatest common divisor property. Then the product $C = AB$ contains only non-negative entries and has at least $\min\{k_i+1, m\}$ strictly positive entries in row i, for each $i \in \{0,1,\ldots,m-1\}$.*

Proof. The elements of the set $I = \{i_1, i_2, \ldots, i_n\} \subset \mathbb{Z}/m\mathbb{Z}$, $n \in \mathbb{N}$, with $\gcd(i_1, i_2, \ldots, i_n, m) = 1$ can be used to generate any element of $\mathbb{Z}/m\mathbb{Z}$ by a linear combination of $i_1, i_2, \ldots, i_n$. This is a well known algebraic result, see e.g. Baldoni et al. [2008].

For our proof we need the following useful property. Given a set $S \subset \mathbb{Z}/m\mathbb{Z}$ with $|S| < |\mathbb{Z}/m\mathbb{Z}|$, and I as above, then for the set $S' = \{s+i \mid s \in S, i \in I \cup \{0\}\}$ the inequality $|S| < |S'|$ holds. Otherwise we cannot generate all elements of $\mathbb{Z}/m\mathbb{Z}$ by linear combinations of $i_1, i_2, \ldots, i_n$.

Let $i \in \{0,1,\ldots,m-1\}$ be a fixed row index with $k_i < m$, let $S = \{j \mid a_{ij} > 0\} \subset \mathbb{Z}/m\mathbb{Z}$ and let $S' = \{j \mid c_{ij} > 0\} \subset \mathbb{Z}/m\mathbb{Z}$. Furthermore let $b_{j_1 0}, b_{j_2 0}, \ldots, b_{j_l 0}$, $j_1 < j_2 < \ldots < j_l$ be the strictly positive entries in the first column of B. Due to the properties of our matrices, we have

$$\begin{aligned}
\left|S'\right| &= \left|\{(s+b) \bmod m \mid s \in S,\ b = -j_i,\ i \in \{1,2,\ldots,l\}\}\right| \\
&= \left|\{(s-j_l+b) \bmod m \mid s \in S,\ b = j_l - j_i,\ i \in \{1,2,\ldots,l\}\}\right| \\
&= \left|\{(s-j_l+b) \bmod m \mid s \in S,\ b \in \{j_l - j_1, j_l - j_2, \ldots, j_l - j_{l-1}\} \cup \{0\}\}\right| \\
&= \left|\{(s+b) \bmod m \mid s \in S,\ b \in \{j_l - j_1, j_l - j_2, \ldots, j_l - j_{l-1}\} \cup \{0\}\}\right| \\
&> |S|,
\end{aligned}$$

which concludes the proof for all rows with less than m strictly positive entries.

Now let $i \in \{0,1,\ldots,m-1\}$ be a fixed row index with $k_i = m$. Since each column of B has at least 1 strictly positive entry, the number of strictly positive

entries in the i-th row of $C = AB$ is m which concludes the proof for rows with exactly m strictly positive entries. □

Using this lemma we can obtain a sufficient condition for the case that the product of a number of transition matrices contains only strictly positive entries.

Corollary 1. *Let $A_1, A_2, \ldots, A_k$ be cyclic $m \times m$ matrices with only non-negative entries and with at least 1 strictly positive entry in each row, and let $m-1$ of these matrices fulfill the greatest common divisor property. Then $B = \prod_{i=1}^{k} A_i$ contains only strictly positive entries.*

Proof. Since the matrices are cyclic and have at least 1 strictly positive entry in each column the number of strictly positive entries in the series of products is never decreasing in any of the rows. Due to lemma 2 a multiplication with one of the matrices, which fulfills the gcd property, increases the number of strictly positive entries in each row by at least 1 (if it has not been maximum already). Putting these observations together the corollary follows easily. □

To generalize the convergence result of theorem 1 we need the following lemma.

Lemma 3. *Let $A = (a_{ij})$ be a $m \times m$ matrix containing only strictly positive entries, with a smallest entry l and a largest entry u and let B be a $m \times m$ column stochastic matrix. Then $u' - l' \leq u - l$, where u' and l' are the largest, respectively the smallest, strictly positive entries in AB.*

Proof. The elements of AB are convex combinations of elements of A and therefore the elements of AB are bounded from below by l and from above by u. □

Now we are able to generalize the convergence result of theorem 1.

Theorem 2. *Let $(A_n)_{n\in\mathbb{N}}$ be a series of doubly stochastic, cyclic $m \times m$ matrices, whose entries are all bounded from below by a constant $L > 0$. Let the greatest common divisor property be fulfilled by infinitely many of these matrices and let $k_1 < k_2 < \ldots$ be the indices of the matrices that fulfill the gcd property. If $\max\{k_{i+1} - k_i \mid i \in \mathbb{N}\} \leq k$ for a constant $k \geq k_1$ then $\lim_{n\to\infty} \prod_{i=1}^{n} A_i = \frac{1}{m} J$, where J is a matrix of ones.*

Proof. To proof this theorem we have to combine the previous results. Here we show like in theorem 1 that the differences between the largest and the

smallest element in each row of the successive products $\prod_{i=1}^{n} A_i$ are monotonically decreasing and converge to 0.

It is possible to partition the matrices into blocks of at most $k(m-1)$ consecutive matrices, such that the matrices in each block fulfill the conditions of corollary 1. Let $1 = j_1 < j_2 < \ldots$ be the indices at which the first, second, ... block begins. Then we can bound the smallest entry of the products $\prod_{i=j_l}^{j_{l+1}-1} A_i$ by $L' = L^{k(m-1)}$.

We can now create a new series $(B_n)_{n\in\mathbb{N}}$, with

$$B_n := \prod_{i=j_n}^{j_{n+1}-1} A_i.$$

Due to corollary 1 these matrices fulfill the conditions of theorem 1 and thus we have $\lim_{n\to\infty} \prod_{i=1}^{n} B_i = \frac{1}{m} J$. Analogue to the proof of theorem 1 we can now bound the difference between the largest and the smallest element in each row of the product $\prod_{i=1}^{n} B_i = \prod_{i=1}^{j_{n+1}-1} A_i$ from above by $(1-2L')^n$. With lemma 3 we can show that the difference between the largest and the smallest element of $\prod_{i=1}^{j} A_i$ can also be bounded from above by $(1-2L')^n$, if $j_{n+1} \leq j < j_{n+2} - 1$. This concludes the proof. □

Let us quickly connect these results with the original stochastic vehicle routing problems. What we have seen on a more abstract level is the following. If we impose some mild conditions on the transition matrices of our Markov chain model, the amount of goods in the vehicle after a customer has been processed is approaching the uniform distribution as we continue on the tour and more and more customers are visited. For the basic restocking strategy which we are using here, the transition matrices are characterized only by the customers' demands (and in the case of the VRPSDC additionally by the probability that the customers require to be visited). Therefore, the mild conditions on the transition matrices are transferred to mild conditions on the customers' demands. At the very beginning of this chapter we have already observed that it is sufficient to optimize the (expected) length of the route if the amount of goods in the vehicle is distributed according to the uniform distribution. Together with our convergence result, this shows that it is sufficient to focus on optimizing the (expected) length of the route in many situations. In the next section we will examine the

convergence speed for a typical demand distribution and in section 2.4 we will discuss the impacts of the results for the VRPSD and the VRPSDC more in detail.

2.3 Convergence Speed for Binomial Demand Distributions

In this section we want to examine the convergence speed in the case where the demands are distributed according to binomial distributions with the same underlying probability of p (Gordon [1997]). For many practical applications binomial demand distributions are appropriate to model certain fluctuations of the demand around a specific value and therefore a reasonable assumption. Mathematically, the sum of multiple binomial distributions with the same probability parameter p can be expressed as a single binomial distribution, which itself can be decomposed into a sum of identical Bernoulli distributions (Grimmett and Welsh [1986]; Gordon [1997]). The Bernoulli distribution is used to model a stochastic experiment with two different outcomes, success with probability p and failure with probability $q = 1 - p$. Due to its simplicity it is convenient to perform the analyses using the Bernoulli distribution and to reason about the original binomial distributions afterwards. In fact, we can bound the convergence speed with respect to the number of Bernoulli trials and therefore also with respect to the expected demand and the expected number of restockings. In this way we provide results, which are useful for practical applications and for further theoretical investigations.

We model the stochastic process using the Markov chain model introduced in section 2.1 and using multiple identical Bernoulli distributions. In the following the success probability, which represents a demand of 1, is denoted by p and the failure probability, which represents a demand of 0, is denoted by $q = 1 - p$. Omitting entries representing a 0, the transition matrix T associated with one Bernoulli distribution can then be written in the following way.

$$T = \begin{pmatrix} q & p & & & \\ & q & p & & \\ & & \ddots & \ddots & \\ & & & q & p \\ p & & & & q \end{pmatrix}.$$

Now the idea is to focus on powers of the transition matrix T. For that purpose we want to use the eigendecomposition (Golub and Van Loan [1996]; Horn

and Johnson [1990]) of T into a product $T = VDV^H$ of complex matrices V and D, where $V^H = V^{-1}$ and D is a diagonal matrix. Then we can calculate the product T^n easily as $T^n = \left(VDV^H\right)^n = VD^nV^H$. For the eigendecomposition of T, we have to calculate its complex eigenvalues and corresponding eigenvectors. To get the complex eigenvalues we have to calculate the roots of the characteristic polynomial. If T is a $m \times m$ matrix, we can write the characteristic polynomial in the following form.

$$\begin{aligned}
\det(\lambda I - T) &= \det \begin{pmatrix} \lambda-q & -p & & & \\ & \lambda-q & -p & & \\ & & \ddots & \ddots & \\ & & & \lambda-q & -p \\ -p & & & & \lambda-q \end{pmatrix} \\
&= (\lambda-q)\det \begin{pmatrix} \lambda-q & -p & & \\ & \ddots & \ddots & \\ & & \lambda-q & -p \\ & & & \lambda-q \end{pmatrix} \\
&\quad + (-1)^{m+1}(-p)\det \begin{pmatrix} -p & & & \\ \lambda-q & -p & & \\ & \ddots & \ddots & \\ & & \lambda-q & -p \end{pmatrix} \\
&= (\lambda-q)^m + (-1)^{m+1}(-p)^m \\
&= (\lambda-q)^m - p^m
\end{aligned}$$

Using this form we can derive the following term for the different eigenvalues. Note that $\sqrt[m]{1}$ represents all m complex roots of 1.

$$\begin{aligned}
& (\lambda-q)^m - p^m &&= 0 \\
\Leftrightarrow\quad & (\lambda-q)^m &&= p^m \\
\Leftrightarrow\quad & \lambda-q &&= p\sqrt[m]{1} \\
\Leftrightarrow\quad & \lambda &&= q + p\sqrt[m]{1}
\end{aligned}$$

So the eigenvalues are distributed around a circle with center q and radius p. Furthermore 1 is always an eigenvalue with a corresponding eigenvector

$(1,1,\ldots,1)^t$. If we denote the different eigenvalues with $\lambda_1,\lambda_2,\ldots,\lambda_m$ and the corresponding normalized eigenvectors with $v_1,v_2,\ldots,v_m$, we can write V and D as

$$V = \begin{pmatrix} v_1 & v_2 & \ldots & v_m \end{pmatrix},$$

$$D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{pmatrix}.$$

Now we can rewrite the product T^n as

$$\begin{aligned} T^n &= VD^nV^H \\ &= \sum_{i=1}^{m} \left(v_i v_i^H \lambda_i^n \right). \end{aligned}$$

With $\lambda_1 = 1$ and the corresponding normalized eigenvector $v_1 = \left(m^{-1/2}, m^{-1/2}, \ldots, m^{-1/2}\right)$ we have

$$\begin{aligned} T^n &= \sum_{i=1}^{m} \left(v_i v_i^H \lambda_i^n \right) \\ &= \frac{1}{m} J + \sum_{i=2}^{m} \left(v_i v_i^H \lambda_i^n \right), \end{aligned}$$

where J is a matrix of ones. That means we are now able to bound the convergence speed using the error term $\sum_{i=2}^{m} \left(v_i v_i^H \lambda_i^n \right)$.

If we denote the entry in row j and column k of V^n by V_{jk}^n and the j-th entry of v_i by $(v_i)_j$, we can bound the error for each $j,k \in \{1,2,\ldots,m\}$ by

$$\begin{aligned}
\left|1/m - V_{jk}^n\right| &= \left|1/m - \sum_{i=1}^{m} \left(v_i v_i^H \lambda_i^n\right)_{jk}\right| \\
&= \left|\sum_{i=2}^{m} \left(v_i v_i^H \lambda_i^n\right)_{jk}\right| \\
&= \left|\sum_{i=2}^{m} ((v_i)_j \overline{(v_i)_k} \lambda_i^n)\right| \\
&\leq \sum_{i=2}^{m} \left|(v_i)_j \overline{(v_i)_k} \lambda_i^n\right| \\
&\leq \sum_{i=2}^{m} \left|\lambda_i\right|^n \\
&\leq (m-1)\left|\lambda_2\right|^n,
\end{aligned}$$

where λ_2 is the second largest eigenvalue regarding the standard norm in $\mathbb{C}$. We can calculate the norms of the eigenvalues in the following way. Here k is representing the different eigenvalues with values $k \in \{0, 1, \ldots, m-1\}$.

$$\begin{aligned}
|\lambda|^2 &= \left|q + p \cdot e^{(ik2\pi/m)}\right|^2 \\
&= \left|q + p\cos(k2\pi/m) + p\sin(k2\pi/m)i\right|^2 \\
&= (q + p\cos(k2\pi/m))^2 + (p\sin(k2\pi/m))^2 \\
&= q^2 + p^2\cos(k2\pi/m)^2 + 2pq\cos(k2\pi/m) + p^2\sin(k2\pi/m)^2 \\
&= p^2 + q^2 + 2pq\cos(k2\pi/m) \\
&= (p+q)^2 - 2pq(1 - \cos(k2\pi/m)) \\
&= 1 - 2pq(1 - \cos(k2\pi/m))
\end{aligned}$$

The eigenvalue with the largest norm is $\lambda_1 = 1$, the eigenvalues with the second largest norm are $q + p \cdot e^{(i2\pi/m)}$ and $q + p \cdot e^{(-ik2\pi/m)}$. Since the norm of the second largest eigenvalues is $(1 - 2pq(1 - \cos(2\pi/m)))^{1/2}$ and thus a strictly positive constant smaller than one, we have finally a linear rate of convergence (Schatzman and Taylor [2002]). For the vehicle routing problems with stochastic demands under investigation, this result means that the difference with respect to the uniform distribution is geometrically decreasing in the cumulative expected demand of goods.

2.4 Discussion

In this section we discuss the impacts of the convergence results, especially regarding practical applications. We also show that with these new theoretical results some observations in literature can be explained.

The first question is, what the results mean for the VRPSD and the VRPSDC with the basic restocking strategy. In section 2.2 we have indicated that the transition matrices of the VRPSD and the VRPSDC can be identified with the matrices of theorem 2. That means if we would have an instance with infinitely many customers (or we would just loop over a finite number of customers), and if the series of transition matrices fulfills the conditions of theorem 2, the distribution of goods in the vehicle converges to the uniform distribution. Now we have to clarify two different things in order to transfer the theoretical results to real world problems. At first, under which conditions does the series of transition matrices fulfill the conditions of theorem 2 and secondly, what happens if we have only a finite number of customers.

The gcd property looks quite technical at first glance. But for practical instances it is usually fulfilled by the transition matrices, or at least by many of them, which would be sufficient to apply our theoretical results. In the previous section we pointed out that the gcd property is fulfilled for a matrix, if it has two strictly positive entries next to each other in the first column. That means for the problems we investigate, that the demand distribution for a customer contains two demands with strictly positive probability next to each other, and this is not a very strong assumption. It is for example fulfilled if the demand distribution is a binomial distribution. On the other hand, the gcd property is not fulfilled, if the differences between the possible demands of a certain customer and the capacity of the vehicle are all a multiple of some integer $k > 1$. This could be due to product specific constraints, e.g. it is only possible to order one package of a product consisting of k units. But in this case the problem can be reduced by examining packages instead of actual units. If it is not due to product specific constraints and if the differences of the possible demands for every customer are multiples of (potentially different) integers greater than 1, the conditions of 2 are not fulfilled, but fortunately this is a really artificial case, usually not occurring in real world problems.

The other point is that we have not an infinite number of customers in real world instances. In section 2.3 we have shown that the difference of the distribution of goods in the vehicle with respect to the uniform distribution is decreasing geometrically in the total expected demand up to a specific customer, if the demand distributions are binomial distributions of the same type. For instances

with a finite number of customers, we know that in the general case the amount of goods approaches the uniform distribution. Preliminary experiments show that for instances which require multiple restocking actions the distribution of goods in the vehicle after performing a few restockings is quite close to the uniform distribution. In this case the probability that a restocking action is necessary at a certain customer depends almost only on the distribution of goods of this customer and only to a small amount on the actual tour. On the other hand, for instances which require only a few restocking actions the convergence is far slower, but in this case the costs of the restocking actions are small compared to the travel costs between customers. In both cases the results underlay our thesis that the VRPSD and the TSP, and the VRPSDC and the PTSP are quite similar.

The second question is, how these theoretical results can be used for the development of good algorithms for the VRPSD and the VRPSDC. A first idea could be to use existing algorithms for the TSP and the PTSP on instances for the VRPSD and the VRPSDC. This approach is used in the so called *cyclic heuristic*, which will be discussed later in this section. Another idea would be to use the theoretical results for the development of good approximations for the solution costs. For example the beginning of the tour could be evaluated by an exact evaluation approach for the VRPSD and the VRPSDC, and after a certain point, the costs could be evaluated by assuming the uniform distribution for the amount of goods in the vehicle. Further research at this point seems promising.

Additionally, our theoretical results give explanations for some observations in literature. In Gendreau et al. [1996a] the authors observed that "One interest of these studies is to show that stochastic customers are a far more complicating factor than stochastic demands". This is in accordance with our results in some sense. The introduction of stochastic demands does not change the problem significantly, while the introduction of stochastic customers changes the TSP into the PTSP which usually cannot be tackled in the same way, especially if the probabilities for visiting the customers are rather low.

In Haimovich and Rinnooy Kan [1985] the authors introduce the so called *cyclic heuristic*, which is used for theoretical analyses in Bertsimas [1992]. The *cyclic heuristic* starts with a solution for the TSP or the PTSP. It inserts the depot at all possible locations and takes the best tour with respect to the solution cost of the problem to be solved. Our results give again an explanation for the practical success of this heuristic. Using the *cyclic heuristic* over a good TSP solution usually yields a good solution for the VRPSD, and using the *cyclic heuristic* over a good PTSP solution usually yields a good solution for the VRPSDC.

An approximation for the difference of the quality of two solutions for the VRPSD and the VRPSDC is introduced in Bianchi et al. [2006]. Here the au-

thors implicitly assume a uniform distribution over the amount of goods in the vehicle and they show that the use of that approximation leads to good results in practice. According to our theoretical results the assumption of the uniform distribution is reasonable and so we can give a theoretical explanation for the success of this method.

2.5 Conclusions

In this chapter we have derived theoretical results which show that the VRPSD and the VRPSDC are under some moderate conditions quite similar to the TSP and the PTSP, respectively. In many settings the introduction of stochastic demand values does not affect the problems too much. These results also generalize to several variants of vehicle routing problems with stochastic demands and in particular to those problems that use multiple vehicles. Maybe analog results can be obtained for other stochastic combinatorial optimization problems. In this case real world problems can be modeled in a more realistic way using stochasticity in their formulation, while well established algorithms and techniques can be used to solve them. Perhaps it is also possible to transfer or improve approximation results from the underlying non stochastic problems to the variants including stochasticity, or even to develop new optimization algorithms directly based on theoretical results, like those presented in this chapter.

Chapter 3

Hardness Results for Stochastic Vehicle Routing Problems on Substantially Stochastic Instances

Many stochastic vehicle routing problems are trivially NP-hard as generalizations of non-stochastic NP-hard problems. Unfortunately, such results completely ignore the possibility of those problems to model certain aspects in a stochastic way. And in fact, modeling certain aspects in a stochastic way is the reason for using stochastic vehicle routing problems and the main difference to non-stochastic vehicle routing problems. Therefore, it is of great importance to investigate the impact of stochastic input on the computational complexity of those problems. And this is what we are doing in this chapter for different stochastic vehicle routing problems.

We start with examining stochastic instances for the PROBABILISTIC TRAVELING SALESMAN PROBLEM and stochastic instances for the METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM, that means for instances where the triangle inequality holds for the distances. For the instances used in this case the probabilities for the customers' presence are very close to 1 and so they do not differ a lot from non-stochastic instances. Therefore, we continue our investigations with what we denote as *substantially* stochastic instances for the METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM. The probabilities used in these instances are all 1/2 and therefore these instances differ a lot from non-stochastic instances. We then extend our investigations with *substantially* stochastic instances for the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS. At the end we show how the results obtained for the PTSP and the VRPSD can be generalized to the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS. All the results presented in this chapter are based on Weyland et al. [2011a].

3.1 The Probabilistic Traveling Salesman Problem

In this section we give three reductions to stochastic instances of the PROBABILISTIC TRAVELING SALESMAN PROBLEM. First, we show with a reduction from the HAMILTONIAN CYCLE PROBLEM (Karp [1972]) to stochastic instances of the PROBABILISTIC TRAVELING SALESMAN PROBLEM, that it is NP-hard to compute a 2^n approximation for stochastic instances of the PTSP. After that we show by a reduction from the METRIC TRAVELING SALESMAN PROBLEM to stochastic instances of the METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM, that it is NP-hard to compute $\frac{117}{116} - \varepsilon$ approximations for stochastic instances of the METRIC PTSP with asymmetric distances and $\frac{220}{219} - \varepsilon$ for stochastic instances of the METRIC PTSP with symmetric distances. Both reductions use instances of the PTSP, where the probabilities are close to 1. In the third reduction we generalize the second reduction to what we denote as *substantially* stochastic instances, where all the probabilities used are $1/2$. Note that these instances are very different from non-stochastic instances with respect to the stochastic input.

3.1.1 Stochastic Instances of the PTSP

Let us start with the formal definition of the well-known HAMILTONIAN CYCLE PROBLEM. For a given undirected graph $G = (V, E)$ with vertex set V and edge set E, the problem is to decide whether there exists a cycle visiting all the vertices exactly once or not. More formal we can state it as follows.

Problem 5 (HAMILTONIAN CYCLE PROBLEM (HC))**.** *Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, the problem is to decide whether there exists a simple cycle of length $|V|$ or not.*

We now present a polynomial reduction from the HAMILTONIAN CYCLE PROBLEM to stochastic instances of the PROBABILISTIC TRAVELING SALESMAN PROBLEM. Given an instance (V, E) for the HC with $|V| = n$ vertices, we create an instance for the PTSP in the following way. We use V as the set of customers for the PTSP instance. For each edge $\{v, w\} \in E$, we set the distance between v and w and between w and v to 1, while all other distances are set to $n2^n$. For each customer we set the probability that this customer requires a visit to $1 - 2^{-3n}$. Note that all of the customers require a visit in a particular realization of the stochastic events with a probability of at least $1 - n2^{-3n}$ due to the union bound (also called Boole's inequality, cf. Grimmett and Welsh [1986]). We begin with some simple properties between solutions for the HC instance and the PTSP instance.

Lemma 4. *Let τ be any solution for the PTSP, then the following two implications hold.*

(i) If τ is a Hamiltonian cycle for the given HC instance, then

$$f_{ptsp}(\tau) \leq \left(1 - n2^{-3n}\right) n + \left(n2^{-3n}\right) n^2 2^n.$$

(ii) If τ is not a Hamiltonian cycle for the given HC instance, then

$$f_{ptsp}(\tau) \geq \left(1 - n2^{-3n}\right) (n2^n + n - 1).$$

Proof. (i): Here we use the fact, that with a probability of at least $1 - n2^{-3n}$ all customers have to be visited, which results in costs of n, since the tour is a Hamiltonian cycle. Otherwise, that means with a probability of at most $n2^{-3n}$, the costs can be trivially bounded by the costs of a maximum cost cycle in the PTSP instance. Such a cycle contains at most n edges with costs of at most $n2^n$ for each of the edges and therefore the costs for a maximum cost cycle can be bounded by $n^2 2^n$. This gives us the desired bound.

(ii): In this case we only consider the costs for the a posteriori tour, in which all customers are visited. The probability for this event is at least $1 - n2^{-3n}$ and the costs for such a tour are bounded from below by $n2^n + n - 1$, since at least one of the expensive distances of $n2^n$ is used. This implies the second bound. □

Lemma 5. *Let τ be an optimal solution for the PTSP instance. Then the following two implications hold.*

(i) If there exists a Hamiltonian cycle for the given HC instance, then

$$f_{ptsp}(\tau) \leq \left(1 - n2^{-3n}\right) n + \left(n2^{-3n}\right) n^2 2^n.$$

(ii) If there does not exist a Hamiltonian cycle for the given HC instance, then

$$f_{ptsp}(\tau) \geq \left(1 - n2^{-3n}\right) (n2^n + n - 1).$$

Proof. (i): If there exists a Hamiltonian cycle δ, we can bound the costs of an optimal solution using lemma 4 by

$$f_{ptsp}(\tau) \leq f_{ptsp}(\delta) \leq \left(1 - n2^{-3n}\right) n + \left(n2^{-3n}\right) n^2 2^n.$$

(ii): If there does not exist a Hamiltonian cycle, due to lemma 4 the costs of all solutions are bounded from below by $\left(1 - n2^{-3n}\right) (n2^n + n - 1)$, and therefore in particular the costs of τ. □

Using these properties we are now able to prove that it is NP-hard to compute a 2^n approximation for stochastic instances of the PTSP.

Theorem 3. *It is NP-hard to compute a 2^n approximation for stochastic instances of the* PROBABILISTIC TRAVELING SALESMAN PROBLEM.

Proof. We show that we can solve the HC problem using a 2^n approximation algorithm for stochastic instances of the PTSP. Since the reduction runs in polynomial time with respect to the size of the HC instance, this implies the desired result.

Note that for sufficiently large n we have

$$\begin{aligned} & 2^n \left[\left(1 - n2^{-3n}\right) n + \left(n2^{-3n}\right) n^2 2^n \right] \\ = \; & n2^n \left(1 - n2^{-3n}\right) + n^3 2^{-n} \\ < \; & n2^n \left(1 - n2^{-3n}\right) + (n-1)\left(1 - n2^{-3n}\right) \\ = \; & \left(1 - n2^{-3n}\right)(n2^n + n - 1). \end{aligned}$$

Given a HC instance (with n sufficiently large), we use the reduction introduced in this section and apply the 2^n approximation algorithm for stochastic instances of the PTSP. If there exists a Hamiltonian cycle, the costs of an optimal solution are bounded from above by $\left(1 - n2^{-3n}\right) n + \left(n2^{-3n}\right) n^2 2^n$ due to lemma 5. Therefore, the costs of a 2^n approximation are bounded from above by $2^n \left[\left(1 - n2^{-3n}\right) n + \left(n2^{-3n}\right) n^2 2^n \right] < \left(1 - n2^{-3n}\right)(n2^n + n - 1)$. On the other hand, if there does not exist a Hamiltonian cycle, then the costs of an optimal solution are bounded from below by $\left(1 - n2^{-3n}\right)(n2^n + n - 1)$ due to lemma 5, and therefore the costs of any 2^n approximation are also bounded from below by $\left(1 - n2^{-3n}\right)(n2^n + n - 1)$.

Now it is easy to decide, whether a Hamiltonian cycle exists or not. We only have to evaluate the costs of the solution computed by the 2^n approximation algorithm. This can be performed in polynomial time. If those costs are below $\left(1 - n2^{-3n}\right)(n2^n + n - 1)$, then there exists a Hamiltonian cycle, if they are at least $\left(1 - n2^{-3n}\right)(n2^n + n - 1)$, then there is no Hamiltonian cycle. □

3.1.2 Stochastic Instances of the Metric PTSP

In this section we derive hardness results for stochastic instances of the METRIC PTSP. For this purpose we use a reduction from the famous METRIC TRAVELING SALESMAN PROBLEM. The task for the TRAVELING SALESMAN PROBLEM is to compute for a given graph a simple cycle of minimum length with respect to the distances

given on the edges. Let V be a set of n vertices, let $d : V \times V \to \mathbb{R}^+$ be a function representing the distances between different vertices and let $\tau : \langle n \rangle \to V$ be a permutation representing a simple cycle of length n. Then the costs of this cycle can be written as

$$f_{\text{tsp}}(\tau) = \sum_{i=1}^{n-1} d(\tau_i, \tau_{i+1}) + d(\tau_n, \tau_1).$$

The TRAVELING SALESMAN PROBLEM can now be formally stated as follows.

Problem 6 (TRAVELING SALESMAN PROBLEM (TSP))**.** *Given a set V of n vertices and function $d : V \times V \to \mathbb{R}^+$, the problem is to compute a permutation $\tau^\star : \langle n \rangle \to V$, such that $f_{tsp}(\tau^\star) \leq f_{tsp}(\tau)$ for any permutation $\tau : \langle n \rangle \to V$.*

If we additionally require that the distance function $d : V \times V \to \mathbb{R}^+$ obeys the triangle inequality, we obtain the METRIC TRAVELING SALESMAN PROBLEM.

Given an instance for the METRIC TSP we create an instance for the METRIC PTSP in the following way. We use the same set of customers and the same distance function. Additionally, we set the probability that a customer requires a visit for each customer to $1 - n^{-k}$ for some $k \in \mathbb{N}, k \geq 2$. Note that all of the customers require a visit in a particular realization of the stochastic events with a probability of at least $1 - n^{-k+1}$ due to the union bound (Grimmett and Welsh [1986]). We start with proving two inequalities between the objective functions for the TSP and the PTSP, applied to the same solution.

Lemma 6. *Let τ be any solution for the TSP or the PTSP, respectively. Then we have the following two inequalities:*

(i) $f_{ptsp}(\tau) \leq f_{tsp}(\tau)$

(ii) $f_{tsp}(\tau) \leq \frac{1}{1-n^{-k+1}} f_{ptsp}(\tau)$

Proof. The first inequality follows directly from the triangle inequality on the distances, since skipping a customer can only result in shorter tours as long as the triangle inequality holds, and therefore the expected length of the a posteriori tour is always bounded by the length of the a priori tour, which is in fact the length of the TSP tour.

The second inequality can be obtained by bounding the cost of the PTSP solution only considering the tour, where all customers require a visit. We have for any solution τ

$$\begin{aligned} f_{ptsp}(\tau) &\geq \mathbb{P}(\text{all customers require a visit}) f_{tsp}(\tau) \\ &\geq \left(1 - n^{-k+1}\right) f_{tsp}(\tau) \end{aligned}$$

and therefore $f_{tsp}(\tau) \leq \frac{1}{1-n^{-k+1}} f_{ptsp}(\tau)$ as claimed. □

Using those two inequalities, we can prove the following connection between solution costs for the TSP and the PTSP.

Proposition 1. *If τ is an α approximation for the PTSP, then τ is an $\alpha \frac{1}{1-n^{-k+1}}$ approximation for the TSP.*

Proof. We have the following chain of inequalities due to lemma 6. Here τ_{tsp} is an optimal solution for the TSP and τ_{ptsp} is an optimal solution for the PTSP.

$$\begin{aligned} f_{tsp}(\tau) &\leq \frac{1}{1 - n^{-k+1}} f_{ptsp}(\tau) \\ &\leq \alpha \frac{1}{1 - n^{-k+1}} f_{ptsp}(\tau_{ptsp}) \\ &\leq \alpha \frac{1}{1 - n^{-k+1}} f_{ptsp}(\tau_{tsp}) \\ &\leq \alpha \frac{1}{1 - n^{-k+1}} f_{tsp}(\tau_{tsp}) \end{aligned}$$

□

Since the parameter $k \in \mathbb{N}$ can be set to any value ≥ 2, an $\alpha - \varepsilon$ approximation for the PTSP gives us an α approximation for the TSP. It has been shown in Papadimitriou and Vempala [2006] that it is NP-hard to compute a $\frac{117}{116}$ approximation for the METRIC TSP with asymmetric distances and a $\frac{220}{219}$ approximation for the METRIC TSP with symmetric distances. Therefore, it is also NP-hard to compute a $\frac{117}{116} - \varepsilon$ approximation for the METRIC PTSP with asymmetric distances and a $\frac{220}{219} - \varepsilon$ approximation for the METRIC PTSP with symmetric distances, even if only stochastic instances are considered. We summarize the results in the following two theorems.

Theorem 4. *It is NP-hard to compute a $\frac{117}{116} - \varepsilon$ approximation for stochastic instances of the* METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM *with asymmetric distances.*

Theorem 5. *It is NP-hard to compute a* $\frac{220}{219} - \varepsilon$ *approximation for stochastic instances of the* METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM *with symmetric distances.*

3.1.3 Substantially Stochastic Instances of the Metric PTSP

In the previous section we used PTSP instances where all probabilities were at least polynomially close to 1. In this section we want to reduce the METRIC TSP to the METRIC PTSP, where each vertex has a probability of exactly $1/2$. These instances differ a lot from non-stochastic instances with respect to the stochastic input and we informally refer to such instances as *substantially* stochastic instances.

Given an instance for the TSP with n customers, we create an instance for the PTSP in the following way. For each customer we create $p(n)$ identical copies, where $p(n)$ is some polynomial in n. Then we assign a probability of $1/2$ to each of the customers in the resulting PTSP instance. Here we cannot immediately prove inequalities analog to the previous section. The problem is that the customers that belong to the same group of identical copies are not necessarily grouped together in a solution for the PTSP. We show that we can transform any solution for the PTSP in polynomial time into a solution, where all customers that belong to the same group of identical copies are visited consecutively, without increasing the solution costs. After that we use this fact to continue analog to the previous section.

First, we introduce some notational conventions. Let V be the set of customers of the given TSP instance and let W be the set of customers of the PTSP instance. The function $g : W \to V$ assigns to each customer of the PTSP instance the corresponding customer of the original TSP instance. Given a solution τ for the PTSP, we call a number of k successive customers $\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}$ a group, if they are all copies of the same customer of the TSP instance and if the customers τ_{i-1} and τ_{i+k+1} (if they exist) belong to different groups, that means, if the following properties hold. Note that the indices for (ii) and (iii) are taken modulo the number of vertices to allow an easier notation.

(i) $\forall x, y \in \{i, i+1, \ldots, i+k\} : g(\tau_x) = g(\tau_y)$

(ii) $i > 1 \Rightarrow g(\tau_{i-1}) \neq g(\tau_i)$

(iii) $i < n-k \Rightarrow g(\tau_{i+k+1}) \neq g(\tau_{i+k})$

It is clear that the order of customers within a group does not effect the solution quality. Furthermore, a group can be treated like a single customer, assigning a proper probability for the event that a visit is required. The minimum number of groups for a PTSP solution is n, since we have copies of all n different customers of the TSP instance. Since the number of customers in the PTSP instance is $np(n)$, the maximum number of groups is also $np(n)$ and thus polynomially bounded. Now we show that we can efficiently transform a solution with k, $k > n$ groups to a solution with at most $k-1$ groups, without increasing the solution cost.

Lemma 7. *Let τ be a solution for the PTSP instance, containing k, $k > n$, groups. Then we can compute in polynomial time a solution τ' with the following properties:*

(i) *τ' contains at most $k-1$ groups.*

(ii) $f_{ptsp}(\tau') \leq f_{ptsp}(\tau)$

Proof. Let X and Y be the sets of customers that belong to two different groups with the same corresponding customer in the TSP instance. We create two new solutions $\tau_{X\to Y}$ and $\tau_{Y\to X}$, where the two groups are merged at the location of the customers of Y and X, respectively. Let S be the set of all possible scenarios, which are defined by different realizations of the random events. We can partition S into

$$S = S_{X,Y} \cup S_{\bar{X},Y} \cup S_{X,\bar{Y}} \cup S_{\bar{X},\bar{Y}},$$

where a subscript of X indicates the set of scenarios where at least one of the customers of X requires a visit, whereas a subscript of $\bar{X}$ indicates the scenarios where none of the customers of X requires a visit. This holds analogously for the subscripts Y and $\bar{Y}$. Note that all those sets are disjoint and we have in fact a partition. If we shift the customers of X next to the customers of Y, the scenarios contained in $S_{\bar{X},Y}$ and $S_{\bar{X},\bar{Y}}$ lead to the same a posteriori tours. Due to the triangle inequality scenarios in $S_{X,Y}$ lead to a posteriori tours which do not increase in their length. Let p_X denote the probability that at least one customer of X requires a visit and let p_Y denote the probability that at least one customer of Y requires a visit. For any set T of scenarios let $f^T_{\text{ptsp}}(\tau)$ denote the costs of solution τ considering the scenarios in T weighted with the probabilities that they occur. This means that for every partition $S = T_1 \cup T_2 \cup \ldots \cup T_m$ of S the expected cost of the a posteriori solution can be expressed by $f^S_{\text{ptsp}}(\tau) =$

$\sum_{i=1}^{m} f_{\text{ptsp}}^{T_i}(\tau)$. Then we can bound the change in the solution cost between $\tau_{X\to Y}$ and τ by

$$\begin{aligned}
& f_{\text{ptsp}}^{S}(\tau_{X\to Y}) - f_{\text{ptsp}}^{S}(\tau) \\
= \; & f_{\text{ptsp}}^{S_{X,Y}}(\tau_{X\to Y}) + f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau_{X\to Y}) + f_{\text{ptsp}}^{S_{\bar{X},Y}}(\tau_{X\to Y}) + f_{\text{ptsp}}^{S_{\bar{X},\bar{Y}}}(\tau_{X\to Y}) \\
& - f_{\text{ptsp}}^{S_{X,Y}}(\tau) - f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau) - f_{\text{ptsp}}^{S_{\bar{X},Y}}(\tau) - f_{\text{ptsp}}^{S_{\bar{X},\bar{Y}}}(\tau) \\
= \; & f_{\text{ptsp}}^{S_{X,Y}}(\tau_{X\to Y}) + f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau_{X\to Y}) - f_{\text{ptsp}}^{S_{X,Y}}(\tau) - f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau) \\
\leq \; & f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau_{X\to Y}) - f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau) \\
\leq \; & f_{\text{ptsp}}^{S_{\bar{X},Y}}(\tau_{X\to Y})\frac{p_X(1-p_Y)}{(1-p_X)p_Y} - f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau) \\
\leq \; & f_{\text{ptsp}}^{S_{\bar{X},Y}}(\tau)\frac{p_X(1-p_Y)}{(1-p_X)p_Y} - f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau).
\end{aligned}$$

Here the last two transformations require some more detailed explanations. As we stated earlier, we can consider the set X as a single customer with visiting probability p_X and the set Y as a single customer with visiting probability p_Y. The scenarios $S_{X,\bar{Y}}$ and $S_{\bar{X},Y}$ lead to the same a posteriori tours for the solution $\tau_{X\to Y}$. The probabilities for the scenarios in $S_{X,\bar{Y}}$ contain the two factors p_X and $1-p_Y$ (included in $f_{\text{ptsp}}^{S_{X,\bar{Y}}}$), which are not present in the probabilities for the scenarios in $S_{\bar{X},Y}$. Analog the scenarios in $S_{\bar{X},Y}$ contain the two factors $1-p_X$ and p_Y, which are not present in the probabilities for the scenarios in $S_{X,\bar{Y}}$. This explains the factor of $\frac{p_X(1-p_Y)}{(1-p_X)p_Y}$ in the second last transformation. Since for all scenarios in $S_{\bar{X},Y}$ the a posteriori tours for the solutions $\tau_{X\to Y}$ and τ are the same, we can replace $\tau_{X\to Y}$ by τ in this case, which explains the last transformation.

Analog we get for the change in the solution cost between $\tau_{Y\to X}$ and τ

$$\begin{aligned}
& f_{\text{ptsp}}^{S}(\tau_{Y\to X}) - f_{\text{ptsp}}^{S}(\tau) \\
\leq \; & f_{\text{ptsp}}^{S_{X,\bar{Y}}}(\tau)\frac{(1-p_X)p_Y}{p_X(1-p_Y)} - f_{\text{ptsp}}^{S_{\bar{X},Y}}(\tau)
\end{aligned}$$

Now, if $f_{\text{ptsp}}^{S}(\tau_{X\to Y}) - f_{\text{ptsp}}^{S}(\tau) \leq 0$, we set $\tau' := \tau_{X\to Y}$. Otherwise, we have

$$
\begin{aligned}
& f^S_{\text{ptsp}}(\tau_{X\to Y}) - f^S_{\text{ptsp}}(\tau) > 0 \\
\Rightarrow\ & f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau)\frac{p_X(1-p_Y)}{(1-p_X)p_Y} - f^{S_{X,\bar{Y}}}_{\text{ptsp}}(\tau) > 0 \\
\Rightarrow\ & f^{S_{X,\bar{Y}}}_{\text{ptsp}}(\tau) < f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau)\frac{p_X(1-p_Y)}{(1-p_X)p_Y}
\end{aligned}
$$

Inserting this into the bound for $f^S_{\text{ptsp}}(\tau_{Y\to X}) - f^S_{\text{ptsp}}(\tau)$ we obtain

$$
\begin{aligned}
f^S_{\text{ptsp}}(\tau_{Y\to X}) - f^S_{\text{ptsp}}(\tau) &\leq f^{S_{X,\bar{Y}}}_{\text{ptsp}}(\tau)\frac{(1-p_X)p_Y}{p_X(1-p_Y)} - f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau) \\
&< f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau)\frac{p_X(1-p_Y)}{(1-p_X)p_Y}\frac{(1-p_X)p_Y}{p_X(1-p_Y)} - f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau) \\
&< f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau) - f^{S_{\bar{X},Y}}_{\text{ptsp}}(\tau) \\
&< 0
\end{aligned}
$$

and set $\tau' := \tau_{Y\to X}$. In both cases the new solution τ' contains at most $k-1$ groups and does not increase the solution cost compared to τ. Therefore, the properties (i) and (ii) of the claim are fulfilled. Furthermore, we can create and evaluate the solutions $\tau_{X\to Y}$ and $\tau_{Y\to X}$ in polynomial time. Hence the total computational time is polynomially bounded, which concludes the proof. □

Applying this lemma consecutively, we obtain the following result.

Proposition 2. *Let τ be a solution for the PTSP instance. Then we can compute in polynomial time a solution τ' with the following properties:*

(i) τ' contains n groups.

(i) $f_{ptsp}(\tau') \leq f_{ptsp}(\tau)$

Proof. Since the number of groups for any solution is bounded from above by $np(n)$, we can yield a solution τ' with the required properties by applying lemma 7 for at most $np(n) - n$ times. We are performing polynomially many steps, which all take polynomial time, and therefore the total computational time is also polynomially bounded. □

We are now able to prove some inequalities about the relation between the two evaluation functions for the TSP and for the PTSP, analog to the previous section.

Lemma 8. *Let τ be a solution for the PTSP. Then we can compute in polynomial time a solution τ' for the PTSP, which contains only n groups, and a solution τ'' for the TSP, with the following properties:*

$$f_{tsp}(\tau'') \leq \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau') \leq \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau)$$

Proof. The second inequality follows immediately from proposition 2. It remains to show that we can compute a solution τ'' for the TSP in polynomial time, starting with the solution τ' for the PTSP, such that the first inequality holds. The n groups of τ' give us an ordering of the n customers in the TSP instances. We use this order to create a TSP instance τ''. If at least one customer in each of the n groups of τ' requires a visit, the length of the a posteriori tour equals the length of the TSP tour defined by τ''. Using this fact, we can bound the cost of the PTSP solution τ' by

$$\begin{aligned} f_{ptsp}(\tau') &\geq \mathbb{P}(\text{at least one customer in each group requires a visit}) f_{tsp}(\tau'') \\ &\geq \left(1 - n2^{-p(n)}\right) f_{tsp}(\tau'') \end{aligned}$$

and therefore we have $f_{tsp}(\tau'') \leq \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau')$ which proves the first inequality in the claim. □

Lemma 9. *Let τ be a solution for the TSP and let $\tau^\star$ be a solution for the PTSP containing n groups with $p(n)$ customers in each group, preserving the ordering of the TSP solution. Then the following inequality holds:*

$$f_{ptsp}(\tau^\star) \leq f_{tsp}(\tau)$$

Proof. Due to the triangle inequality the length of every possible a posteriori tour, which is derived from $\tau^\star$ for the PTSP, is bounded from above by the length of the TSP tour defined by τ. □

Using those two inequalities, we can establish the following connection between solution costs for the TSP and the PTSP.

Proposition 3. *If τ is an α approximation for the PTSP, then we can compute in polynomial time an $\alpha \frac{1}{1-n2^{-p(n)}}$ approximation τ'' for the TSP.*

Proof. Due to lemma 8 and lemma 9 we can compute in polynomial time solutions τ' for the PTSP and τ'' for the TSP, such that the following chain of inequalities hold. Here τ_{tsp} is an optimal solution for the TSP and τ_{ptsp} is an optimal solution for the PTSP. $\tau^{\star}_{tsp}$ denotes the PTSP solution derived from τ_{tsp} according to lemma 9.

$$\begin{aligned} f_{tsp}(\tau'') &\leq \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau') \\ &\leq \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau) \\ &\leq \alpha \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau_{ptsp}) \\ &\leq \alpha \frac{1}{1-n2^{-p(n)}} f_{ptsp}(\tau^{\star}_{tsp}) \\ &\leq \alpha \frac{1}{1-n2^{-p(n)}} f_{tsp}(\tau_{tsp}) \end{aligned}$$

□

Since we can use any polynomial $p(n)$, an $\alpha - \varepsilon$ approximation for the PTSP gives us an α approximation for the TSP. As we mentioned in the previous section, it has been shown in Papadimitriou and Vempala [2006] that it is NP-hard to compute a $\frac{117}{116}$ approximation for the METRIC TSP with asymmetric distances and a $\frac{220}{219}$ approximation for the METRIC TSP with symmetric distances. Therefore, it is also NP-hard to compute a $\frac{117}{116} - \varepsilon$ approximation for the METRIC PTSP with asymmetric distances and a $\frac{220}{219} - \varepsilon$ approximation for the METRIC PTSP with symmetric distances, even if only *substantially* stochastic instances are considered. We summarize these results in the following two theorems.

Theorem 6. *It is NP-hard to compute a $\frac{117}{116} - \varepsilon$ approximation for **substantially** stochastic instances of the* METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM *with asymmetric distances.*

Theorem 7. *It is NP-hard to compute a $\frac{220}{219} - \varepsilon$ approximation for **substantially** stochastic instances of the* METRIC PROBABILISTIC TRAVELING SALESMAN PROBLEM *with symmetric distances.*

3.2 The Vehicle Routing Problem with Stochastic Demands

In this section we give a reduction from the METRIC TRAVELING SALESMAN PROBLEM to *substantially* stochastic instances of the METRIC VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS. These instances are using binomial demand distributions. We show with this reduction that retrieving an optimal solution for *substantially* stochastic instances of this problem is NP-hard. In chapter 2 we have shown that the distribution of goods in the vehicle converges to the uniform distribution under some mild conditions. Once approaching the uniform distribution, the probability that a restocking action is necessary after serving a specific customer depends almost entirely on the demand distribution of that customer and only slightly on the tour. The overall idea is to introduce some dummy nodes and to enforce that they have to be served at the very beginning of the tour in an optimal VRPSD solution. After that, the distribution of goods in the vehicle has approached the uniform distribution and the remaining customers are served in a tour which is very close to an optimal TSP tour.

Given an instance for the METRIC TSP with n vertices, we construct an instance for the METRIC VRPSD in the following way. All the vertices of the TSP instance are used as customers in the VRPSD instance, while one arbitrary vertex is selected as the depot. The same distances are used in the new instance. Additionally, we introduce $p(n)$ dummy customers, where $p(n)$ is some polynomial in n, which will be specified later. Let $L := 2^{3Q+1}nl$, where l is the longest distance in the original TSP instance. The distances between the dummy customers are all 0, while the distance between any dummy customer and the depot is L and the distance between any dummy customer and any of the other customers is L plus the distance from the depot to that customer. We can express this in a more formal way. Given an instance for the TSP by a set $V = \{v_1, v_2, \ldots, v_n\}$ of $n \in \mathbb{N}$ cities and a distance function $d : V \times V \to \mathbb{R}^+$, we construct the following instance for the VRPSD. We use $V' = V \cup W$ as the set of nodes for the VRPSD instance, where $W = \{w_1, w_2, \ldots, w_{p(n)}\}$ represents the additional dummy customers. We set the depot to v_1 and we extend d to a distance function $d' : V' \times V' \to \mathbb{R}^+$ on V' by setting

$$d'(x,y) = \begin{cases} d(x,y) & x,y \in V \\ L + d(v_1,y) & x \in W, y \in V \\ d(x,v_1) + L & x \in V, y \in W \\ 0 & x,y \in W \end{cases}$$

Moreover, we set the vehicle capacity to some constant $Q \in \mathbb{N}$, which will also be specified later. We set the demand distributions of the customers $V' \setminus \{v_1\}$ to a $B(Q, 1/2)$ binomial distribution, which is itself a sum of Q Bernoulli trials with success probability $1/2$. For details on these distributions we refer to chapter 2.

It is obvious that the reduction can be performed in polynomial time. In the remaining part of this section we show that we can retrieve a very good approximation for the given TSP instance, if we have an optimal solution for the newly-created VRPSD instance. We will see that retrieving such an approximation for the TSP is NP-hard and therefore getting an optimal solution for the VRPSD is NP-hard, even if we focus on inherent stochastic instances. We start with two basic properties, which are used afterwards.

Lemma 10. *The smallest probability occurring in the binomial demand distribution can be bounded from below by 2^{-Q}. This also holds for any sum of the binomial demand distribution.*

Proof. The probabilities that occur in the binomial demand distributions are $\binom{Q}{k}2^{-Q}$, $k \in \{0, 1, \ldots, Q\}$. Those probabilities are at least 2^{-Q}, in fact this value is attained for $k = 0$ and $k = Q$. The second statement follows from the fact that those probabilities are convex combinations of the probabilities for a single binomial distribution. □

Since the total demand of the first i customers modulo Q is Q minus the vehicle load after the first i customers, we get the following helpful result.

Lemma 11. *The probability that the total demand of any number of consecutive customers modulo Q equals k, $k \in \{0, 1, \ldots, Q-1\}$, is at least 2^{-Q}.*

Proof. This follows directly from lemma 10. □

Now we show that the dummy customers have to be visited first in an optimal VRPSD tour. In particular we show the following two facts. At first, we show that the dummy customers have to be visited subsequently in an optimal tour. Then we show that this block of dummy customers has to be visited at the beginning of an optimal tour.

Proposition 4. *Given a tour τ, with at least two non successive blocks of dummy customers, we can create a tour τ' with the following properties:*

(i) The number of non successive blocks of dummy customers in τ' is decreased by at least 1.

(ii) $f_{vrpsd}(\tau') < f_{vrpsd}(\tau)$.

Proof. We show that merging the first two blocks of dummy customers in the tour τ at the position of the first block, yields a tour τ' with strictly less costs. The costs for τ can be bounded by

$$\begin{aligned} f_{vrpsd}(\tau) &\geq L + 2e_1L + L + L + 2e_2L + L + c_{\text{end}} \\ &= 4L + e_1 2L + e_2 2L + c_{\text{end}}. \end{aligned}$$

Here we only consider the travel costs to both blocks of dummy nodes, the expected restocking costs while serving the first block of dummy nodes, the expected restocking costs while serving the second block of dummy nodes and the costs for the other dummy nodes which occur after the second block of dummy nodes has been fully served. In this formula e_1 denotes the expected number of restockings while serving the first block of dummy nodes and e_2 denotes the expected number of restockings while serving the second block of dummy nodes. Note that the case in which the vehicle gets empty after serving one of the blocks of dummy nodes is included in the travel costs and not in the expected restocking costs. c_{end} denotes the costs caused by the dummy nodes, which are not in one of the two blocks.

Similarly we can bound the costs for τ'.

$$\begin{aligned} f_{vrpsd}(\tau') &\leq L + 2eL + L + 3nl + c_{end} \\ &= 2L + e2L + 3nl + c_{end}. \end{aligned}$$

Here we consider the travel costs of the merged block of dummy nodes, the expected restocking costs while serving this block of dummy nodes, the costs of the dummy nodes which occur after the old position of the second block of dummy nodes (which are identical to the costs in τ and again denoted by c_{end}) and the travel and restocking costs for all the non dummy customers, which are tolerantly bounded by $3nl$. Here e denotes the expected number of restockings while serving the merged block of dummy nodes.

For the difference of the solutions costs we now have

$$f_{vrpsd}(\tau) - f_{vrpsd}(\tau') \geq 2L + (e_1 + e_2 - e)2L - 3nl$$

Here e is bounded from above by $e_1 + e_2 + 1$. Even if in τ the vehicle is almost full with high probability, before the second block of dummy customers

is served, and even if in τ' the vehicle is almost empty with high probability, before the dummy customers of the second block (in τ) are served, there is never more than 1 additional restocking necessary in τ'. On the other hand there are cases, in which the number of restockings is the same. For example, if the total demand of the second block of dummy customers is equal to $Q-1$ mod Q and the vehicles are loaded in both cases prior to those customers with exactly $Q-2$ goods. The probability for each of those three events is bounded from below by 2^{-Q} due to lemma 10 and lemma 11. Therefore, we can bound e by

$$e \leq e_1 + e_2 + 1 - 2^{-3Q}$$

For the difference of the solution costs follows

$$\begin{aligned} f_{vrpsd}(\tau) - f_{vrpsd}(\tau') &\geq 2L + (e_1 + e_2 - e)2L - 3nl \\ &\geq 2L + \left(e_1 + e_2 - \left(e_1 + e_2 + 1 - 2^{-3Q}\right)\right) 2L - 3nl \\ &\geq 2^{-3Q}2L - 3nl \\ &\geq 2^{-3Q+1}2^{3Q+1}nl - 3nl \\ &\geq 4nl - 3nl \\ &> 0 \end{aligned}$$

That means we have $f_{vrpsd}(\tau') < f_{vrpsd}(\tau)$. Additionally, by construction the number of non successive blocks of dummy customers in τ' is decreased by at least 1, which concludes the proof. □

Proposition 5. *Given a tour τ, with exactly one block of dummy customers, which is not located at the beginning of the tour, we can create a tour τ' with the following properties:*

(i) *τ' contains exactly one block of dummy customers, which is located at the beginning of the tour.*

(ii) *$f_{vrpsd}(\tau') < f_{vrpsd}(\tau)$.*

Proof. This proof is similar to that of proposition 4. Here we compare the solution τ with the solution τ', which is derived from τ by putting the only block of dummy customers to the very beginning of the tour. We know that the vehicle is fully loaded with probability 1 at the beginning, whereas this is not the case, after at least one customer has been served.

We can bound the costs for τ by only considering the travel and restocking costs caused by the dummy nodes.

$$\begin{aligned} f_{vrpsd}(\tau) &\geq L + e2L + L \\ &= 2L + e2L \end{aligned}$$

For the costs of τ' we consider the travel and restocking costs caused by the dummy nodes and we additionally bound the travel and restocking costs for the other customers tolerantly by $3nl$.

$$\begin{aligned} f_{vrpsd}(\tau') &\leq L + e'2L + L + 3nl \\ &= 2L + e'2L + 3nl \end{aligned}$$

The difference between $f_{vrpsd}(\tau)$ and $f_{vrpsd}(\tau')$ can now be bounded by

$$\begin{aligned} f_{vrpsd}(\tau) - f_{vrpsd}(\tau') &\geq 2L + e2L - (2L + e'2L + 3nl) \\ &= (e - e')2L - 3nl \end{aligned}$$

While the vehicle is always fully loaded at the beginning, e' is trivially bounded from above by e. Furthermore, there are cases, in which τ' requires one restocking less than τ. For example, if the total demand of the dummy customers is equal to $Q-1 \mod Q$ and the vehicle is loaded with $Q-2$ goods before the dummy customers are visited in τ. The probability for both events is at least 2^{-Q} due to lemma 10 and lemma 11, and therefore we can bound e' from above by $e - 2^{-2Q}$. For the difference between the solution costs we have now

$$\begin{aligned} f_{vrpsd}(\tau) - f_{vrpsd}(\tau') &\geq (e - e')2L - 3nl \\ &\geq 2^{-2Q}2L - 3nl \\ &\geq 2^{-2Q+1}2^{3Q+1}nl - 3nl \\ &= 2^{Q}4nl - 3nl \\ &> 0 \end{aligned}$$

That means we have $f_{vrpsd}(\tau') < f_{vrpsd}(\tau)$ as desired. Additionally, by construction the block of dummy customers in τ' is located at the beginning, which concludes the proof. □

Putting these two propositions together, we obtain the following result.

Theorem 8. *In an optimal tour for the VRPSD, the dummy customers have to be visited at the beginning of the tour.*

Proof. Due to proposition 4 all tours which contain at least two blocks of dummy customers are suboptimal. Therefore, an optimal tour contains only one block of dummy customers. Due to proposition 5 all tours, which contain one block of dummy customers, not located at the beginning of the tour, are suboptimal. Therefore, in an optimal tour the dummy customers have to be visited at the very beginning of the tour. □

In an optimal tour all the dummy customers are visited first. After visiting these customers the distribution of goods in the vehicle is very close to the uniform distribution. Due to the results in chapter 2 we have the following bounds for the probabilities involved in those distributions.

Lemma 12. *After the dummy customers have been visited, the probabilities for the different amounts of goods in the vehicle are bounded from below by* $1/Q - (Q-1)c^{Qp(n)}$ *and from above by* $1/Q+(Q-1)c^{Qp(n)}$*, where* $p(n)$ *is the number of dummy customers and* c *is a constant smaller than* 1*, depending only on* Q*.*

Proof. The cumulative demand distribution of all the dummy customers is a $B(Qp(n), 1/2)$ binomial distribution. That means it is the sum of $Qp(n)$ Bernoulli trials with success probability of $1/2$. Due to the results in chapter 2 there exists a constant $c \in \mathbb{R}, c < 1$, depending only on Q, such that the following bound for the probabilities p_i, that the amount of goods in the vehicle is exactly i, $i \in \{1, 2, \ldots, Q\}$, after visiting the dummy customers, including a possible restocking action at the end, holds.

$$\forall i \in \{1, 2, \ldots, Q\} : |1/Q - p_i| \leq (Q-1)c^{Qp(n)}$$

This gives the desired upper and lower bounds for the probabilities. □

Using this lemma, we can bound the costs of solutions for the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS, where the dummy customers are visited first. Since the order of the dummy nodes does not matter, we can now represent the solution just by a tour τ, which starts at the depot τ_1 and visits

each of the non dummy customers exactly once. Let τ be such a solution for the VRPSD. Then we can bound the solution costs from below in the following way. Here c_{dummy} are the costs caused by the dummy customers, without the travel costs of $d(\tau_1, \tau_2)$, which occur either implicitly in the travel costs from the last dummy customer to the first non dummy customer or explicitly, if the vehicle is empty after the last dummy customer has been served. The first double sum represents the restocking costs at the customers, the second double sum represents the restocking costs between each pair of two consecutive customers (if the vehicle gets exactly empty after serving a customer) and the third double sum represents the travel costs between each pair of two consecutive customers (if the vehicle does not get exactly empty after serving a customer). Furthermore, we set $\Delta_p = (Q-1)c^{Qp(n)}$ with reference to lemma 12 and use $\tau_{n+1} = \tau_1$ for ease of notation.

$$
\begin{aligned}
& f_{vrpsd}(\tau) \\
= \quad & c_{\text{dummy}} + d(\tau_1, \tau_2) + d(\tau_n, \tau_1) \\
& + \sum_{i=2}^{n} \sum_{j=1}^{Q} \mathbb{P}(\text{load at cust. } i \text{ is } j)\, \mathbb{P}(\text{demand} > j)\, (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\
& + \sum_{i=2}^{n-1} \sum_{j=1}^{Q} \mathbb{P}(\text{load at cust. } i \text{ is } j)\, \mathbb{P}(\text{demand} = j)\, (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\
& + \sum_{i=2}^{n-1} \sum_{j=1}^{Q} \mathbb{P}(\text{load at cust. } i \text{ is } j)\, \mathbb{P}(\text{demand} \neq j)\, d(\tau_i, \tau_{i+1}) \\
\geq \quad & c_{\text{dummy}} + d(\tau_1, \tau_2) + d(\tau_n, \tau_1) \\
& + \sum_{i=2}^{n} \sum_{j=1}^{Q} (1/Q - \Delta_p)\, \mathbb{P}(\text{demand} > j)\, (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\
& + \sum_{i=2}^{n-1} \sum_{j=1}^{Q} (1/Q - \Delta_p)\, \mathbb{P}(\text{demand} = j)\, (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\
& + \sum_{i=2}^{n-1} \sum_{j=1}^{Q} (1/Q - \Delta_p)\, \mathbb{P}(\text{demand} \neq j)\, d(\tau_i, \tau_{i+1})
\end{aligned}
$$

$$
\begin{aligned}
= \quad & c_{\text{dummy}} + d(\tau_1, \tau_2) + d(\tau_n, \tau_1) \\
& + (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\
& + (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} = j) \sum_{i=2}^{n-1} (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\
& + (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} \neq j) \sum_{i=2}^{n-1} d(\tau_i, \tau_{i+1}) \\
= \quad & c_{\text{dummy}} + d(\tau_1, \tau_2) + d(\tau_n, \tau_1) \\
& + (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\
& + (1/Q - \Delta_p) \left(1 - 2^{-Q}\right) \sum_{i=2}^{n-1} (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\
& + (1/Q - \Delta_p) \left(Q - 1 + 2^{-Q}\right) \sum_{i=2}^{n-1} d(\tau_i, \tau_{i+1}) \\
= \quad & c_{\text{dummy}} + \Delta_p Q d(\tau_1, \tau_2) + \Delta_p Q d(\tau_n, \tau_1) \\
& + (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\
& + (1/Q - \Delta_p) \left(1 - 2^{-Q}\right) \sum_{i=1}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\
& + (1/Q - \Delta_p) \left(Q - 1 + 2^{-Q}\right) \sum_{i=1}^{n} d(\tau_i, \tau_{i+1}) \\
= \quad & c_{\text{dummy}} + \Delta_p Q d(\tau_1, \tau_2) + \Delta_p Q d(\tau_n, \tau_1) \\
& + (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\
& + (1/Q - \Delta_p) \left(1 - 2^{-Q}\right) \sum_{i=1}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\
& + (1/Q - \Delta_p) \left(Q - 1 + 2^{-Q}\right) f_{tsp}(\tau)
\end{aligned}
$$

Analog we get the following upper bound, which holds for any solution τ.

$$\begin{aligned} f_{vrpsd}(\tau) \quad \leq \quad & c_{\text{dummy}} - \Delta_p Q d(\tau_1, \tau_2) - \Delta_p Q d(\tau_n, \tau_1) \\ + \quad & (1/Q + \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_i)) \\ + \quad & (1/Q + \Delta_p) \left(1 - 2^{-Q}\right) \sum_{i=1}^{n} (d(\tau_i, \tau_1) + d(\tau_1, \tau_{i+1})) \\ + \quad & (1/Q + \Delta_p) \left(Q - 1 + 2^{-Q}\right) f_{tsp}(\tau) \end{aligned}$$

Using these bounds with an optimal solution τ^{vrpsd} for the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS and an optimal solution τ^{tsp} for the TRAVELING SALESMAN PROBLEM, we get the following result.

Theorem 9. *Let τ^{vrpsd} be an optimal solution for the* VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS *and let τ^{tsp} be an optimal solution for the* TRAVELING SALESMAN PROBLEM. *Then we have the following bound for the solution costs of τ^{vrpsd} for the* TRAVELING SALESMAN PROBLEM.

$$f_{tsp}\left(\tau^{vrpsd}\right) \leq \frac{2\Delta_p (Q+1) n + \left(1/Q + \Delta_p\right) \left(Q - 1 + 2^{-Q}\right)}{\left(1/Q - \Delta_p\right) \left(Q - 1 + 2^{-Q}\right)} f_{tsp}\left(\tau^{tsp}\right).$$

Proof. Let $\Delta = f_{vrpsd}(\tau^{tsp}) - f_{vrpsd}\left(\tau^{vrpsd}\right)$. Since τ^{vrpsd} is an optimal solution for the VRPSD, we have $\Delta \geq 0$. Furthermore, we can bound Δ from above in the following way.

$$
\begin{aligned}
\Delta \leq\ & c_{\text{dummy}} - \Delta_p Q d\left(\tau_1^{tsp}, \tau_2^{tsp}\right) - \Delta_p Q d\left(\tau_n^{tsp}, \tau_1^{tsp}\right) \\
+\ & (1/Q + \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} \left(d\left(\tau_i^{tsp}, \tau_1^{tsp}\right) + d\left(\tau_1^{tsp}, \tau_i^{tsp}\right)\right) \\
+\ & (1/Q + \Delta_p)\left(1 - 2^{-Q}\right) \sum_{i=1}^{n} \left(d\left(\tau_i^{tsp}, \tau_1^{tsp}\right) + d\left(\tau_1^{tsp}, \tau_{i+1}^{tsp}\right)\right) \\
+\ & (1/Q + \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{tsp}\right) \\
-\ & c_{\text{dummy}} - \Delta_p Q d\left(\tau_1^{vrpsd}, \tau_2^{vrpsd}\right) - \Delta_p Q d\left(\tau_n^{vrpsd}, \tau_1^{vrpsd}\right) \\
-\ & (1/Q - \Delta_p) \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} \left(d\left(\tau_i^{vrpsd}, \tau_1^{vrpsd}\right) + d\left(\tau_1^{vrpsd}, \tau_i^{vrpsd}\right)\right) \\
-\ & (1/Q - \Delta_p)\left(1 - 2^{-Q}\right) \sum_{i=1}^{n} \left(d\left(\tau_i^{vrpsd}, \tau_1^{vrpsd}\right) + d\left(\tau_1^{vrpsd}, \tau_{i+1}^{vrpsd}\right)\right) \\
-\ & (1/Q - \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{vrpsd}\right) \\
\leq\ & 2\Delta_p \sum_{j=1}^{Q} \mathbb{P}(\text{demand} > j) \sum_{i=2}^{n} \left(d\left(\tau_i^{tsp}, \tau_1^{tsp}\right) + d\left(\tau_1^{tsp}, \tau_i^{tsp}\right)\right) \\
+\ & 2\Delta_p\left(1 - 2^{-Q}\right) \sum_{i=1}^{n} \left(d\left(\tau_i^{tsp}, \tau_1^{tsp}\right) + d\left(\tau_1^{tsp}, \tau_{i+1}^{tsp}\right)\right) \\
+\ & (1/Q + \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{tsp}\right) \\
-\ & (1/Q - \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{vrpsd}\right) \\
\leq\ & 2\Delta_p Q n f_{tsp}\left(\tau^{tsp}\right) + 2\Delta_p n f_{tsp}\left(\tau^{tsp}\right) \\
+\ & (1/Q + \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{tsp}\right) \\
-\ & (1/Q - \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{vrpsd}\right) \\
\leq\ & \left(2\Delta_p(Q+1)n + (1/Q + \Delta_p)\left(Q - 1 + 2^{-Q}\right)\right) f_{tsp}\left(\tau^{tsp}\right) \\
-\ & (1/Q - \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{vrpsd}\right)
\end{aligned}
$$

Putting both inequalities together, we have

$$
\begin{aligned}
0 \leq \Delta \leq\ & \left(2\Delta_p(Q+1)n + (1/Q + \Delta_p)\left(Q - 1 + 2^{-Q}\right)\right) f_{tsp}\left(\tau^{tsp}\right) \\
& - (1/Q - \Delta_p)\left(Q - 1 + 2^{-Q}\right) f_{tsp}\left(\tau^{vrpsd}\right),
\end{aligned}
$$

and therefore

$$f_{tsp}\left(\tau^{vrpsd}\right) \leq \frac{2\Delta_p(Q+1)n + (1/Q+\Delta_p)\left(Q-1+2^{-Q}\right)}{(1/Q-\Delta_p)\left(Q-1+2^{-Q}\right)} f_{tsp}\left(\tau^{tsp}\right),$$

as desired. □

Since we can use any constant integral number for Q and any polynomial in n for $p(n)$, the factor in the formula of theorem 9 can be adjusted arbitrarily close to 1. That means, an optimal solution for the VRPSD gives us a $1+\varepsilon$ approximation for the TSP, for any $\varepsilon > 0$. Since computing the latter approximation is NP-hard, we get the following result.

Theorem 10. *It is NP-hard to compute an optimal solution for* ***substantially*** *stochastic instances of the* Metric Vehicle Routing Problem with Stochastic Demands.

This is the first hardness result for *substantially* stochastic instances of the Vehicle Routing Problem with Stochastic Demands and gives a strong motivation for using heuristics to optimize stochastic instances of this problem.

3.3 The Vehicle Routing Problem with Stochastic Demands and Customers

In this section we give a reduction from the Metric Traveling Salesman Problem to *substantially* stochastic instances of the Metric Vehicle Routing Problem with Stochastic Demands and Customers. The reduction used here is a combination of the reductions from the previous two sections. In this section we only provide sketches of the proofs and point out differences to the previous sections.

Given an instance for the Metric TSP with n vertices, we construct an instance for the Metric VRPSDC in the following way. Like in section 3.2 all the vertices are used as customers in the VRPSDC instance, while one arbitrary vertex is used as the depot . Additionally, $p(n)$ dummy customers are used, where $p(n)$ is some polynomial in n. Analog to section 3.1 we use $q(n)$ copies for each of the non dummy customers, where $q(n)$ is also a polynomial in n. The probabilities for requiring a visit are set to $1/2$ for all customers and the demand distributions are set to $B(Q, 1/2)$ binomial distributions for all customers. The distances between non dummy customers and the depot remain unchanged compared to the TSP instance. Distances between dummy customers are set to 0 and the

distance between dummy customers and the depot are set to $L := 2^{3Q+4}nq(n)l$, where l is the longest distance in the original TSP instance. The distances between dummy customers and non dummy customers are set analog to section 3.2 to the sum of the distances between the dummy customers and the depot and between the depot and the corresponding non dummy customer.

Now we can proof results analog to lemma 10 and lemma 11 in section 3.2. Since the customers are only visited with a certain probability, we have to say that a customer has been processed (instead of visited). Otherwise, both lemmata remain valid, using a bound for the probabilities of 2^{-Q-1} instead of 2^{-Q}. Note that the constant L has been adjusted, such that the other proofs remain valid with the new bound of the probabilities.

In the next step we show that in an optimal solution the dummy customers have to be processed at the very beginning of the tour. We have to change the proofs slightly in this case, because a block of dummy customers can be skipped completely for some realizations of the random events. Here we show in a first step with a proof analog to that of proposition 5, that the first block of dummy customers has to be visited at the very beginning of the tour. In a second step we show that if there are multiple blocks of dummy customers (where the first block is located at the very beginning of the tour), we can improve the solution quality by merging the second block at the location of the first one. In those cases, where the second block is skipped in the a posteriori tour, the solution costs are not affected. In the cases, where the first block is skipped, the solution costs decrease, which can be shown analog to proposition 5. In those cases, where both blocks are not skipped, the solution costs decrease, which can be shown analog to proposition 4. All in all, we can retrieve a result analog to theorem 8.

It remains to show that we can transform any optimal tour containing more than $n-1$ groups of the non dummy customers to an optimal tour with $n-1$ groups in polynomial time. This can be shown completely analog to proposition 2.

Finally, we have to show that an optimal solution with $n-1$ groups of non dummy customers provides us with a good approximation for the original TSP instance. Analog to section 3.1 we derive a TSP solution from the ordering of the depot and the $n-1$ groups of the optimal VRPSDC solution. The bounds used in section 3.2 for the solution costs also change. For the lower bound we only focus on the case where at least one customer in each of the groups has to be visited, which introduces an additional factor of $1-n2^{-q(n)}$. For the upper bound we use this same factor and additionally bound the other case, which occurs with a probability of at most $n2^{-q(n)}$, appropriately. Note that although customers are skipped, the costs for the a posteriori solution could in principle increase due to

different restocking costs. But since this case occurs with a probability arbitrarily close to 0, we can bound the costs tolerantly by a multiple of the costs for the other case. In total the approximation factor of theorem 9 changes slightly, but it is still possible to adjust it arbitrarily close to 1.

All in all, we have shown that we can transform any optimal solution for the VRPSDC instance to an optimal solution for the VRPSDC instance with exactly $n-1$ groups of the non dummy customers. In such a solution the dummy customers have to be visited at the very beginning of the tour. Using adequate polynomials $p(n)$ and $q(n)$ as well as an adequate vehicle capacity Q, we can derive any $1+\varepsilon$ approximation for the original TSP instance from the optimal VRPSDC solution in polynomial time. Since it is NP-hard to compute such an approximation, we have shown the following result.

Theorem 11. *It is NP-hard to compute an optimal solution for* ***substantially*** *stochastic instances of the* Metric Vehicle Routing Problem with Stochastic Demands and Customers.

Like for the problems in the previous sections it is the first time that hardness results for *substantially* stochastic instances of the Vehicle Routing Problem with Stochastic Demands and Customers problem are shown. These results are of great importance to justify the usage of heuristics for the optimization of those problems.

3.4 Discussion and Conclusions

In this chapter we have shown that it is NP-hard to compute optimal solutions and certain approximations for (*substantially*) stochastic instances of some important stochastic vehicle routing problems. In fact, stochastic instances are of high interest for practical applications and they are the reason why stochastic vehicle routing problems problems are not treated as non-stochastic problems.

With our results we could gain more insight into the structure of these problems. Furthermore, our results justify the common usage of heuristics for large *substantially* stochastic instances of stochastic vehicle routing problems. We hope that the results can be extended and generalized, for example to other stochastic combinatorial optimization problems, or to stronger results for the problems investigated in this chapter. Finally, we hope that the insights gained about the structure of the problems considered in this chapter can also be used for the development of better heuristics for these problems.

We finish this chapter with a discussion of the results obtained within the sections 3.1, 3.2 and 3.3. In particular, we focus on possible generalizations of our results.

3.4.1 Stochastic Instances of the PTSP

We have shown that it is NP-hard to compute a 2^n approximation for stochastic instances of the PTSP. Although these instances are stochastic, the probabilities that are used are very close to 1. In this way the PTSP instances are very similar to TSP instances and therefore it is not surprising that inapproximability results can be transferred from the TSP to the PTSP. The only important condition is that the distances and probabilities used in the PTSP instance can be encoded efficiently with regard to the size of the HC instance. This can be guaranteed for approximation factors, which can be encoded efficiently regarding the size of the HC instance. Thus our result can be strengthened easily to such approximations factors. The results could also be generalized to instances with heterogeneous probabilities, as long as the probability for the event that all customers require a visit in one particular realization is sufficiently close to 1.

3.4.2 Stochastic Instances of the Metric PTSP

We have proved that it is NP-hard to compute a $\frac{117}{116}-\varepsilon$ approximation for stochastic instances of the METRIC PTSP with asymmetric distances and a $\frac{220}{219}-\varepsilon$ approximation for stochastic instances of the METRIC PTSP with symmetric distances. The key fact is that the probability for the event that all customers require a visit simultaneously is asymptotically converging to 1. Thus the results could be generalized to instances with heterogeneous probabilities or instances with probabilities not that close to 1, especially considering stronger bounds instead of the union bound. Another possible generalization would be to use instances with probabilities, such that the probability for the event, that all customers require a visit simultaneously, is asymptotically a constant close to 1. In that case the assumptions on the probabilities are not that strong, but therefore the inapproximability result gets weaker. In fact, we have shown later in section 3.1 that we can use weaker assumptions on the probabilities, still obtaining the same inapproximability results.

A very important generalization is with respect to hardness results for special classes of metric instances. One such class, which is especially important in practical applications, consists of Euclidean instances, where the travel times are taken as distances from an Euclidean space. For example, in Trevisan [2000] it

has been shown, that the TSP on Euclidean instances in $\log n$ dimensions with any l_p norm, is Max SNP-hard. This result can be easily used to show that the PTSP on stochastic Euclidean instances in $\log n$ dimensions with any l_p norm is also Max SNP-hard.

3.4.3 Substantially Stochastic Instances of the Metric PTSP

Here we have shown that it is NP-hard to compute a $\frac{117}{116} - \varepsilon$ approximation for *substantially* stochastic instances of the METRIC PTSP with asymmetric distances and a $\frac{220}{219} - \varepsilon$ approximation for *substantially* stochastic instances of the METRIC PTSP with symmetric distances. The important difference to the previous results is that the instances are using constant probabilities and are therefore denoted as *substantially* stochastic. In Bianchi et al. [2003] it has been shown that the PTSP and the TSP differ in general quite a lot on such instances and therefore these instances are of particular interest.

In our analysis we used the fact that the probability of the event that at least one customer in each group requires a visit is asymptotically converging to 1. That means the results can be easily generalized to instances with other, possibly heterogeneous probabilities and to other, possibly heterogeneous numbers for the copies of customers. Obviously larger probabilities can be used without affecting the results, but even polynomially small probabilities could be used leading to the same results.

As for the previous reduction, one very important generalization is with respect to hardness results for special classes of metric instances. In this way we can show that the PTSP on *substantially* stochastic Euclidean instances in $\log n$ dimensions with any l_p norm is also Max SNP-hard.

3.4.4 The Vehicle Routing Problem with Stochastic Demands

For the METRIC VRPSD we could show that it is NP-hard to compute an optimal solution for *substantially* stochastic instances. The results can be generalized to other, possibly heterogeneous demand distributions, as long as it is possible to show that in an optimal solution all the dummy customers are visited at the beginning and as long as the distribution of goods in the vehicle is close to the uniform distribution after the dummy customers are processed. In chapter 2 some mild conditions for the latter requirement are given.

As for the PTSP reductions, the results can also be generalized with respect to hardness results for special classes of metric instances. For example, it can be shown that it is NP-hard to compute optimal solutions for Euclidean instances in

$\log n$ dimensions with any l_p norm. Otherwise, we would have a PTAS for the TSP on Euclidean instances in $\log n$ dimensions with any l_p norm and this would be a contradiction to the results in Trevisan [2000].

3.4.5 The Vehicle Routing Problem with Stochastic Demands and Customers

For the METRIC VRPSDC we could prove that it is NP-hard to compute an optimal solution for *substantially* stochastic instances. Since the reduction was a combination of the reductions for the PTSP and the VRPSD, we refer for possible generalizations to the generalizations for these reductions. We only want to mention that also for this problem generalizations with respect to hardness results for special classes of metric instances are possible. For example, it is NP-hard to compute optimal solutions for Euclidean instances in $\log n$ dimensions with any l_p norm due to the same argument which has been used for the VRPSD.

Chapter 4

Hardness Results for the Probabilistic Traveling Salesman Problem with Deadlines

In this chapter we will discuss some hardness results regarding the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. The corresponding publications are Weyland et al. [2012a,d]. Although the PTSPD seems to be similar to the PTSP, and in fact the problem definitions are quite similar as we have seen, the introduction of deadlines significantly changes the structure of the problem. Most of the solution approaches for the PTSP cannot be used for the PTSPD in a straightforward manner. In particular, the polynomial time approach for the evaluation of the PTSP objective function cannot be adapted for the PTSPD. Additionally, applying the available analytical approximations for the PTSP objective function does not result in a satisfactory approximation behavior, since it is not clear at all how penalties for missed deadlines can be approximated properly. We will see in this chapter that there are also significant differences regarding the complexity of different computational tasks for the PTSPD and the PTSP.

The remaining part of this chapter is organized in the following way. We start with results about the complexity of several computational tasks regarding the PTSPD. After that we focus on the PTSPD objective function. Here we discuss the worst case performance of the existing approximations for the objective function. Finally, we show an inapproximability result for the objective function of a slightly more general problem. We finish with a discussion of our results.

4.1 Hardness Results for the PTSPD

In this section we prove results about the complexity of several computational tasks related to the PTSPD. In fact, we will see that there is an interesting connection between the PTSPD and the class of counting problems, #P (Arora and Barak [2009]). The task for these counting problems is to compute the number of feasible solutions for problems in NP. There are also other equivalent definitions, but we do not want to go too much into detail here. For our work it is of great importance that the whole polynomial hierarchy (including the complexity class NP) can be solved using a polynomial time computation with one call to a problem in #P (Arora and Barak [2009]). Showing that a problem is #P-hard is therefore a stronger result than showing that it is "only" NP-hard. In Dyer and Stougie [2006] it has been already shown that two-stage stochastic programming is #P-hard in general. In fact, it has been shown that even the computation of the recourse costs for such problems is #P-hard in general. These results were shown by a polynomial reduction from a #P-complete problem to an artificial two-stage stochastic programming problem. In this section we strengthen these results and show for the first time that a particular a priori optimization problem with practical relevance is #P-hard. More in detail, we will show that it is #P-hard to compute the probabilities of deadline violations for Euclidean instances of the PTSPD in general. Based on this result we continue to show that the evaluation of the objective function is #P-hard for Euclidean instances of the PTSPD. The same holds for the closely related task of computing the difference between the costs of two solutions for any *reasonable local search neighborhood*. Although even the evaluation of the objective function is a computationally demanding task, this does not immediately imply any results about the complexity of the optimization and decision variants of the PTSPD. Nonetheless, we will show that also these two tasks are #P-hard for Euclidean instances.

In this chapter we focus on the PTSPD RECOURSE I with fixed penalties (Campbell and Thomas [2008b]). This is the variant of the PTSPD which has been formally introduced in chapter 1. Nonetheless, the same results extend to all the four variants of the PTSPD. Before we finally start with the proofs, we want to formally introduce the counting version of the well-known KNAPSACK PROBLEM (Salkin and De Kluyver [1975]), called #KNAPSACK, since the proofs are all based on a reduction from this problem. Note that #KNAPSACK is #P-complete (Morris and Sinclair [2004]).

Problem 7 (#KNAPSACK)**.** *Given a row vector $w \in \mathbb{R}^n$ and a bound $W \in \mathbb{R}$, the problem is to compute the cardinality of the set $S = \{x \in \mathbb{B}^n \mid wx \leq W\}$.*

In words, the task is to compute the number of feasible solutions for the KNAPSACK PROBLEM. These are solutions which respect the upper bound for the total weight of selected items. Note that for this problem only the weights and not the values of the items are of interest. Therefore, the values of the items are not included in the problem definition.

4.1.1 Computing the Probabilities of Deadline Violations

We begin with showing that computing the probabilities of deadline violations for Euclidean instances of the PTSPD is #P-hard. For this purpose we use a polynomial time reduction from #KNAPSACK. We show that we are able to solve #KNAPSACK if we are able to compute the probabilities with which deadlines for Euclidean instances of the PTSPD are violated.

Theorem 12. *Computing the probabilities of deadlines violations for Euclidean instances of the PTSPD is #P-hard.*

Proof. Given an instance for the #KNAPSACK PROBLEM according to definition 7, we create an instance for the Euclidean PTSPD as depicted in figure 4.1. We put the depot and n vertices equally spaced on a line. The distances between two consecutive vertices is some constant q and the probabilities for all these vertices are set to 1. The depot, v_0, is located at the very left, the other vertices are denoted from left to right with $v_1, v_2, \ldots, v_n$. Between each pair of vertices v_{i-1} and v_i, $i \in \{1, 2, \ldots, n\}$, we put another vertex u_i in a way, such that the distance to both vertices v_{i-1} and v_i is $q/2 + w_i/2$. The probabilities for these new vertices are set to $1/2$. Finally, vertex v_n gets assigned a deadline of $nq+W$, for all the other vertices we do not impose a deadline (i.e. we set the deadline to some sufficiently high value).

We have constructed the instance in a way, such that there exists a bijection between the different realizations of the random events and the different binary vectors x of the original #KNAPSACK PROBLEM. Here we denote by the term realization of the random events a specific scenario in which certain vertices are present while the other vertices are not present. For a realization r, where the vertices $u_i, i \in T$, are present, the corresponding vector $x(r)$ is defined according to $x_i = 1$ if and only if $i \in T$. Now let our a priori solution visit the vertices from left to right, i.e. in the order $v_0, u_1, v_1, u_2, v_2, \ldots, u_n, v_n$. The arrival time at vertex v_n for a certain realization r is then $nq + wx(r)$. The inequality $nq + wx(r) \leq nq + W$ for the deadline of vertex v_n is equal to the inequality $wx(r) \leq W$ for the original problem. That means a realization for which the deadline at vertex v_n is met corresponds to a feasible solution of the original #KNAPSACK PROBLEM. On

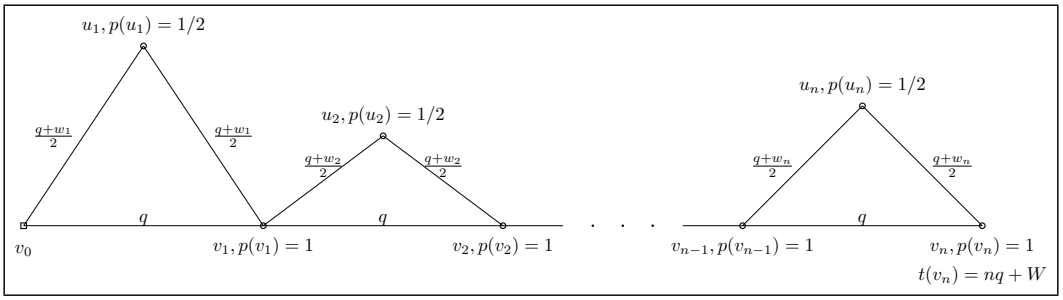


Figure 4.1. The instance of the Euclidean PTSPD used in theorems 12, 13 and 14. Penalty values are not included in the visualization. Missing deadlines correspond to sufficiently high values, such that they do not impose any constraints.

the other hand, realizations which violate the deadline at vertex v_n correspond to infeasible solutions of the original problem. Since all realizations occur with the same probability of 2^{-n}, the probability with which the deadline of vertex v_n is met, multiplied with 2^n, yields a solution for the original problem. In other words, we are able to solve the original problem if we can compute the probability with which the deadline at vertex v_n is met (or violated).

Since the computational time for this reduction is polynomially bounded in the input size of the original #KNAPSACK PROBLEM, we can conclude the proof. □

4.1.2 Evaluating the Objective Function

In the previous section we have seen that it is #P-hard to compute the probabilities for deadline violations of Euclidean PTSPD instances. The reduction used for this proof contained only one vertex for which a deadline was imposed. We have seen already that the objective function consists of the expected travel time plus the expected penalties. Since we are able to efficiently compute the expected travel time, we can derive the probabilities with which a deadline for the instances of the previous section is violated from the costs of such a solution. In the following we formalize this idea.

Theorem 13. *Evaluating the objective function for Euclidean instances of the PTSPD is #P-hard.*

Proof. We use the same reduction as in the previous section and we additionally show that we can efficiently compute the probability for a deadline violation at

vertex v_n using the costs of the given a priori solution. We have the following formula for the costs of the given solution τ.

$$\begin{aligned} f_{\text{ptspd}}(\tau) &= \mathbb{E}(\text{travel time for } \tau) + \mathbb{E}(\text{penalties for } \tau) \\ &= \mathbb{E}(\text{travel time for } \tau) + \mathbb{P}(\text{deadline violation at vertex } v_n) \cdot h(v_n) \end{aligned}$$

Here $h(v_n)$ is the fixed penalty value for a deadline violation at vertex v_n according to the definition of the PTSPD. That means we can express the probability for a deadline violation at vertex v_n by

$$\mathbb{P}(\text{deadline violation at vertex } v_n) = \left[f_{\text{ptspd}}(\tau) - \mathbb{E}(\text{travel time for } \tau) \right] / h(v_n).$$

Given the costs of the solution, we are now able to efficiently compute the probability with which the deadline at vertex v_n is violated, since we are able to efficiently compute the expected travel time. This concludes the proof. □

4.1.3 Delta Evaluation in Local Search

Before we give hardness results for the decision variant and the optimization variant of the PTSPD, we want to discuss hardness results regarding a very important speed-up technique for local search algorithms, namely *delta evaluation*. *Delta evaluation* is used for computing the difference of the costs between two neighbor solutions in a given local search neighborhood. We discuss these results here, since they are strongly related to our previous results. Local search algorithms play an important role for different stochastic vehicle routing problems as we will see later and the technique of *delta evaluation* has been successfully applied in many cases, usually leading to major runtime improvements. In fact, such heuristics are currently the state-of-the-art methods for the PTSP (Birattari et al. [2008a]; Weyland et al. [2009a]). Here we show that *delta evaluation* is #P-hard in so-called *reasonable local search neighborhoods* for Euclidean instances of the PTSPD. The overall idea is to use the same instances and solutions as in the previous sections. We then show that starting from a solution, whose costs can be computed efficiently, we arrive with a polynomially bounded number of local search steps at the solution used in the previous reductions. That means we can compute the costs of this solution with polynomially many *delta evaluations*. Before we give the formal proof, we define what we understand of a *reasonable local search neighborhood*.

Definition 6. *We call a local search neighborhood **reasonable** if every solution can be reached from any starting solution within polynomially many local search steps, and if these local search steps can be determined efficiently.*

Note that this definition does not impose strong constraints on the local search neighborhood. In fact, most of the local search neighborhoods used for routing problems are *reasonable local search neighborhoods* according to our definition (for example the local search neighborhoods described in Johnson and McGeoch [1997]).

Theorem 14. ***Delta evaluation** in **reasonable local search neighborhoods** for Euclidean instances of the PTSPD is #P-hard.*

Proof. We start again with the Euclidean PTSPD instances used in the proof of theorems 12 and 13. This time we focus on the solution starting at the depot, visiting customer v_n, followed by all the customers from left to right and finishing at the depot. The only deadline in this instance is imposed for customer v_n. For this solution the deadline is met, since the distance between v_0 and v_n is nq and the deadline of v_n is $t(v_n) = nq + W$. Therefore the total costs of this solution consist only of the expected travel times. And these expected travel times can be computed efficiently.

Now let us call this solution τ_1. Since we are using a *reasonable local search neighborhood* there exists a sequence of at most polynomially many solutions τ_1, τ_2, ..., τ_m (that can be computed efficiently), where τ_m is the solution used in the proof of theorems 12 and 13. Starting with the solution costs for τ_1, we are able to compute the solution costs of τ_m using $m-1$ *delta evaluations* in the local search neighborhood. It is #P-hard to compute the costs of τ_m due to theorem 13, which concludes the proof. □

4.1.4 The Decision Variant of the PTSPD

The decision variant of the PTSPD is the problem of deciding whether there exists a solution with costs at most k or not. Here we show that the decision variant for Euclidean instances of the PTSPD is #P-hard as well. The overall idea is to modify the initial reduction of theorem 12 such that we know the optimal solution. Finally, a binary search on the values of k for the decision variant can be used to determine the probability with which the deadline of the last customer is violated, which then allows us to solve the original instance of #KNAPSACK PROBLEM.

Theorem 15. *The decision variant of the Euclidean PTSPD is #P-hard.*

Proof. Given an instance for the #KNAPSACK PROBLEM according to definition 7, we create an instance for the Euclidean PTSPD similar to that used in the previous proofs. The PTSPD instance is visualized in figure 4.2. We use the vertices of the previous proofs and add a new vertex x at the right side with a distance of Q from v_n. The deadline for this new vertex is set to $nq+Q+W$, which corresponds to the constraint of the original #KNAPSACK PROBLEM instance. The deadlines for all other vertices are set to Q.

With a sufficiently large value for Q (e.g. $nq + \sum_{i=1}^{n} u_i$) and sufficiently large values for the penalties for the vertices $v_1, v_2, \ldots, v_n$ and $u_1, u_2, \ldots, u_n$ we can ensure that x is visited in an optimal solution at the very end. We can then further show that in an optimal solution all vertices are visited from left to right (This can also be trivially guaranteed by using more complex values for the deadlines of the vertices $v_1, v_2, \ldots, v_n$ and $u_1, u_2, \ldots, u_n$).

In this case the costs for an optimal solution are the expected travel times plus the penalties for deadline violations at vertex x. Since we are able to efficiently compute the expected travel times, we can determine the penalty costs at vertex x with a binary search on the decision variant of the Euclidean PTSPD. Let $\mathbb{E}(\text{travel time})$ denote the expected travel time and let $\mathbb{P}(\text{deadline violation at vertex } x)$ denote the probability for a deadline violation at vertex x. Then the costs for an optimal solution are

$$\mathbb{E}(\text{travel time}) + \mathbb{P}(\text{deadline violation at vertex } x) \cdot h(x).$$

With a binary search on the decision variant of the PTSPD, starting with a bound of $k = \mathbb{E}(\text{travel time}) + h(x)/2$, the costs of the optimal solution can be determined within n steps. These costs can then be used to determine the probability with which the deadline at customer x is met, which enables us to solve the original #KNAPSACK PROBLEM instance.

□

4.1.5 The Optimization Variant of the PTSPD

Although we know already that the decision variant of the PTSPD is #P-hard, we cannot immediately conclude that the optimization variant is #P-hard as well (as it is for example done for NP-hard decision/optimization problems in Arora and Barak [2009]), since we are not able to efficiently evaluate solutions. For showing that the optimization variant of Euclidean instances of the PTSPD is #P-hard, we further modify the instance used in the previous proof. The idea here is that we create an instance where, depending on the probability with which the

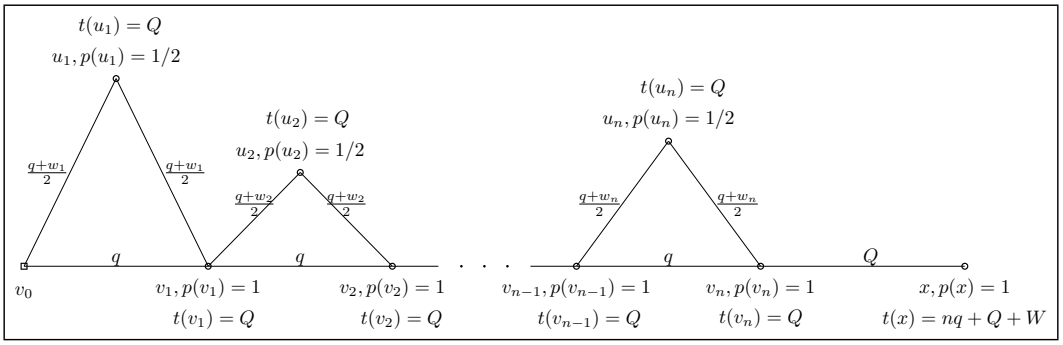


Figure 4.2. The instance of the Euclidean PTSPD used in theorem 15. Penalty values are not included in the visualization.

deadline which corresponds to the #KNAPSACK PROBLEM constraint is met, one of two solutions τ_1 and τ_2 is the optimal solution. If this probability is below a certain threshold, τ_1 is the optimal solution, if the probability is above that threshold, τ_2 is the optimal solution. With a binary search on these threshold values, we are then able to determine the probability with which the deadline is met, which finally enables us to solve the original instance of the #KNAPSACK PROBLEM.

Theorem 16. *The optimization variant of the Euclidean PTSPD is #P-hard.*

Proof. Given an instance for the #KNAPSACK PROBLEM according to definition 7, we create an instance for the Euclidean PTSPD as visualized in figure 4.3. The main difference is that instead of the single vertex x that was used in the previous proof, we have now two vertices x_1 and x_2, which are both at distance Q from the vertex v_n. The distance between x_1 and x_2 is also Q. The deadline for vertex x_1 is set to $nq + Q + W$ and corresponds to the constraint of the original #KNAPSACK PROBLEM, for x_2 no deadline is imposed. As in the previous proof, it is possible to show that the vertices are visited from left to right in an optimal solution. For the last part of the tour there are two possibilities. Either x_1 is visited prior to x_2 (we call this tour τ_1) or x_2 is visited prior to x_1 (we call this tour τ_2). Using $\Delta = d(x_2, v_0) - d(x_1, v_0)$, the difference between the costs of τ_2 and τ_1 can be expressed by

$$f_{\text{ptspd}}(\tau_2) - f_{\text{ptspd}}(\tau_1) = h(x_1) - ph(x_1) - \Delta.$$

Here p is the probability that the deadline at customer x_1 is violated for solution τ_1. In the case where τ_2 is the optimal solution we have

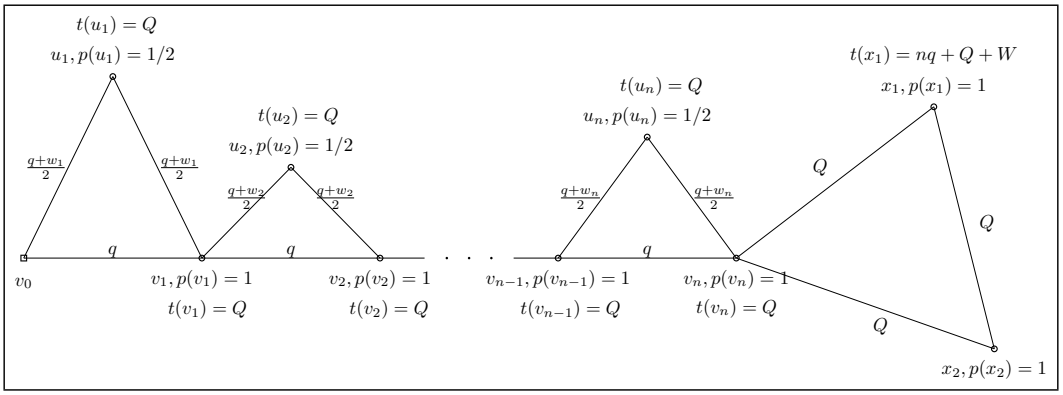


Figure 4.3. The instance of the Euclidean PTSPD used in theorem 16. Penalty values are not included in the visualization. Missing deadlines correspond to sufficiently high values, such that they do not impose any constraints.

$$\begin{aligned} & h(x_1) - ph(x_1) - \Delta \leq 0 \\ \Leftrightarrow \quad & (1-p)h(x_1) \leq \Delta \\ \Leftrightarrow \quad & 1 - p \leq \Delta / h(x_1). \end{aligned}$$

In that case we know that the probability with which the deadline is met for solution τ_1 is bounded from above by $\Delta/h(x_1)$. Analog we can show that in the case where τ_1 is the optimal solution the probability with which the deadline at customer x_1 is met for solution τ_1 is bounded from below by $\Delta/h(x_1)$. By using different values for Δ and/or $h(x_1)$ we are able to adjust this threshold value.

Now we can use a binary search on the optimization variant on instances with different threshold values, starting with $\Delta/h(x_1) = 1/2$. Within n steps we are then able to determine the probability with which the deadline at customer x_1 is met for solution τ_1, which allows us then to solve the original #KNAPSACK PROBLEM instance. □

4.2 Approximations for the PTSPD Objective Function

In the previous section we have seen that the evaluation of the PTSPD objective function is #P-hard for Euclidean instances. The approximability of the PTSPD objective function is still an open problem. For practical applications different approximations of this objective function are used (Campbell and Thomas

[2009]; Weyland et al. [2012b]). In this section we focus on these approximations and investigate the worst-case approximation guarantees that are obtained by these approaches. In Campbell and Thomas [2009] three different approximations are introduced. One approach is based on the aggregation on a temporal level and called *temporal aggregation approximation*. Using appropriate aggregation factors, this approach requires a polynomial runtime. Note that approaches using temporal aggregation have been successfully applied for different problems, including the PTSPD (Campbell and Thomas [2009]) and the PTSP (Campbell [2006]). The second approach, denoted by *expected value approximation*, computes the expected arrival times at customers which are then used for the computation of penalties. That means that instead of the probability distributions of the arrival times only the expected value of these arrival times are used. The last approach is based on the recursive computation of the objective function and truncates parts that are supposed to contribute only very small values to the overall costs. This approach is called *truncation approximation*. Unfortunately, this approach does not guarantee a polynomial runtime and therefore we do not consider it in this section. We just want to note that similar results hold also for this approach, although it is not a polynomial time algorithm. Additionally, an approach based on Monte Carlo sampling has been introduced in Weyland et al. [2012b] in the context of the PTSPD. Here Monte Carlo sampling is used to approximate the expected travel times and penalties. We denote this approach by *Monte Carlo sampling approximation*.

In the remaining part of this section we investigate the worst-case approximation guarantees of the three approximations *temporal aggregation approximation*, *expected value approximation* and *Monte Carlo sampling approximation*. We give a detailed explanation for each of these approximations and present examples to derive lower bounds for the worst-case approximation guarantees of these approaches.

4.2.1 The Expected Value Approximation

As we stated before the *expected value approximation* does not use the probability distributions of the arrival times at the customers to compute the penalties. Instead of these probability distributions only the expected arrival times (under the assumption that a visit is required) are used. This means that the formula for the expected penalties of a solution τ changes to

$$\mathbb{E}(\text{penalties for } \tau) = \sum_{i=2}^{n} p(\tau_i) h(\tau_i) V_{\tau_i},$$

where V_{τ_i} is an indicator variable with a value of 1 if the expected arrival time at customer τ_i violates the deadline of that customer and 0 otherwise. Since the expected arrival times and the expected travel times can both be computed in polynomial time, the *expected value approximation* requires only a polynomial runtime.

By using only the expected arrival times this approximation completely abstracts from the probability distributions describing the deadline violations at the different customers. In the following we use this fact to prove that the expected value approximation cannot approximate the PTSPD objective function within any reasonable factor.

Theorem 17. *The expected value approximation cannot approximate the PTSPD objective function within a factor of* 2^n.

Proof. To show that the expected value approximation cannot approximate the PTSPD objective function within a factor of 2^n in general, we create for a given instance size n the instance depicted in figure 4.4. The a priori tour τ that is used in this proof starts at the depot v_0 and visits the customers in the order $v_1, v_2, \ldots, v_n$. The vertices are located at three different locations. One for the depot v_0, one for v_1 and one for all the other vertices. The distances between these locations are all 4. All the vertices are visited with probability 1, except v_1, which has a visiting probability of $1/2$. The only deadline is imposed on v_2 with $t(v_2) = 7$.

The expected travel time for the given a priori tour is $1/2 \cdot 12 + 1/2 \cdot 8 = 10$. For the arrival time at customer v_2 we have to distinguish two cases. If v_1 is visited, then we arrive at vertex v_2 at time 8. If v_1 is not visited, we arrive at vertex v_2 at time 4. In the former case we do not meet the deadline of vertex v_2, in the latter case the deadline of vertex v_2 is met. Since both events occur with a probability of $1/2$ the costs for τ are

$$f_{\text{ptspd}}(\tau) = 10 + 1/2 \cdot h(v_2).$$

On the other hand, the expected arrival time at vertex v_2 is $1/2 \cdot 8 + 1/2 \cdot 4 = 6$, the expected value approximation computes costs of

$$f^{\text{ev}}_{\text{ptspd}}(\tau) = 10.$$

For a penalty value of $h(v_2) = 20 \cdot 2^n$ for vertex v_2 the fraction between the exact costs and the costs computed by the expected value approximation are

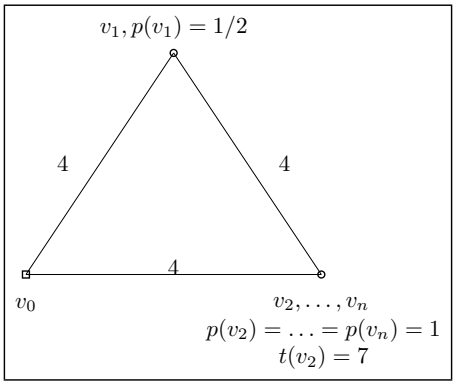


Figure 4.4. An Euclidean instance of the PTSPD that cannot be approximated within a factor of 2^n by the expected value approximation.

$$f_{\text{ptspd}}(\tau)/f^{\text{ev}}_{\text{ptspd}}(\tau) = (10 + 1/2 \cdot h(v_2))/10 = 1 + 1/20 \cdot h(v_2) \geq 2^n,$$

which concludes the proof. □

4.2.2 The Temporal Aggregation Approximation

In chapter 1 we have discussed a recursive approach for the computation of the expected penalties. It can be shown that this approach requires a computational time of $\mathcal{O}(n^2T)$, where T is an upper bound of the latest possible arrival time at a customer. The idea of the *temporal aggregation approximation* is to scale the travel times between the different locations by a common factor and to round these resulting values. In this way the upper bound of the latest possible arrival time is also decreased. For an aggregation factor of a the computational time for the *temporal aggregation approximation* is $\mathcal{O}(n^2T/a)$. Using appropriate aggregation factors, a polynomial computational time can be obtained, since the computation of the expected travel times requires only a polynomial runtime.

The approximation error for this method results from rounding errors of the scaled distances between the different locations. These rounding errors influence the approximation of the expected travel time as well as the approximation of the probabilities with which deadlines are violated at the different customers. In the latter case even a small inaccuracy can lead to a huge approximation error for the overall costs. This observation is used in the following theorem to prove that the temporal aggregation approximation is not able to approximate the PTSPD objective function within any reasonable factor. In fact, we show that

this even holds for the TRAVELING SALESMAN PROBLEM WITH DEADLINES (TSPD), the non-stochastic variant of the PTSPD.

Theorem 18. *The temporal aggregation approximation is not able to approximate the PTSPD objective function within a factor of* 2^n.

Proof. To show the desired result, we create for a given instance size n the PTSPD instance depicted in figure 4.5. The depot v_0 is located on the very left, then a single vertex v_1 is located right of the depot at a distance of 1 and all the other vertices are located at the very right with a distance of 2^n to v_1 and a distance of $2^n + 1$ to the depot v_0. The probabilities for all customers are set to 1, which means that we are even considering an instance of the TSPD, the non-stochastic variant of the PTSPD. A deadline is only imposed on customer v_2 with a value of 2^n. The a priori tour τ used for this proof starts at the depot v_0, visits v_1 followed by the other vertices $v_2, \ldots, v_n$ and finishes at the depot v_0.

Since there is no stochasticity involved in the instance, we can easily compute the costs of the a priori tour. The travel time simply sums up to $2(2^n + 1)$. The arrival time at customer v_2 is $2^n + 1$ and therefore the deadline is always missed, which leads to additional costs of $h(v_2)$. In total the costs of τ are

$$f_{\text{ptspd}}(\tau) = 2(2^n + 1) + h(v_2).$$

Let us focus on the costs of τ computed by the temporal aggregation approximation. To ensure that the temporal aggregation approximation runs in polynomial time, we have to use an aggregation factor of $\geq 2^n / g(n)$ for some polynomial $g(n)$. For sufficiently large n the aggregated travel time between v_0 and v_1 is 0. The aggregated travel time between v_0 and any of the remaining vertices is $g(n)$. The same holds for the aggregated travel time between v_1 and any of the remaining vertices and for the deadline imposed at customer v_2. Using these aggregated values, the deadline at customer v_2 is always met and therefore the costs computed by the temporal aggregation approximation are

$$f^{\text{ta}}_{\text{ptspd}}(\tau) = 2 \cdot 2^n.$$

For a penalty value of $h(v_2) = 2^{2n+1}$ for vertex v_2 the fraction between the exact costs and the costs computed by the expected value approximation are

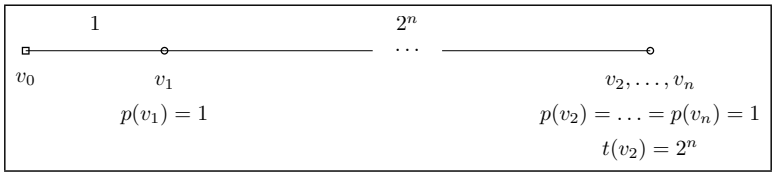


Figure 4.5. An Euclidean instance of the PTSPD that cannot be approximated within a factor of 2^n by the temporal aggregation approximation.

$$\begin{aligned} f_{\text{ptspd}}(\tau)/f^{\text{ta}}_{\text{ptspd}}(\tau) &= (2(2^n+1)+h(v_2))/2\cdot 2^n \\ &\geq h(v_2)/2^{n+1} \\ &= 2^{2n+1}/2^{n+1} \\ &= 2^n. \end{aligned}$$

This concludes the proof. □

4.2.3 The Monte Carlo Sampling Approximation

For the *Monte Carlo sampling approximation* a set of s identically and independently distributed scenarios, $\omega_1, \omega_2, \ldots, \omega_s$, are sampled according to the probability distribution given by the function p. Each of these scenarios determines for each customer if a visit is required or not (in that specific scenario). Given a specific scenario, the costs for the travel times and penalties for the a posteriori solution under that specific scenario can be computed straightforward in linear runtime. Let $c(\tau, \omega_i)$ be those costs for a given solution τ and a given sample ω_i. Then $s^{-1}\sum_{i=1}^{s} c(\tau, \omega_i)$ gives us an estimation for the exact objective function. Since each evaluation of the function c can be performed in linear time, the total computational time is $\mathcal{O}(ns)$. For a constant or a polynomially bounded number of samples s, the *Monte Carlo sampling approximation* requires a polynomial runtime.

Since the approximation quality varies with the randomly generated samples, we have to consider this for our results. Therefore, we show in this section that with high probability (regarding the generated samples) the *Monte Carlo sampling approximation* is not able to approximate the PTSPD objective function within a factor of 2^n. We formalize this in the following theorem.

Theorem 19. *The Monte Carlo Sampling approximation using $g(n)$ samples, where $g(n)$ is a polynomial in the input size, is not able to approximate the PTSPD objective function within a factor of 2^n with a probability of at least $1 - g(n)2^{-n}$.*

Proof. For this proof we create a PTSPD instance as shown in figure 4.6. All the vertices are located at only two different positions. Without loss of generality we assume that the number of vertices is even. Vertices $v_0, v_2, \ldots, v_{n-1}$ are located at the first position and vertices $v_1, v_3, \ldots, v_n$ are located at the second position. The travel time between these two positions is 2. The probabilities for all the customers are set to $1/2$ and the only deadline is imposed at customer v_n with a value of $t(v_n) = 2n - 1$. The a priori solution τ used in this proof starts at the depot v_0, visits the customers in increasing order with respect to their indices and finishes at the depot v_0.

We will now bound the costs computed by the PTSPD objective function and the costs computed by the *Monte Carlo Sampling approximation* using $g(n)$ samples selected according to the given probabilities. For the exact objective function we focus only on the costs imposed by a deadline violation at customer v_n. The deadline at this customer is only violated if customer v_n requires to be visited and all the other customers were visited before. The probability for the deadline violation is therefore 2^{-n} and the costs can be bounded by

$$f_{\text{ptspd}}(\tau) \geq 2^{-n} h(v_n).$$

Let us focus on the costs computed by exactly one sample selected randomly according to the given probabilities. With a probability of 2^{-n} all the customers require to be visited in the randomly selected sample. In this case the costs would be $2(n+1) + h(v_n)$. On the other hand, if not all customers require to be visited in the randomly selected sample, the deadline is not violated and the costs can be bounded from above by $2(n+1)$, which is an upper bound on the total travel time for any sample. This second case occurs with a probability of $1 - 2^{-n}$.

To show that the *Monte Carlo sampling approximation* is not able to compute a reasonable approximation using $g(n)$ samples with a high probability, we further investigate the second case. If we have $g(n)$ samples, the probability that the deadline at customer v_n is met in all of these samples is at least $1 - g(n)2^{-n}$. The upper bound on the total travel time is an upper bound for the costs computed by the *Monte Carlo sampling approximation* in this case. That means we have

$$f_{\text{ptspd}}^{\text{mcs},g(n)}(\tau) \leq 2(n+1)$$

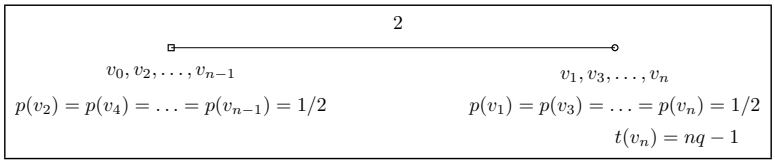


Figure 4.6. An Euclidean instance of the PTSPD that cannot be approximated within a factor of 2^n by the Monte Carlo sampling approximation with high probability (regarding the chosen samples).

with a probability of at least $1-g(n)2^{-n}$ regarding the chosen samples. Using a penalty value for customer v_n of $h(v_n) = 2^{2n}2(n+1)$ we have with the same probability

$$\begin{aligned} f_{\text{ptspd}}(\tau)/f_{\text{ptspd}}^{\text{mcs},g(n)}(\tau) &\geq 2^{-n}h(v_n)/2(n+1) \\ &= 2^{-n}2^{2n}2(n+1)/2(n+1) \\ &= 2^n, \end{aligned}$$

which concludes the proof. □

4.3 Inapproximability Results for the Dependent PTSPD

As stated before, the approximability of the PTSPD objective function is still a very interesting open problem. In this section we show that the objective function of a slightly more general problem cannot be approximated within any reasonable factor. Here we allow certain dependencies among the random events. In fact, we consider only dependencies in which two random events are either independent of each other, always occur together or are mutually exclusive. We call the resulting problem the Dependent Probabilistic Traveling Salesman Problem with Deadlines (Dependent PTSPD). We omit a formal definition of this problem here, since it is almost identical to that of the PTSPD. Just note that the dependencies can be efficiently modeled by pairs of mutually disjoint sets.

Using the additional structural properties given by the possible dependencies among the random events we can show that the objective function of the Dependent PTSPD cannot be approximated within a factor of 2^n. To prove this

statement we use a polynomial reduction from the well-known decision problem 3SAT (Karp [1972]).

For the 3SAT problem we have given a set of boolean variables Y. A literal of a variable $y \in Y$ is either the variable y itself or the negation of that variable, $\bar{y}$. Additionally, m clauses with exactly 3 literals over the variables in Y are given. The k-th clause can be represented as $x_{k,1} \vee x_{k,2} \vee x_{k,3}$, where $x_{k,1}, x_{k,2}, x_{k,3}$ are literals over the variables in Y. A clause is satisfied for a given assignment of the variables, if at least one of the literals of that clause evaluates to 1. We further say that the formula given by the m clauses is satisfied for a given assignment of the variables, if all clauses are satisfied by that assignment. The 3SAT problem is now to decide for a given formula, if an assignment of the variables exists that satisfies this formula. Let us state this more formally.

Problem 8 (3SAT)**.** *Given a set Y of n variables and a formula of m clauses $x_{1,1} \vee x_{1,2} \vee x_{1,3}$, ..., $x_{m,1} \vee x_{m,2} \vee x_{m,3}$, each with exactly 3 literals over the variables in Y, the problem is to decide if there exists an assignment of the variables which satisfies the given formula.*

The overall idea for showing the desired inapproximability result is to construct an Euclidean instance for the DEPENDENT PTSPD with imposing a deadline on only a single customer. If there does not exist an assignment for the 3SAT instance satisfying all the clauses, the deadline is always met. On the other hand, if there exists an assignment satisfying all the clauses, the deadline is violated at least with a very small probability. Using an appropriate value for the penalty value, we can adjust the factor between the costs for these two cases to an exponentially large value. Therefore a 2^n approximation for the DEPENDENT PTSPD objective function could be used to decide the original 3SAT instance. We formalize this idea in the proof of the following theorem.

Theorem 20. *The objective function of the* DEPENDENT PTSPD *cannot be approximated within a factor of 2^n.*

Proof. Given an instance for the 3SAT problem with m clauses over n variables, we construct the following instance for the DEPENDENT PTSPD. For each clause we construct a gadget as depicted for clause k in figure 4.7. This gadget consists of two vertices v_{k-1} and v_k and a location with three vertices which correspond to the literals that occur in clause k. The travel time between these three locations is 1. The probabilities for the vertices v_{k-1} and v_k are 1, the probabilities for the vertices corresponding to the literals are $1/2$. The gadgets for the different clauses are then connected at the common vertices v_i. Additionally, we make

use of the dependencies among random events for the vertices corresponding to literals. If a literal occurs multiple times, the corresponding vertices are either all present or all not present. If a literal occurs in negated and non-negated form, exactly one of the corresponding vertices are present. In this way we have a bijection between variable assignments for the original 3SAT instance and realizations of the random events for our instance of the DEPENDENT PTSPD.

As we stated earlier we want to show that if there exists an assignment satisfying all the clauses, a deadline is violated with positive probability, while if such an assignment does not exist, that deadline is not violated at all. To show this, we define our a priori tour τ to start at the depot v_0, visit the vertices corresponding to the literals of the first clause, continue with v_1 and the vertices corresponding to the literals of the second clause, and so on. At the end the tour finishes at the depot after vertex v_m has been visited. Additionally, we set the deadline of vertex v_m to $2m - 1/2$.

Let us assume that we have an assignment satisfying all the clauses. Then there exists a realization of the random events which corresponds to this assignment and which occurs with a probability of 2^{-n}. Since the assignment is satisfying all the clauses there is at least one of the vertices that correspond to literals in each of the gadgets which requires to be visited. That means that the travel time within each gadget is 2. We have m gadgets in total and therefore the arrival time at vertex v_m is $2m$. In this case the deadline at vertex v_m is violated. The costs for our solution under the assumption that a truth assignment exists can be bound by

$$f_{\text{true}}(\tau) \geq 2^{-n} h(v_m).$$

On the other hand, if there does not exist an assignment satisfying all the clauses, the deadline at vertex v_m is never violated. This is due to the fact that for any realization of the random events there exists at least one gadget with a total travel time of only 1. This gadget corresponds to a clause which is not satisfied in the corresponding variable assignment. Therefore the arrival time at customer v_m is bounded from above by $2m - 1$ for any realization of the random events. Using an upper bound of the expected travel time, we can bound the costs for our solution under the assumption that no truth assignment exists by

$$f_{\text{false}}(\tau) \leq 3m.$$

Using a penalty value for vertex v_m of $h(v_m) = 2^{2n}4m$ the ratio between these two costs can be bounded from below by

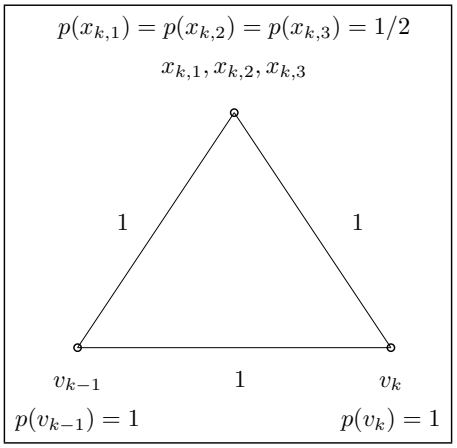


Figure 4.7. The gadget for the instance of the Dependent PTSPD used in theorem 20 for the k-th clause of the original 3SAT instance.

$$\begin{aligned} f_{\text{true}}(\tau)/f_{\text{false}}(\tau) &\geq 2^{-n}h(v_m)/3m \\ &= 2^{-n}2^{2n}4m/3m \\ &> 2^n. \end{aligned}$$

Since the ratio between the costs for a solution under the assumption that a truth assignment exists and the costs for a solution under the assumption that no such truth assignment exists is bounded from below by 2^n, a 2^n approximation algorithm for the DEPENDENT PTSPD objective function could be used to the decide the original 3SAT instance. This concludes the proof. □

4.4 Discussion and Conclusions

In this chapter we have shown that various computational tasks related to the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES are #P-hard even for Euclidean instances: The computation of probabilities with which deadlines are violated, the evaluation of solutions, *delta evaluation* in *reasonable local search neighborhoods*, the decision variant of the PTSPD and the optimization variant of the PTSPD. To our knowledge the former results are the first results of this kind for stochastic vehicle routing problems. The latter results strengthen the hardness results that the PTSPD inherits from the TSP. We have further investigated the worst-case approximation guarantees obtained by the existing

approximations for the PTSPD objective function. None of those approaches is able to approximate the PTSPD objective function even within an exponential approximation ratio. Finally, we focused on the DEPENDENT PTSPD, a slightly more general problem than the PTSPD. We were able to derive a strong inapproximability result for the objective function of the DEPENDENT PTSPD.

The results presented here open some directions for further research. First of all, the approximability of the PTSPD objective function is a very interesting open problem with practical relevance. Another very interesting topic is the relation of counting problems and stochastic combinatorial optimization problems. It seems that there is a deep connection between these two classes of problems and further research in this direction looks promising. Apart from that it would be also of interest to transfer the results presented in this article to other stochastic combinatorial optimization problems or other stochastic vehicle routing problems. Maybe it is also possible to use the new results to strengthen the inapproximability results for the optimization variant of the PTSPD.

Chapter 5

Heuristics for the Probabilistic Traveling Salesman Problem

In this chapter we start with the more practically oriented part of the thesis. In fact, we focus on the development of efficient heuristics for the PROBABILISTIC TRAVELING SALESMAN PROBLEM. As we have stated already in chapter 1, instances of up to 50 customers can be solved to optimality using a branch-and-cut algorithm (Laporte et al. [1994]). Although this work dates back almost 20 years, no significant improvements could be obtained meanwhile and it is still not possible to solve instances of realistic size to optimality. Therefore, the development of efficient optimization approaches for the PTSP is focusing a lot on heuristics. The current state-of-the-art methods are heuristics based on strong local search algorithms combined with an approximation of the PTSP objective function (Birattari et al. [2008a,b]; Balaprakash et al. [2009a,b, 2010]). We believe that this is a very promising approach, not only for the PTSP, but for stochastic vehicle routing problems in general. Therefore, our work is continuing in this direction.

We start with the development of efficient local search algorithms for the PTSP. First, we examine different approaches for the approximation of the PTSP objective function. Then we give an overview about common local search neighborhoods for routing problems. After that we introduce several local search algorithms using different local search neighborhoods and different approximations of the PTSP objective function. In an extensive computational study we compare their performance on common benchmark instances and identify the strongest local search algorithm. Finally, we use the strongest local search algorithm within some more enhanced heuristics. Comparisons with state-of-the-art approaches clearly show that we were able to obtain new state-of-the-art heuris-

tics for the PTSP in this way. The results presented in this chapter are based on the publications Weyland et al. [2009a] and Weyland et al. [2009b].

5.1 Approximations for the PTSP Objective Function

As we have stated before, the state-of-the-art heuristics for the PTSP are using approximations of the objective function to evaluate solutions. Heuristics are known to be robust against slight noise in the objective function and therefore a significant amount of computational time can be saved in this way. The exact evaluation of solutions for the PTSP can be performed in a computational time of $\mathcal{O}(n^2)$ for solutions of size n (Jaillet [1985]). In this section we present three different approximations for the PTSP objective function which achieve a linear computational time (Branke and Guntsch [2004]; Birattari et al. [2008b]). Two of them are based on the closed-form mathematical expression introduced in chapter 1, the third one is based on Monte Carlo sampling.

5.1.1 The Depth Approximation

The *depth* approximation has been introduced in Branke and Guntsch [2004] and is directly based on the closed-form mathematical expression of the objective function introduced in chapter 1. The costs of a tour represented by the permutation $\tau : \langle n \rangle \to V$ is the sum of the costs caused by all of the edges. We state this formula here again for convenience.

$$\begin{aligned} f_{\text{ptsp}}(\tau) \;=\; & \sum_{i=1}^{n} \sum_{j=i+1}^{n} d(\tau_i, \tau_j) p(\tau_i) p(\tau_j) \prod_{k=i+1}^{j-1} (1 - p(\tau_k)) \\ & + \sum_{i=1}^{n} \sum_{j=1}^{i-1} d(\tau_i, \tau_j) p(\tau_i) p(\tau_j) \prod_{k=i+1}^{n} (1 - p(\tau_k)) \prod_{k=1}^{j-1} (1 - p(\tau_k)) \end{aligned}$$

The key observation for the *depth* approximation is that edges whose nodes are distant from each other in the given tour contribute less to the overall costs than edges whose nodes are nearby in the given tour. This is due to the simple fact that the probability for the occurrence of one of the former edges in an a posteriori solution is smaller than the probability for the occurrence of one of the latter edges. The idea is now to modify the mathematical formulation and to truncate the second sum after adding a total of $d \in \mathbb{N}, d < n$, summands. The mathematical formulation changes to

$$f_{\text{ptsp}}^{\text{depth}}(\tau) = \sum_{i=1}^{n} \sum_{j=i+1}^{\min\{i+d,n\}} d(\tau_i,\tau_j)p(\tau_i)p(\tau_j) \prod_{k=i+1}^{j-1}(1-p(\tau_k)) + \sum_{i=1}^{n} \sum_{j=1}^{i+d-n} d(\tau_i,\tau_j)p(\tau_i)p(\tau_j) \prod_{k=i+1}^{n}(1-p(\tau_k)) \prod_{k=1}^{j-1}(1-p(\tau_k)).$$

The computational time for this method drops to $\mathcal{O}(dn)$ and is therefore linear in the input size for a fixed value of d. The approximation accuracy of the *depth* approximation for usage in heuristics is usually sufficient for values of d between 20 and 40.

5.1.2 The Threshold Approximation

The *threshold* approximation is based on the same idea as the *depth* approximation (Branke and Guntsch [2004]). But instead of truncating the sum at a different depth, the sum is truncated as soon as the probability for the corresponding edges drops between a given threshold value $t \in (0,1)$. Since the number of summands varies due to this property, the mathematical formulation for this approximation is a little bit cumbersome. For a given threshold value $t \in (0,1)$ we define for all $i \in \langle n \rangle$ the set

$$T(i) = \left\{ j \in \{i+1,\ldots,n\} : p(\tau_i)p(\tau_j) \prod_{k=i+1}^{j-1}(1-p(\tau_k)) \geq t \right\} \cup \left\{ j \in \{1,\ldots,i-1\} : p(\tau_i)p(\tau_j) \prod_{k=i+1}^{n}(1-p(\tau_k)) \prod_{k=1}^{j-1}(1-p(\tau_k)) \geq t \right\}.$$

The formula for the *threshold* approximation can then be stated as follows.

$$f_{\text{ptsp}}^{\text{thr.}}(\tau) = \sum_{i=1}^{n} \sum_{j \in \{i+1,\ldots,n\} \cap T(i)} d(\tau_i,\tau_j)p(\tau_i)p(\tau_j) \prod_{k=i+1}^{j-1}(1-p(\tau_k)) + \sum_{i=1}^{n} \sum_{j \in \{1,\ldots i-1\} \cap T(i)} d(\tau_i,\tau_j)p(\tau_i)p(\tau_j) \prod_{k=i+1}^{n}(1-p(\tau_k)) \prod_{k=1}^{j-1}(1-p(\tau_k))$$

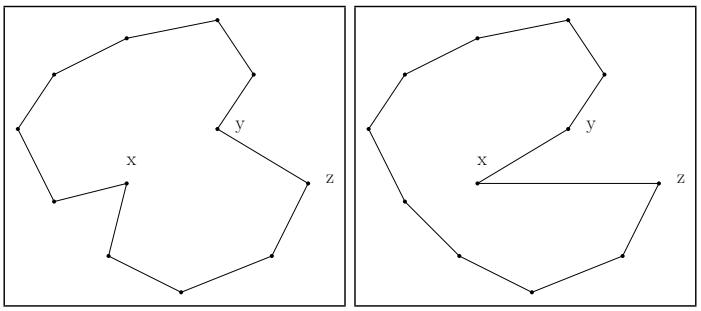
The computational time for the *threshold* approximation depends heavily on the probabilities of the different customers. For homogeneous probabilities it

drops to a linear runtime, for heterogeneous probabilities a rigorous analysis is extremely difficult. For practical applications it is only of importance that also in these cases a significant amount of computational time can be saved with respect to an exact evaluation.

5.1.3 The Monte Carlo Sampling Approximation

The evaluation of the PTSP objective function is equivalent to compute an expected value and therefore it is quite natural to consider an approximation based on Monte Carlo sampling (Shapiro [2003]). This approximation has been successfully used for the PTSP in Birattari et al. [2008b]. Formally, we can describe this approximation approach in the following way. Let X be the set of all a priori solutions and let Ω be the set of all scenarios. Here a scenario is the realization of the stochastic events corresponding to the presence/absence of customers. Furthermore, let $p : \Omega \to [0,1]$, with $\sum_{\omega\in\Omega} p(\omega) = 1$, represent the probabilities of the different scenarios and let $c : X \times \Omega \to \mathbb{R}^+$ represent the costs of the a posteriori solutions for all combinations of solutions and scenarios. Note, that we can formulate the PTSP objective function in this way as $f_{\text{ptsp}}(\tau) = \sum_{\omega\in\Omega} c(\tau,\omega)p(\omega)$ for all $\tau \in X$. Now we use Monte Carlo sampling to estimate the exact objective function. For this purpose, we sample s identically and independently distributed scenarios, $\omega_1, \omega_2, \ldots, \omega_s$, according to the probability distribution given by p. Then $f^{\text{mcs}}_{\text{ptsp}}(\tau) = s^{-1} \sum_{i=1}^{s} c(\tau,\omega_i)$ gives us an estimation for the exact objective function. Since the computation of the costs of the a posteriori tour for a given solution and a given scenario requires a runtime of $\mathcal{O}(n)$, the *Monte Carlo sampling* approximation has a total computational time of $\mathcal{O}(ns)$. For a fixed number of samples, this is linear in the size of the solution.

This approach could lead to some difficulties if it is directly used within heuristics. While evaluating a single solution or comparing multiple solutions, different sets of samples can lead to different results. For example, it could lead to the phenomenon of cycling in a local search algorithm, where a solution τ_1 is better than a solution τ_2 for one set of samples, while for another set of samples the solution τ_1 is worse than the solution τ_2. Furthermore, for each solution evaluation s new samples have to be generated, which requires additional random numbers and consumes additional computational time. To overcome these difficulties s fixed samples can be used for all the solution evaluations, or at least for a certain number of evaluations, after which the samples are replaced by new ones. In this case the heuristic is no longer optimizing the exact objective function, but like for the other approximations a slightly perturbed one.



(a) the initial tour (b) the resulting tour

Figure 5.1. Illustration of a move in the 1-shift local search neighborhood. Part (a) shows the initial situation. Then node x is removed from the tour and reinserted between nodes y and z. The resulting tour is shown in part (b).

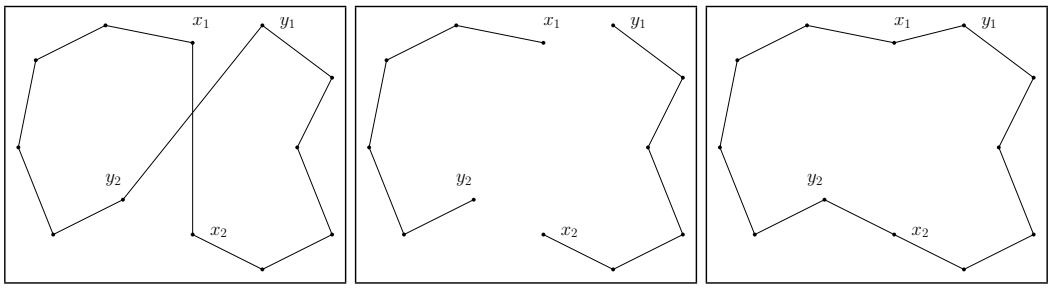
5.2 Local Search Neighborhoods

In this section we start with a detailed presentation of common local search neighborhoods for routing problems. These local search neighborhoods are of great importance for this thesis and are frequently used throughout the remainder of it. For a more comprehensive overview of such local search neighborhoods in the context of the TRAVELING SALESMAN PROBLEM we refer to Johnson and McGeoch [1997].

We start with a presentation of the *1-shift* neighborhood and the *2-opt* neighborhood. These neighborhoods together form the *2.5-opt* neighborhood. Finally, we present the *3-opt* neighborhood, which includes all the other ones as special cases.

5.2.1 The 1-shift Neighborhood

The *1-shift* local search neighborhood is commonly used for routing problems and defined on tours. It consists of all solutions that can be reached by removing one node from the tour and inserting it at a different position in the tour. This neighborhood is defined for tours starting at a specific location and also for tours without a fixed starting point. Given a tour with n locations, the size of the neighborhood is $n(n-2) = \mathcal{O}(n^2)$, since each of the n locations can be inserted at $n-2$ potential insertion points. An illustration of a move in the *1-shift* neighborhood is given in figure 5.1.



(a) the initial tour (b) the two separate parts (c) the resulting tour

Figure 5.2. Illustration of a move in the 2-opt local search neighborhood. Part (a) shows the initial situation. Then the edges between the nodes x_1 and x_2 and between the nodes y_1 and y_2 are removed. The two separate parts are shown in part (b). These two parts are then reconnected to obtain the tour shown in part (c).

5.2.2 The 2-opt Neighborhood

Like the *1-shift* local search neighborhood, the *2-opt* local search neighborhood is commonly used for routing problems and defined on tours. A move in the *2-opt* neighborhood consists of the following steps. First, two non adjacent edges are removed. This splits the tour into two separate parts. Those two parts are then reconnected such that a different tour is obtained. For a given tour with n nodes, we have $n(n-3)/2$ possibilities to remove two non adjacent edges. Without an orientation of the tour, there is only one unique possibility to reconnect the two parts to obtain a different tour. Therefore the size of this neighborhood is $n(n-3)/2 = \mathcal{O}(n^2)$ in this case. If the orientation of the tour matters, we have two different possibilities to reconnect the two parts. For each possibility the original orientation of one of the two parts is reversed. The size of the neighborhood in this case is therefore $n(n-3) = \mathcal{O}(n^2)$. An illustration of a typical move in the *2-opt* local search neighborhood is given in figure 5.2.

5.2.3 The 2.5-opt Neighborhood

The *2.5-opt* local search neighborhood is also commonly used for routing problems. It is a combination of the *1-shift* neighborhood and the *2-opt* neighborhood. It consists of all the moves from these two neighborhoods. For a given tour with n nodes, the size of the *2.5-opt* neighborhood is therefore $n(n-2)+n(n-3)/2 = \mathcal{O}(n^2)$ if the tour does not have an orientation, and $n(n-2)+n(n-3) = \mathcal{O}(n^2)$ if the orientation of the tour matters.

5.2.4 The 3-opt Neighborhood

The *3-opt* neighborhood also belongs to the class of common local search neighborhoods for routing problems. It is similar to the *2-opt* neighborhood, the only difference is that three edges are removed instead of two and that the resulting three separate parts can be combined to a new tour in up to seven different ways (15 if the orientation of the tour matters). In most of the cases this upper bound is obtained. Only if adjacent edges are removed, which is not forbidden for the *3-opt* neighborhood, there might be less possibilities. For a given tour of n nodes, we have $n(n-1)(n-2)/6$ different possibilities to remove three edges and therefore the size of the *3-opt* neighborhood is $\mathcal{O}(n^3)$. Note that this is a higher order of growth compared to the other local search neighborhoods that were introduced before. It is also interesting that the *3-opt* neighborhood contains the other neighborhoods as special cases. The *2-opt* moves are obtained by reconnecting two of the three segments in the same way as they have been before. The *1-shift* moves are obtained in the case where two of the three removed edges were adjacent. An illustrative example for the *3-opt* neighborhood is given in figure 5.3.

5.3 Local Search Algorithms

In this section we present different local search algorithms for the PTSP. These algorithms are based on the local search algorithm proposed in Birattari et al. [2008b]. A high level description of a local search algorithm for optimization problems in general is given in algorithm 1. In the first step we create an initial solution. This can be done by a fast construction heuristic or purely random. As long as this solution is not a local optimum with respect to the local search neighborhood we are using, we replace the current solution with an improving one. If there is no improving solution in the local search neighborhood, we terminate the algorithm and return our current solution.

Algorithm 1 High level description of a local search algorithm.

```
1: Create an initial solution S
2: while S is not a local optimum
       (with respect to the chosen neighborhood) do
3:   Replace S by an improving solution
4: end while
5: return S
```

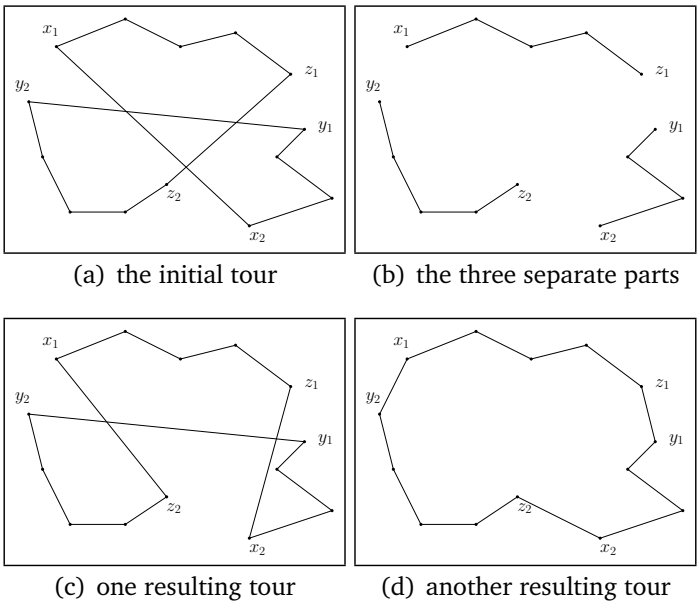


Figure 5.3. Illustration of a move in the 3-opt local search neighborhood. Part (a) shows the initial situation. Then the edges between the nodes x_1 and x_2, between the nodes y_1 and y_2 and between the nodes z_1 and z_2 are removed. The three separate parts are shown in part (b). These parts can then be reconnected to obtain, among others, the tours shown in parts (c) and (d).

For the local search algorithms in this thesis we follow the first improvement strategy with a random exploration of the local search neighborhood. That means we examine the neighbor solutions in a random order and replace the current solution as soon as we find an improving one. Preliminary experiments have shown that this strategy leads to a better performance compared to other common local search strategies. In Birattari et al. [2008b] four techniques were used to further improve the performance. First of all, as we have already indicated, approximations of the PTSP objective function are used for the evaluation of solutions. The usage of these approximations allows us to use another technique, called *delta evaluation*. To compare the current solution with the solutions in its neighborhood, we only compute the difference in the solution costs, instead of evaluating the neighbor solutions from scratch. This technique leads to a major improvement of the computational time. For all the approximations we have introduced, the computational time is reduced by a remarkable factor of n in this way. Then, a technique called *neighborhood lists* is used. Due to some heuristic criterion each node has assigned a list with preferable neighbor nodes. The heuristic criterion in our case is the distance between the nodes. We now limit the local search moves to those moves that establish at least one new edge between a node and another node in its *neighborhood list*. In this way the computational time can be reduced, without decreasing the quality of the final solution significantly. The last technique, which has a similar purpose as the *neighborhood lists* is called *don't look bits*. Each node has assigned a bit, which indicates whether moves including this node are still worth to be examined. A value of *false* indicates that we still consider moves including this node, a value of *true* indicates that we "don't look" for moves including this node anymore. Initially, all moves are considered and the bits are set to *false*. As soon as we have evidence that no move that includes a specific node leads to an improving solution, the corresponding bit for this node is set to *true*. The node is only reactivated if the tour is changed locally around that node. The implementations of the latter two techniques are straightforward and we will not go more into detail here. For the local search algorithms using the *Monte Carlo sampling* approximation we propose another improvement. Every time the current solution is updated, we compute for each sample and each node the predecessor and the successor of the given node in the a posteriori tour imposed by the given sample. This small computation enables us to use *delta evaluation* more efficiently.

What is still open, is the selection of the local search neighborhood and the selection of the approximation of the PTSP objective function. For our experiments we propose the following combinations. The *1-shift* neighborhood is used with the *Monte Carlo sampling* approximation (1shift-delta). The *2.5-opt*

neighborhood is used with the *Monte Carlo sampling* approximation (2.5opt-sampling), the *depth* approximation (2.5opt-depth) and the *threshold* approximation (2.5opt-threshold). Like the *2.5-opt* neighborhood, the *3-opt* neighborhood is used with the *Monte Carlo sampling* approximation (3opt-sampling), the *depth* approximation (3opt-depth) and the *threshold* approximation (3opt-threshold).

Additionally, we use the *2.5-opt* and the *3-opt* neighborhood with a combined approach (2.5opt-combined, 3opt-combined). Here we alternate the local search algorithms using the *threshold* approximation and the *Monte Carlo sampling* approximation for i iterations. Here i is usually a value of between 1 and 3. A short overview about this approach is given in algorithm 2.

Algorithm 2 High level description of the combined local search algorithm.

1: Create an initial solution S
2: **for** i times **do**
3: Apply the local search algorithm using the *threshold* approximation on S and store the result in S
4: Apply the local search algorithm using the *Monte Carlo sampling* approximation on S and store the result in S
5: **end for**
6: **return** S

We have stated earlier in this chapter that one of our goals was to identify the strongest local search algorithm among existing local search algorithms and the approaches introduced here. Here we mean with strongest local search algorithm the algorithm which achieves the best tradeoff between computational time and solution quality. For this purpose we performed computational experiments on different common benchmark instances for the PTSP and with different local search algorithms. In the remainder of this section we will discuss the benchmark instances we have used for our experiments. After that we explain our experimental setup. We finish with a presentation and discussion of the results obtained.

5.3.1 Benchmark Instances

For our experiments we use a large set of different common benchmark instances. Here we distinguish between three different types of instances. We use instances from the TSPLIB benchmark (those instances are available at `www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`) supple-

mented with the probabilities for the customers (tsplib instances), Euclidean instances, in which customers are distributed uniformly at random in the square $[0, 10^6] \times [0, 10^6]$ (uniform instances) and Euclidean instances, in which customers are normally distributed around a certain number of centers, which themselves are distributed uniformly at random in the square $[0, 10^6] \times [0, 10^6]$ (clustered instances). We either have used the same probability for each customer with typical values of $0.05, 0.1, 0.2, \ldots, 0.5$, or we have selected the probabilities uniformly from a fixed interval. Note that these kind of instances are commonly used when different heuristics for the PTSP are compared (Birattari et al. [2008a]; Balaprakash et al. [2010]).

5.3.2 Experimental Setup

The algorithms used for the experiments are those local search algorithms that were discussed in section 5.3. For each of the algorithms we tried different parameters. Usually these parameters (number of samples, depth, threshold value, number of iterations) define a tradeoff between the computational time and the approximation accuracy. A series of 50 independent runs is then performed for each algorithm on each benchmark instance. Here we calculated the average runtime and the average solution quality (using the exact objective function), together with their standard deviations. The algorithms were implemented in C++ and compiled with common optimization flags. All the experiments were performed on Quad-Core AMD Opteron systems running at 2GHz.

5.3.3 Results

Since local search algorithms should both be fast and produce good solutions, the development of these algorithms can itself be seen as a kind of multi-objective optimization problem. That means the goal is to find non-dominated algorithms close to the (unknown) Pareto front. In figure 5.4 the average computational times and the average solution costs of the new local search algorithms with different parameterizations (number of samples, depth, threshold value, number of iterations) for uniform instances of size 1000 with probabilities of $p = 0.1$ are visualized. Note that both axes here are logarithmic.

Representative numerical results, together with a comparison to state-of-the-art local search algorithms from Birattari et al. [2008b], are given in table 5.1 for uniform instances of size 1000 with probabilities $p = 0.1$. The state-of-the-art methods are highlighted in boldface and denoted by *2.5-opt-EEs-100* and *2.5-opt-EEs-1000*. The algorithms are grouped according to the local search

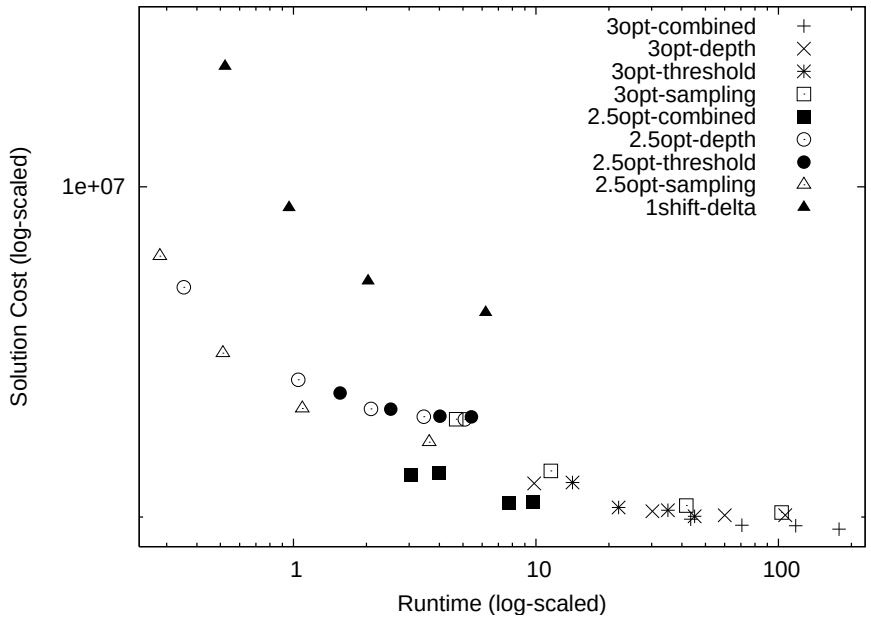


Figure 5.4. Solution cost / runtime surface of local search algorithms with different parameterizations on uniform instances of size 1000 with probabilities $p = 0.1$.

neighborhood that was used and within each group according to the solutions costs. The first column contains the algorithm together with the parameters, the second and third columns contain the average solution cost and the standard deviation of the solution cost, and finally the fourth and fifth columns contain the average runtime and the standard deviation of the runtime. Results for other uniform/tsplib instances and other probability distributions are similar. A fair comparison with state-of-the-art algorithms on clustered instances was not possible, since the solution quality depends heavily on the (randomly generated) instances that are used and therefore the variance with respect to the solution quality is remarkably high. It is not clear, which (randomly generated) instances have been used by other researchers in previous experiments. Due to the high variance, the only way to compare different algorithms on this class of instances, is to use the same subset of instances for all the different approaches. Unfortunately, we could not perform such a comparison, but preliminary results with reimplementations of other approaches suggest, that results are also similar for clustered instances. All in all, each of the state-of-the-art local search algorithms is dominated by at least one of the new local search algorithms with statisti-

cal significance. That means that one of the new algorithms is better in both, runtime and solution quality, with respect to a t-test using a significance level of 95%. The complete numerical results are available at the author's website (`www.idsia.ch/~weyland/`).

Algorithm	Solution Cost		Runtime	
	avg.	s.d.	avg.	s.d.
3opt-combined ($s = 200, t = 0.01, i = 3$)	8974762	71386	117.8	3.3
3opt-combined ($s = 200, t = 0.01, i = 1$)	8993851	68678	43.5	2.3
3opt-threshold ($t = 0.01$)	9002175	77380	45.1	2.9
3opt-depth ($d = 40$)	9016480	88722	30.2	1.9
3opt-sampling ($s = 200$)	9133516	74004	11.5	0.6
2.5opt-combined ($s = 200, t = 0.01, i = 3$)	9041823	86862	9.7	0.3
2.5opt-combined ($s = 200, t = 0.01, i = 1$)	9127135	97081	4.0	0.2
2.5-opt-EEs-1000	9269830	120191	7.6	0.8
2.5opt-threshold ($t = 0.01$)	9292290	125378	5.4	0.4
2.5opt-depth ($d = 40$)	9292524	104783	3.4	0.2
2.5opt-sampling ($s = 200$)	9317110	89689	1.1	0.1
2.5-opt-EEs-100	9462476	89942	1.5	0.1
1shift-delta ($s = 200$)	9703702	199377	2.0	0.3

Table 5.1. Solution costs and runtimes of the new local search algorithms and state-of-the-art local search algorithms on uniform instances of size 1000 with probabilities $p = 0.1$. The former state-of-the-art methods are highlighted in boldface.

Our results clearly indicate that the 2.5opt-combined and the 3opt-combined local search algorithms are the most efficient approaches. In the next section we use the local search algorithms presented in this section within some more complex heuristics. Based on our results we expect the approaches using the 2.5opt-combined and the 3opt-combined local search algorithms to perform best.

5.4 Heuristics

In this section we investigate the performance of random restart local search algorithms (RRLS) and iterated local Search algorithms (ILS). For the underlying local search algorithm we use the new local search algorithms 2.5opt-sampling,

2.5opt-depth, 2.5opt-threshold, 2.5opt-combined, 3opt-sampling, 3opt-depth, 3opt-threshold and 3opt-combined.

The random restart local search algorithm creates iteratively random solutions and applies the underlying local search algorithm on these solutions. After applying the local search algorithm, the solution cost is computed exactly (or approximated with a sufficient accuracy) and compared to the cost of the best solution found so far. If the new solution is better, it replaces the current best solution for the next iteration. At the end the best solution found is returned.

The iterated local search algorithm uses the nearest neighbor heuristic to create an initial solution. Then iteratively the current solution is perturbed using multiple double-bridge moves (Johnson and McGeoch [1997]), and the underlying local search algorithm is applied on that solution. We have chosen double-bridge moves for the perturbation since it is not possible to reverse these moves using moves from the local search neighborhoods we are using. Analog to the RRLS the solution cost is computed exactly (or approximated with a sufficient accuracy) and compared to the cost of the best solution found so far. If the new solution is better, it replaces the current best solution. At the end the best solution found is returned.

A description of these algorithms is given in pseudocode in algorithm 3 and algorithm 4.

Algorithm 3 Random restart local search for the PTSP

```
repeat
    Create a random solution S
    Use a local search algorithm with S as the initial solution and store the
    result in S
    if S is better than the best solution found so far then
        Store S in S*
    end if
until the termination criterion is fulfilled
return S*
```

We denote the resulting approaches by adding a prefix of ils- (for the iterated local search algorithm) or rrls- (for the random restart local search algorithm) to the identifier of the underlying local search algorithm. The iterated local search algorithm using the 2.5opt-combined local search is then denoted by ils-2.5opt-combined and the random restart local search algorithm using the 3opt-threshold local search is denoted by rrls-3opt-threshold.

The computational studies that we will present in the remainder of this sec-

Algorithm 4 Iterated local search for the PTSP

```
1: Create an initial solution S
2: repeat
3:    Use a local search algorithm with S as the initial solution and store the
      result in S
4:    if S is better than the best solution found so far then
5:       Store S in S*
6:    end if
7:    Perform a perturbation of the solution S
8: until the termination criterion is fulfilled
9: return S*
```

tion were performed to compare the efficiency of the heuristics using the new local search algorithms with the efficiency of existing heuristics for the PTSP. For this purpose we performed computational experiments on the same benchmark instances as in the previous section. We will now explain our experimental setup. After that we will finish the section with a presentation and discussion of the results.

5.4.1 Experimental Setup

For the experiments with the random restart local search algorithms and the iterated local search algorithms, we use basically the same setup as for the local search algorithms. Here the execution time on each instance was set to one hour and for each algorithm 20 runs were performed. For every run, intermediate results were stored after each iteration. In this way fair comparisons to other approaches, where runtimes of less than one hour were used, are possible.

5.4.2 Results

Representative numerical results on the tsplib instance rat783 with probabilities $p = 0.1$ and on uniform instances of size 1000 with probabilities $p = 0.1$ are given in table 5.2 and table 5.3, together with a comparison to state-of-the-art algorithms from Balaprakash et al. [2009b]. The state-of-the-art methods are highlighted in boldface and denoted by *pACS+2.5-opt-EEais*, *ACS-EE* and *MMAS-EE*. In both tables the new algorithms are presented in the upper part of the table and the state-of-the-art approaches are presented in the lower part of the table. Within both groups the results are sorted according to the solution costs.

The first column contains the denotation of the algorithm, the second column contains the average solution quality and the last column contains the standard deviation of the solution quality. For a fair comparison with other results from literature we report the results after a runtime of $n^2/1000$ CPU seconds. Note that in table 5.3 one of the approaches from literature used a runtime of 10000 CPU seconds.

The new algorithms based on the combined local search clearly outperform the state-of-the-art algorithms using the same computational time and even outperform the best known PTSP heuristics regarding the final solution quality. The complete numerical results are available at the author's website (`www.idsia.ch/~weyland/`). Results for other probability distributions are similar. Those results are all statistically significant with respect to a t-test using a significance level of 95%. It was not possible to make a fair comparison with state-of-the-art algorithms on clustered instances due to the same reasons as in the computational studies of the previous section.

Algorithm	Solution Cost	
	Average	Standard Deviation
ils-3opt-combined ($s = 200, t = 0.01, i = 3$)	3242.3	4.8
ils-2.5opt-combined ($s = 200, t = 0.01, i = 3$)	3242.5	2.8
rrls-3opt-combined ($s = 200, t = 0.01, i = 3$)	3244.3	6.0
rrls-2.5opt-combined ($s = 200, t = 0.01, i = 3$)	3245.3	3.3
ils-3opt-depth ($d = 40$)	3255.3	5.5
ils-2.5opt-depth ($d = 40$)	3268.6	6.1
rrls-3opt-depth ($d = 40$)	3280.0	12.2
ils-3opt-sampling ($s = 200$)	3286.5	2.9
rrls-3opt-sampling ($s = 200$)	3288.5	3.4
ils-2.5opt-sampling ($s = 200$)	3294.1	3.8
rrls-2.5opt-sampling ($s = 200$)	3354.1	6.6
rrls-2.5opt-depth ($d = 40$)	3373.9	14.9
pACS+2.5-opt-EEais	3258	5
ACS-EE	3260	6

Table 5.2. Solution costs obtained by the RRLS algorithms, the ILS algorithms and state-of-the-art algorithms on the tsplib instance rat783 with probabilities $p = 0.1$ for $n^2/1000$ CPU seconds. The former state-of-the-art methods are highlighted in boldface.

Algorithm	Solution Cost	
	Average	Standard Deviation
ils-2.5opt-combined ($s = 200, t = 0.01, i = 3$)	8868969	53539
rrls-2.5opt-combined ($s = 200, t = 0.01, i = 3$)	8876505	53033
rrls-3opt-depth ($d = 40$)	8885893	51515
ils-3opt-depth ($d = 40$)	8886279	53694
ils-3opt-combined ($s = 200, t = 0.01, i = 3$)	8887252	58646
rrls-3opt-combined ($s = 200, t = 0.01, i = 3$)	8895624	55689
ils-2.5opt-depth ($d = 40$)	8922169	57428
ils-3opt-sampling ($s = 200$)	8977799	61330
ils-2.5opt-sampling ($s = 200$)	8987609	51497
rrls-3opt-sampling ($s = 200$)	8990448	54652
rrls-2.5opt-depth ($d = 40$)	9022913	57942
rrls-2.5opt-sampling ($s = 200$)	9079896	57086
pACS+2.5-opt-EEais	8906639	49546
ACS-EE	8912265	49076
MMAS-EE (10000 CPU seconds)	8884442	73941

Table 5.3. Solution costs obtained by the RRLS algorithms, the ILS algorithms and state-of-the-art algorithms on uniform instances of size 1000 with probabilities $p = 0.1$ for $n^2/1000 = 1000$ CPU seconds. The former state-of-the-art methods are highlighted in boldface.

It is obvious, that the algorithms using the combined local search approach are the strongest. But it is not clear, if the approaches based on 3-opt local search algorithms are superior to the approaches based on 2.5-opt local search algorithms and, if the iterated local search algorithms perform significantly different from the random restart local search algorithms. Further investigations, focusing on larger runtimes and/or larger input instances, are necessary to resolve this problem. It is also possible that the comparison of absolute solution quality values is the limiting factor here. The development of strong lower bounds for the optimal solution quality would allow a comparison of relative solution qualities (with respect to the optimal value) and probably in this case a difference between the approaches will become apparent.

5.5 Discussion and Conclusions

In this chapter we have presented new local search algorithms for the PTSP. These algorithms are not dominated by any existing local search algorithm for the PTSP in a Pareto sense on common benchmark instances. Many of these algorithms require more computational time, but therefore they produce better solutions. Using these new local search algorithms in a random restart local search algorithm and an iterated local search algorithm, we obtained new state-of-the-art heuristics for the PTSP. It remains an open question, whether the random restart local search heuristic and the iterated local search heuristic perform better with the 3-opt local search operator or with the 2.5-opt local search operator.

A possibility for future research would be to replace the local search algorithms with more elaborate methods like tabu search or simulated annealing. The previous state-of-the-art algorithms for the PTSP (Balaprakash et al. [2009b]) and also some other approaches (Branke and Guntsch [2004]) were based on a hybridization of ant colony optimization with local search algorithms. Therefore, an additional possibility for further research would be to use ant colony optimization or other metaheuristics in combination with the new local search algorithms.

Since the differences regarding the solution qualities are getting smaller for recently introduced approaches, it would be very interesting to develop techniques to obtain strong lower bounds for the solution qualities. With such lower bounds, different approaches could be compared in a better way. Furthermore, they would provide a measure of the potential for further improvements.

Chapter 6

Heuristics for the Probabilistic Traveling Salesman Problem with Deadlines

This chapter deals with the development of efficient heuristics for the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES (Campbell and Thomas [2008b]). For this purpose we try to use our experience with the development of efficient heuristics for the PROBABILISTIC TRAVELING SALESMAN PROBLEM, which was the central topic of the previous chapter. The underlying publications for this chapter are Weyland et al. [2011b] and Weyland et al. [2012b].

Although the two problems seem to be quite similar, there are significant differences as we have seen already in chapter 4. Many different computational tasks related to the PTSPD are #P-hard: the evaluation of solutions, the optimization variant, the decision variant and also delta evaluation in reasonable local search neighborhoods. The situation is completely different for the PTSP. There the evaluation of solutions can be performed in polynomial time. Furthermore, the decision variant is NP-complete and the optimization variant is NP-hard. The introduction of time dependencies in terms of deadlines changes the computational complexity in a fundamental way. As for the PTSP, only very small instances of the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES can be solved to optimality using a reasonable amount of runtime. So far, the only alternative approach is to use heuristics. The main difficulty here is the computationally demanding objective function of the PTSPD. Therefore, different analytical approximations for the objective function have been proposed in Campbell and Thomas [2009]. In this chapter we introduce an approximation for the objective function of the PTSPD based on Monte Carlo sampling and

using the novel method of quasi-parallel evaluation of samples. With comprehensive computational studies we reveal the efficiency of this approximation. Additionally, based on this new approximation, we present an improved local search algorithm and a random restart local search algorithm for solving the PTSPD and provide an extensive computational study on a large set of benchmark instances.

The remainder of this chapter is organized as follows. We first introduce in detail an approximation for the PTSPD objective function based on Monte Carlo sampling. Then we propose an improvement for this approach, the quasi-parallel evaluation of samples. We investigate the required computational time and present some techniques and improvements regarding practical applications of this method. After that we compare the new approximation of the objective function with existing approximations, following the approach in Campbell and Thomas [2009]. Based on the potential revealed in those experiments, we then propose an improved local search algorithm and a random restart local search algorithm for solving the PTSPD. Here we also present an extensive computational study of these heuristics on common benchmark instances for the PTSPD. Finally, we finish the chapter with a discussion of the results and with conclusions.

6.1 An Approximation for the PTSPD Objective Function using Monte Carlo Sampling

In this part we show how Monte Carlo sampling can be used for an efficient approximation of the PTSPD objective function. We first discuss how the sampling-based approximation for the PTSP objective function of chapter 5 can be adapted for the PTSPD objective function. After that we introduce the novel method of quasi-parallel evaluation of samples. With an analysis of the asymptotic computational time and with additional computational studies we show that this method leads to significant speed-ups for the approximative evaluation of PTSPD solutions compared to the sequential sampling-based approach.

6.1.1 The Basic Approximation of the Objective Function based on Monte Carlo Sampling

The Monte Carlo sampling approximation of the PTSPD objective function is analog to the Monte Carlo sampling approximation of the PTSP objective func-

tion which has been discussed in chapter 5 (Birattari et al. [2008a]). Therefore, we only give a brief overview of this approach and discuss the modifications that are necessary to adapt this approach for the PTSPD.

As for the PTSP, we use Monte Carlo sampling to sample s scenarios according to the given probabilities for the customers' presence. For a given a priori tour, we then compute the costs of the a posteriori tours for each of the s scenarios and use the average of these costs as an approximation of the exact costs. Given an a priori solution for the PTSPD and a sample, we can compute the costs of the a posteriori tour for the given solution and the given sample in a straightforward way in linear time $\mathcal{O}(n)$. Here we follow the approach for the PTSP (Birattari et al. [2008a]) with a modification for the computation of the penalties for missed deadlines. We just process the customers in the order given by the a priori tour and for each customer that is present in the given sample, we add the travel costs between the previously visited customer and the processed customer to the overall costs, we update the total travel time and, in case of a deadline violation, we add the corresponding penalty to the overall costs. At the end we just add the travel costs between the last visited customer and the depot to the overall costs. Since we have to examine s samples, the total computational time for this approach is $\mathcal{O}(ns)$, including the time to generate the s samples. In this approach the number of samples can be used to adjust the runtime versus the approximation accuracy. For the usage in heuristics we will use s fixed samples for all the solution evaluations due to the issues already discussed in chapter 5.

Additionally, we want to mention at this point that most of the sampling-based heuristics for the PTSP rely on the combination of the Monte Carlo sampling approximation of the objective function with delta evaluation, an important speed-up technique for the comparison of two solutions in a local search neighborhood (Birattari et al. [2008a]). Unfortunately, it is not possible to use delta evaluation in the context of the PTSPD. While the effect of a small change in a solution is bounded locally for the PTSP, this is not longer the case for the PTSPD due to the time dependencies introduced by the deadlines. This is one of the reasons for developing the quasi-parallel evaluation of samples which will be described in the following section.

6.1.2 Quasi-parallel Evaluation of Samples

An important improvement for the computational time of the approach discussed above can be achieved by using a quasi-parallel approach to evaluate the samples. We discuss this technique for the PTSPD RECOURSE I with propor-

tional penalties, but it also works with the other models and it can easily be used for other stochastic vehicle routing problems in which the presence of customers is stochastic. In principle it could also be used for the PTSP, but most of the approaches for the PTSP rely on a strong local search using the sampling-based approximation of the objective function in combination with delta evaluation (see chapter 5). Since the quasi-parallel evaluation of samples is not compatible with delta evaluation it is not possible to further improve those approaches. Nonetheless, it could be an interesting technique also for the PTSP, especially for approaches that rely on the full evaluation of solutions (and are so far not competitive with approaches based on local search). In the remaining part of this section we discuss the quasi-parallel evaluation for the PTSPD RECOURSE I with proportional penalties in detail. Note that only minor changes are necessary to use this technique for the other models. Additionally, we give a theoretical analysis of the computational time required by this method. A computational study comparing the efficiency of the sequential method with that of the quasi-parallel evaluation concludes this section.

The costs of a solution for a specific sample depend only on the customers that are present in that particular sample. This observation is the basis for the quasi-parallel evaluation of samples. Instead of evaluating the samples sequentially, the customers are processed in the order given by the solution and for each customer only those samples are considered, in which this customer is present. To apply this approach, additional data structures are required. For each customer $v \in V \setminus \{v_1\}$ we create a set R_v containing the samples in which the customer v is present and needs to be visited. This can already be done during the creation of the samples. Moreover, we use for each sample r the variables last_r, time_r and penalty_r. During the run of the quasi-parallel evaluation these variables store the last position visited for sample r (last_r), the cumulative travel time for sample r (time_r) and the cumulative penalty for sample r (penalty_r). Additionally, we refer with R to the set of the available samples. Using those notations the pseudocode for the quasi-parallel evaluation of samples is presented in algorithm 12 for the recourse model with proportional penalties. At the beginning of an evaluation, we set for each sample r the last location visited to the depot, v_1, and the cumulative travel time as well as the cumulative penalty to 0. Now we process the customers in the order given by the a priori solution. At each customer v we only have to consider the samples contained in the set R_v, since the customer v is skipped in the other samples. For each sample r in R_v we first update time_r by adding the travel time between the last visited location, last_r, and v. If the deadline for this customer is not met, we add the corresponding penalty to the cumulative penalty for this sample, penalty_r. Finally we set

the last location visited, last_r, to v. After all customers have been processed in this way, we have to add for each sample r the travel time between the last customer visited and the depot to the cumulative travel time, in order to close the tour (i.e. the travel time between last_r and v_1). Adding the average travel times and the average penalties yields the desired estimation of the objective function.

Algorithm 5 Quasi-parallel evaluation of samples

1: **for** each sample $r \in R$ **do**
2: $\quad \text{last}_r := v_1$
3: $\quad \text{time}_r := 0$
4: $\quad \text{penalty}_r := 0$
5: **end for**
6: **for** each customer v (in the order given by the a priori solution) **do**
7: $\quad$ **for** each sample $r \in R_v$ **do**
8: $\quad\quad \text{time}_r := \text{time}_r + d(\text{last}_r, v)$
9: $\quad\quad$ **if** $\text{time}_r > t(v)$ **then**
10: $\quad\quad\quad \text{penalty}_r := \text{penalty}_r + h(v) \cdot (\text{time}_r - t(v))$
11: $\quad\quad$ **end if**
12: $\quad\quad \text{last}_r := v$
13: $\quad$ **end for**
14: **end for**
15: **for** each sample $r \in R$ **do**
16: $\quad \text{time}_r := \text{time}_r + d(\text{last}_r, v_1)$
17: **end for**
18: **return** $|R|^{-1} \sum_{r \in R} \text{time}_r + |R|^{-1} \sum_{r \in R} \text{penalty}_r$

We denote this approach with the term "quasi-parallel" since all samples are evaluated at the same time, but not using multiple computational devices like in the field of parallel computing.

In the remainder of this section we analyze the asymptotic computational time for the quasi-parallel evaluation of samples and perform computational studies comparing the computational times of the quasi-parallel evaluation of samples and the sequential approach.

6.1.3 Theoretical Analysis of the Quasi-Parallel Evaluation of Samples

Whereas the computational time for the sequential approach is $\mathcal{O}(ns)$, the computational time using the technique of quasi-parallel evaluation drops to $\mathcal{O}(n +$

$s+q$), where q is the sum of the number of customers that are present and need to be visited over all samples. We will emphasize this fact in the following theorem, where q is expressed in terms of the instance size n, the number of samples s and the average probabilities of the customers, $\bar{p}$.

Theorem 21. *The expected computational time (with respect to the used samples) for the evaluation of a solution using the technique of quasi-parallel sampling is $\mathcal{O}(n+s+ns\bar{p})$. Here $\bar{p}$ is the mean of the probabilities of the customers.*

Proof. The initialization of the data structures and the summation at the end require a computational time proportional to s, which explains the second term of the total computational time. Since we iterate over n customers (even if not all of them need to be visited in the used samples), we have to add for technical reasons a term of n for the total computational time. Now let us focus on the expected computational time of the actual computations. In total we have to process q customers, where q is the total number of customers (over all samples) that need to be visited. The computational time for each of the q customers is constant. This results in an expected computational time of $\mathbb{E}(q)$. The expectation of q can be calculated in the following way. Here $Z_{r,v}$ describes the random variable that has a value of 1 if customer v needs to be visited in sample r and a value of 0 if customer v does not need to be visited in sample r. Additionally, we denote by $\sum_r$ the summation over all samples and by $\sum_v$ the summation over all customers.

$$\begin{aligned}\mathbb{E}(q) &= \mathbb{E}\left(\sum_r \sum_v Z_{r,v}\right) = \sum_r \sum_v \mathbb{E}(Z_{r,v}) \\ &= \sum_r \sum_v p(v) = \sum_r n\bar{p} = sn\bar{p}\end{aligned}$$

□

The computational time is dominated by the last term for typical instances and reasonable numbers of samples and therefore the expected computational time (with respect to the used samples) for the complete quasi-parallel evaluation is $\mathcal{O}(ns\bar{p})$ compared to the computational time of $\mathcal{O}(ns)$ for the sequential approach. Note that $\bar{p}$ is significantly smaller than 1 in reasonable instances for the PTSPD. That means, the asymptotic computational time is improved with respect to the sequential approach, in particular if the probabilities associated with the customers are rather small.

6.1.4 Computational Studies for the Quasi-Parallel Evaluation of Samples

Since constant factors, which are relevant for practical applications, are hidden within the asymptotic notation of the computational time, we additionally perform a computational study to compare the sequential method and the quasi-parallel evaluation in terms of efficiency. More in detail, we compare the number of samples that can be evaluated by both methods within a certain fixed amount of time on benchmark instances for the model with proportional penalties. For this purpose we use common benchmark instances (Campbell and Thomas [2008b]) of sizes 40, 60 and 100 with four different types of customer probabilities (denoted by 0.1, range, mixed, 0.9). These probability types lead to different average probabilities. The benchmark instances are discussed more in detail together with the computational studies later in this chapter. For each combination of instance size and probability type we measure how many samples can be evaluated within a computational time of 60 seconds by both methods. The number of samples used for the evaluations is fixed to 1000. On the one hand, this is a reasonable number of samples for usage within heuristics, on the other hand the performance of both methods depends mainly on the number of customers that need to be visited, and by using 1000 samples statistical fluctuations are negligible. The experiments are performed on a Quad-Core AMD Opteron system running at 2GHz, the same system used for the experiments in the following sections. Table 6.1 summarizes the number of samples (in millions) that can be evaluated per second using the quasi-parallel evaluation of samples and the sequential approach. Additionally, the speed-up gained by using the quasi-parallel evaluation of samples instead of the serial approach is shown. Here the speed-up is the ratio of the number of samples that are evaluated per second using the quasi-parallel approach and the number of samples that are evaluated per second using the sequential approach. For all different instances, even those with an average customer probability of 0.9, significant speed-ups are achieved. The speed-ups are slightly affected by the instance size and tend to increase with larger instances. This is due to the fact that the algorithmic overhead for the quasi-parallel evaluation of samples diminishes with the input size in comparison with the actual evaluation (the main loop in algorithm 12). Nonetheless, they are mostly affected by the customer probabilities and range from a factor of around 3.5 for instances with $\bar{p} = 0.9$ to a factor of around 21 for instances with $\bar{p} = 0.1$.

All in all, the quasi-parallel evaluation leads to a significant improvement of the efficiency for the approximative evaluation of solutions in comparison to

customer probabilities	instance size	serial	quasi-parallel	speed-up
0.1, $\bar{p} = 0.1$	40	1.33	27.10	20.40
	60	0.89	19.20	21.64
	100	0.52	11.35	21.76
range, $\bar{p} \approx 0.5$	40	1.04	6.05	5.80
	60	0.69	4.65	6.75
	100	0.40	2.81	7.01
mixed, $\bar{p} \approx 0.55$	40	1.17	6.67	5.68
	60	0.76	4.17	5.52
	100	0.49	2.6	5.46
0.9, $\bar{p} = 0.9$	40	1.10	3.86	3.49
	60	0.76	2.66	3.52
	100	0.42	1.58	3.78

Table 6.1. The number of samples evaluated per second (in millions) using the quasi-parallel evaluation of samples and the sequential approach, as well as the speed-up obtained by using the quasi-parallel evaluation of samples instead of the sequential approach, for instances of sizes 40, 60 and 100 with the available four different types of customer probabilities denoted by 0.1, range, mixed and 0.9.

the sequential approach. This is underlined by the theoretical analysis of the asymptotic computational time, as well as by our computational studies.

6.2 A Comparison between Approximations for the Objective Function

In the previous section we have introduced an approximation of the PTSPD objective function based on Monte Carlo sampling and using the novel method of quasi-parallel evaluation of samples. In this section we want to compare the performance of our new approximation with the three approximations introduced in Campbell and Thomas [2009]. It has already been shown that they can be used within heuristics in combination with the exact objective function, obtaining solutions of competitive quality while the computational time is reduced dramatically. Since we are also interested in using the new approximation within heuristics, we follow the approach of Campbell and Thomas [2009]. Here the performance of the different approximations and the exact computation of the objective function are compared using a simple local search algorithm. Note that in the computational studies presented in Campbell and Thomas [2009] it was not possible to match the results of the local search algorithm with the exact objective function just by replacing the exact objective function with one of the approximations. Only a combination of the approximations with the exact objective function led to competitive results. As we will see in this section, the situation is different for the sampling based approximation. Here we are able to obtain competitive results without using the exact objective function at all.

6.2.1 Benchmark Instances

The benchmark instances used for our experiments were introduced and used in Campbell and Thomas [2008b, 2009]. They are derived from instances for the Traveling Salesman Problem with Time Windows (TSPTW, Dumas et al. [1995]). More in detail, the instances for the PTSPD are derived from the TSPTW instances with time window lengths of 20 and instance sizes of 40, 60 and 100 in the following way. There are five TSPTW instances available for every instance size of 40, 60 and 100. For each of those instances four different types of customer probabilities are added. The first one uses probabilities taken uniformly at random from $[0, 1]$ and is referred to as *range*. Then two types with homogeneous probabilities of 0.1 and 0.9 are used. The last type is referred to as *mixed*, here customer probabilities are taken uniformly at random from the two

values $\{0.1, 1.0\}$. Moreover, in those instances the penalty values are the same for all customers. We distinguish between two different values for the penalties, 5 and 50, and between two different types of deadlines. The first deadline type is called *early* and uses the starting times of the time windows for the original TSPTW instances as deadlines. The second one is called *late* and uses the finishing times of the time windows as deadlines. In total we have 48 different instance classes (3 different instance sizes, 4 probability types, 2 penalty values, 2 deadline types) consisting of 5 instances each.

6.2.2 Experimental Setup

For the experiments we have implemented the local search algorithm used in Campbell and Thomas [2009]. This is a best improvement local search algorithm using the 1-shift neighborhood (Bertsimas and Howell [1993]). The starting solutions are generated in the same way as in Campbell and Thomas [2009] using a quick heuristic for the non-stochastic variant of the PTSPD. Preliminary experiments have shown that a number of 500 samples provides a good tradeoff between approximation accuracy and computational time. Therefore, we set the number of samples to 500 for the experiments in this section. In total we perform 20 runs on each of the available benchmark instances, which results in 100 runs for each instance class. As before, the experiments are all performed on a Quad-Core AMD Opteron system running at 2GHz. Here we compare our approach with the three approximation-based approaches of Campbell and Thomas [2009]. One approach is based on the recursive computation of the objective function and truncates parts that are supposed to contribute only very small values to the overall costs. This approach is denoted in the following by TC. Another approach is based on the aggregation on a temporal level and denoted by TA. Note that approaches using temporal aggregation have been successfully applied for different problems, including the PTSPD (Campbell and Thomas [2009]) and the PTSP (Campbell [2006]). The last approach, denoted by EA, computes the expected arrival times at customers which are then used for the computation of penalties. These approaches start with a run of the local search algorithm using one of the approximations of the objective function. Then iteratively the last local optimum is used as a starting solution for another run of the local search algorithm using a more accurate approximation of the objective function, until finally the exact objective function is used. Note that for the approach denoted by EA this results in only 2 iterations, while the other approximations allow to gradually adjust the approximation accuracy at the expense of

additional computational time. For further details about those approximations we again refer to Campbell and Thomas [2009].

6.2.3 Results

The results of our comparison are summarized in tables 6.2 and 6.3. Table 6.2 contains the average percental computational time for the local search algorithm using the new approximation with respect to the local search algorithm based on the approximations of Campbell and Thomas [2009]. Following the notation of Campbell and Thomas [2009] we refer to these approaches with TC, TA and EA. Let $\text{time}_A(I)$ denote the average computational time required by algorithm A on the instance I. Then the average computational time for algorithm A (in this case the sampling-based approach) with respect to algorithm B (in this case one of the other approaches) is defined as the average value $\text{time}_A(I)/\text{time}_B(I)$ over all instances I. Here a value of 5% indicates that the new approach requires 5% of the computational time of the other approach. Table 6.3 contains the average percental increase/decrease of the solution costs of the local search algorithm using the new approximation with respect to the local search algorithm based on the approximations of Campbell and Thomas [2009] for different classes of instances. Again we refer to these approaches with TC, TA and EA, following the notation of Campbell and Thomas [2009]. Let $\text{cost}_A(I)$ denote the average cost of the solution obtained by algorithm A on the instance I. Then the average increase/decrease of the solution costs of algorithm A (in this case the sampling-based approach) with respect to algorithm B (in this case one of the other approaches) is defined as the average value $(\text{cost}_A(I) - \text{cost}_B(I))/\text{cost}_B(I)$ over all instances I. That means a value of 5% represents an average increase of the solution costs of 5%, whereas a value of -5% represents an average decrease of the solution costs of 5%. In the part denoted by *with outliers* we are considering all the runs of the heuristic using our new approximation, in the part denoted by *without outliers* we do not consider the 3 worst solutions from each of the 20 runs.

Let us start with the discussion of the computational times. First of all, it is not clear which machines are used for the experiments in Campbell and Thomas [2008b, 2009]. Therefore, we have reimplemented the approach using the exact objective function and performed some preliminary experiments on the machine that was used for our experiments. The computational times (and also the solution costs) reported in Campbell and Thomas [2008b] were matched up to a very small constant factor. Still, we have to respect this in the interpretation of the results. Over all instances, the new approach we propose requires in average

between 5.5% and 7.5% of the computational time of the other approaches. This corresponds to a speed-up of a factor between 13 and 18. The most important difference among the various classes of instances occurs for instances of different sizes. While the average relative computational time is between 7% and 11% for instances of size 40, it drops to a value between 1.5% and 4% for instances of size 100. This shows that the performance of the new method scales much better regarding the size of the instances. All in all, in terms of computational time, the new approach is much more efficient than any of the other approaches.

Now the interesting question is if the new approach is able to obtain solutions of competitive quality. The discussion of this aspect is a little bit more complicated for various reasons and will be performed in the remaining part of this section.

First of all, we can see in table 6.3 that the solutions obtained by our new approach are in average slightly better for probabilities of 0.1, competitive for the probabilities *range*, slightly worse for the probabilities *mixed* and significantly worse for probabilities of 0.9. Apart from that, the performance of the new approach is much better on larger instances. At the first glance those results do not look very promising, especially for the instances with probabilities of 0.9, but a more careful analysis reveals an interesting fact that explains this behavior. Looking at the results of the different runs in detail, one can see that in a very small number of runs, the new approach returns a local optimal solution of very poor quality. Unfortunately, only one or two of those solutions of poor quality are able to significantly influence the average solution costs. This phenomenon occurs even more frequently for instances with customer probabilities of 0.9, which explains the observations made above. Just as an example, for one of the instances with size 100, proportional penalties, early deadlines and a penalty value of 5 all the solutions have costs of around 800, except one solution with costs of around 2100 and another one with remarkably huge costs of around 5800. This observation has already been made in Campbell and Thomas [2009], where bad local optima occur very rarely. In Campbell and Thomas [2009] one run for each instance was performed, whereas we perform 20 runs on each instance. Therefore, it is very likely that the results of our experiments are influenced in a negative way by those local optimal solutions of poor quality. It is hard to say if the average solution costs are affected by the same phenomenon as observed in Campbell and Thomas [2009] or if the occurrence of bad local optima is more problematic for heuristics using the approximation of the objective function based on Monte Carlo sampling. Nonetheless, such solutions of poor quality are only obtained in very few runs. To get an impression about the influence of the local optimal solutions of poor quality, table 6.3 also

instance	TC [in %]	TA [in %]	EA [in %]
all	5.82	5.60	7.49
size 40	6.93	9.23	10.74
size 60	6.09	4.63	7.36
size 100	3.76	1.61	2.82
probability 0.1	4.13	4.23	3.79
probability 0.9	4.50	5.02	9.61
probability range	8.32	7.39	8.87
probability mixed	6.35	5.75	7.72
penalty 5	7.02	7.27	8.44
penalty 50	5.11	4.60	6.93
deadline early	5.54	4.85	4.70
deadline late	6.10	6.35	10.29

Table 6.2. The average relative computational time over different classes of instances for the 1-shift best improvement local search algorithm using the new approximation of the objective function with respect to the 1-shift best improvement local search algorithm using the approximations of Campbell and Thomas [2009].

contains the average percental increase/decrease of the solution costs are shown without considering the 3 worst solutions in each of the 20 runs. Here the results are better in average and even competitive for instances with probabilities of 0.9. Again, the performance on instances with penalty values of 50 and on larger instances is much better.

Repeating the main results, the new approach is able to obtain solutions of competitive or even better quality in most of the runs, whereas the computational time is reduced by a factor of between 13 and 18 in average, and even up to a factor of around 50 for the instances of size 100. In contrast to the other approaches, the exact evaluation of the objective function is no longer needed and the approach is able to obtain solutions of good quality on its own. Nonetheless, in very few runs local optima of poor quality are obtained, especially for instances with customer probabilities of 0.9. This fact will be considered in the following sections for the development of efficient heuristics for the PTSPD based on the new approximation.

instance	with outliers			without outliers		
	TC [in %]	TA [in %]	EA [in %]	TC [in %]	TA [in %]	EA [in %]
all	2.87	4.00	4.03	-2.36	-1.25	-1.22
size 40	5.20	5.50	6.03	-1.07	-0.73	-0.24
size 60	3.69	4.93	4.65	-1.09	0.12	-0.15
size 100	-1.84	0.34	0.11	-6.19	-4.08	-4.31
probability 0.1	-2.33	-0.24	-0.68	-2.85	-0.77	-1.21
probability 0.9	12.38	12.98	12.81	-0.90	-0.34	-0.49
probability range	-0.66	0.13	0.96	-4.04	-3.21	-2.45
probability mixed	2.10	3.11	3.03	-1.63	-0.68	-0.75
penalty 5	4.56	4.55	4.71	1.18	1.18	1.33
penalty 50	1.86	3.66	3.63	-4.48	-2.70	-2.76
deadline early	2.81	4.11	4.36	-2.27	-0.98	-0.76
deadline late	2.94	3.88	3.70	-2.44	-1.51	-1.68

Table 6.3. The average increase/decrease of the solution costs over different classes of instances for the 1-shift best improvement local search algorithm using the new approximation of the objective function with respect to the 1-shift best improvement local search algorithm using the approximations of Campbell and Thomas [2009] considering all the runs (with outliers) and without considering the 3 worst solutions obtained in each of the 20 runs (without outliers).

6.3 Local Search Algorithms for the PTSPD

In this section we will examine different local search algorithms for the PTSPD and identify the most efficient among them. Based on this local search algorithm we then present in the next section a random restart local search algorithm. The decisions taken during the development of the algorithms are highly influenced by our previous work for the PTSP (see chapter 4) and by the results of the previous sections. Heuristics using an approximation of the objective function based on Monte Carlo sampling are widely used for the PTSP (Bianchi and Gambardella [2007]; Balaprakash et al. [2010, 2009a,b]; Birattari et al. [2008b]) and are currently the most efficient heuristics for this problem (Balaprakash et al. [2010]; Weyland et al. [2009b]). Especially, local search algorithms and iterated versions like random restart local search algorithms and iterated local search algorithms are extremely successful. We want to follow this approach also for the PTSPD. First of all, the approximation of the PTSPD objective based on Monte Carlo sampling together with the quasi-parallel evaluation of samples allows for very efficient implementations of local search algorithms. The absolute computational times for such local search algorithms are quite low and within a few seconds, which also allows to use an iterated version of such a local search algorithm. Furthermore, the facts that the local search algorithm returns a solution of very poor quality in rare cases also supports the usage of an iterated version of a local search algorithm. The overall idea is to develop an efficient local search algorithm which is then used within a random restart local search algorithm to efficiently compute solutions of high quality for the PTSPD.

In a first experiment we will therefore compare the efficiency of different local search algorithms with different common local search neighborhoods. We then take the best local search algorithm and integrate it into a random restart local search algorithm in the following section. In section 6.2 we have only used a 1-shift best improvement local search algorithm starting from a quick heuristic solution for the non-stochastic variant of the PTSPD. We used this local search algorithm mainly to compare the new approximation for the objective function with other available approximations. For the PTSP it has been shown that first improvement local search algorithms are much more efficient in terms of computational time, while the quality of the solutions obtained was the same as for the best improvement local search algorithm. Furthermore, it was not sufficient to use the 1-shift local search neighborhood to obtain solutions of high quality for the PTSP. Merely, a combination of the 1-shift neighborhood with the 2-opt neighborhood (resulting in the so-called 2.5-opt neighborhood) turned out to be efficient in terms of computational time and additionally is able to find solutions

	1-shift	2.5-opt
relative solution costs	0.993	0.754
relative computational times	0.304	1.199

Table 6.4. Average relative solution costs and average relative runtimes of the 1-shift, 2-opt and 2.5-opt first improvement local search algorithms using the sampling-based approximation of the objective function with 500 samples with respect to the 1-shift best improvement local search algorithm using the sampling-based approximation of the objective function with 500 samples over all benchmark instances.

of high quality. In this section we compare first improvement local search algorithms using the 1-shift and the 2.5-opt neighborhoods with the best improvement local search algorithm from the last section using the 1-shift neighborhood. Note that the local search algorithms used here are starting from a random solution, since we want to be able to integrate them later into a random restart local search algorithm. We also use the same benchmark instances already introduced in section 6.2.

6.3.1 Experimental Setup

For all the local search algorithms, we perform 20 runs on each of the available benchmark instances, which results in 100 runs for each instance class. We then measure the average computational time as well as the average solution costs over all of the 100 runs per instance class. All experiments are performed on a Quad-Core AMD Opteron system running at 2GHz and the complete numerical results are available at the author's website (`www.idsia.ch/~weyland/`).

6.3.2 Results

The results of this comparison are displayed in table 6.4. Here we see the average relative computational time and the average relative solution costs for the different first improvement local search algorithms with respect to the 1-shift best improvement local search algorithm. More in detail, let $\text{cost}_A(I)$ denote the average cost of the solution obtained by algorithm A on the instance I and let $\text{time}_A(I)$ denote the average computational time required by algorithm A on the instance I. Then the average relative solution costs of algorithm A (in this case the first improvement local search algorithm using the 1-shift/2.5-opt

neighborhood) with respect to algorithm B (in this case the best improvement local search algorithm using the 1-shift neighborhood) is the average value of $\text{cost}_A(I)/\text{cost}_B(I)$ over all instances I. In the same way, the average relative computational time of algorithm A (in this case the first improvement local search algorithm using the 1-shift/2.5-opt neighborhood) with respect to algorithm B (in this case the best improvement local search algorithm using the 1-shift neighborhood) is the average value of $\text{time}_A(I)/\text{time}_B(I)$ over all instances I.

First of all, we see that we can drastically reduce the computational time if we use a first improvement approach instead of a best improvement approach. Additionally, even the costs of solutions are slightly lower for the first improvement local search algorithm. As expected the average relative solution costs are the best for the 2.5-opt first improvement local search algorithms. They are around 25% lower than for the algorithms using the 1-shift neighborhood, which is a remarkable improvement. On the other hand, this algorithm requires also the largest computational time. Considering the fact that the absolute computational times for all approaches are within some seconds, it is certainly worth to use the additional effort in terms of computational time to obtain such high quality solutions. This means, that we have identified the 2.5-opt first improvement local search algorithm for usage within the random restart local search algorithm.

6.4 A Random Restart Local Search Algorithm for the PTSPD

In this section we introduce a random restart local search algorithm for the PTSPD. This algorithm is based on the local search algorithms used in the previous section. We will give a detailed description of the algorithm and finish with an extensive computation study.

We have seen in section 6.2 that the 1-shift best improvement local search algorithm using the approximation of the objective function based on Monte Carlo sampling obtained solutions of poor quality in very few runs. Preliminary experiments have shown that this also happens for the first improvement local search algorithms using the 1-shift and the 2.5-opt neighborhoods. This is problematic for a heuristic that is meant to efficiently compute solutions of high quality in every run and therefore we propose the usage of the first improvement 2.5-opt local search algorithm within a random restart local search algorithm. The random restart local search algorithm (RRLS) iteratively performs a local search

starting from a random solution for a specific number of iterations. At the end the best local optimum found is returned.

Based on preliminary experiments with the 2.5-opt first improvement local search algorithm, we set the number of iterations to 3. Since obtaining a local optimal solution of poor quality occurred only in rare cases, in this way the probability to obtain such a bad solution in all of the iterations is negligible. The local optimal solutions obtained in the different iterations are then compared with the sampling-based approximation using 10^5 samples. The sampling-based approach with this large number of samples provides a sufficiently good approximation to determine the overall best solution, while only a fraction of the computational time of the exact evaluation is used. For the underlying local search algorithm we use the 2.5-opt first improvement local search algorithm using the approximation of the objective function based on Monte Carlo sampling. The number of samples is set to 500 like for the experiments in section 6.2. Note that we use the same benchmark instances as before.

6.4.1 Experimental Setup

For the experiments we use all the available benchmark instances. For each instance we perform 5 runs, resulting in 25 runs for each instance class. We then measure the average computational time as well as the average solution costs over all of the 25 runs per instance class. Again, all the experiments are performed on a Quad-Core AMD Opteron system running at 2GHz and the complete numerical results are available at the author's website (`www.idsia.ch/~weyland/`).

6.4.2 Results

The main goal of our computational studies is to provide numerical results on the common benchmark instances for the PTSPD which can be used for comparisons with more sophisticated methods in the future. Nonetheless, we will show that the random restart local search algorithm outperforms the heuristics used in Campbell and Thomas [2008b, 2009] as well as the heuristics used in the previous sections. The full numerical results of this computational studies are available at the author's website (`www.idsia.ch/~weyland/`).

First of all, in contrast to the local search algorithms on their own, the random restart local search algorithm did not compute solutions of poor quality with respect to the average solution quality in any of the 25 runs per instance

class. This means we can expect this approach to obtain solutions of good quality in each run, which is an important fact for an efficient heuristic. Secondly, as expected, the qualities of the final solutions are better for the random restart local search algorithm in comparison to the local search algorithms used in this chapter and in Campbell and Thomas [2009]. In table 6.5 the average decrease in solution costs for the RRLS with respect to the 1-shift best improvement local search algorithm using the approximations of Campbell and Thomas [2009] (denoted by TC, TA and EA) and the approximation introduced in this chapter (denoted by MCS) are shown. Again, we want to stress that those heuristics were not designed with the goal of efficiency, and therefore it is not surprising that the RRLS is multiple orders of magnitude faster than those approaches. Nonetheless, we use the comparison in table 6.5 to show that the usage of the RRLS results in obtaining better solutions compared to those non iterative methods. Additionally, we can see that also for instances with customer probabilities of 0.9 solutions of high quality are obtained, which was not the case for the local search algorithm using the sampling-based approximation of the objective function. In table 6.6 we have depicted the absolute computational times in seconds for the different classes of instances. The computational times for the new approaches depend mainly on the size of the instances and on the customer probabilities. For all the instances the computational times are in the order of some seconds, ranging from around 11 seconds for the instances of size 40 to around 300 seconds for the instances of size 100. Regarding the computational times for instance classes of different customer probabilities, the results confirm our experimental and theoretical results about the quasi-parallel evaluation of samples. Customer probabilities of 0.1 result in the lowest computational times, customer probabilities of 0.9 result in the highest computational times and the computational times for the other customer probabilities (range, mixed) are in between.

6.5 Discussion and Conclusions

In this chapter we introduced an approximation for the objective function of the Probabilistic Traveling Salesman Problem with Deadlines based on Monte Carlo sampling and using the novel approach of quasi-parallel evaluation of samples. With a theoretical analysis of the asymptotic computational time and additional computational studies we could show that the quasi-parallel evaluation of samples significantly improves the conventional sampling-based approximation of the PTSPD objective function. We then compared the new approximation of

	TC	TA	EA	MCS
all	-5.23%	-4.14%	-4.12%	-7.31%
size 40	-3.84%	-3.48%	-3.01%	-7.67%
size 60	-3.16%	-1.97%	-2.23%	-6.24%
size 100	-10.41%	-8.40%	-8.62%	-8.38%
prob. 0.1	-4.93%	-2.88%	-3.30%	-2.62%
prob. 0.9	-3.90%	-3.35%	-3.50%	-13.76%
prob. range	-6.48%	-5.64%	-4.90%	-5.58%
prob. mixed	-5.60%	-4.70%	-4.77%	-7.27%
pen. 5	-1.97%	-1.97%	-1.82%	-6.11%
pen. 50	-7.18%	-5.44%	-5.50%	-8.03%
deadl. early	-5.19%	-3.95%	-3.73%	-7.35%
deadl. late	-5.26%	-4.34%	-4.51%	-7.27%

Table 6.5. Average decrease in solution costs for the RRLS algorithm with respect to the 1-shift best improvement local search algorithms using the different approximations on different classes of instances.

instances	computational time
all	96.61
size 40	11.16
size 60	47.71
size 100	298.12
probability 0.1	16.73
probability 0.9	176.79
probability range	93.22
probability mixed	99.68
penalty 5	116.99
penalty 50	84.38
deadline early	97.30
deadline late	95.92

Table 6.6. Average absolute computational times in seconds for the RRLS algorithm on different classes of instances.

the objective function to other approximations by integrating them into a simple local search algorithm. The new approach was able to obtain solutions of better or at least competitive quality in most of the runs, whereas the computational time could be significantly reduced. In contrast to the other approximations, which still require the usage of the exact objective function to obtain solutions of good quality, our approach is completely independent of the exact objective function. In very few runs local optima of poor quality were obtained by our approach. Considering this fact and based on the experiences with the development of efficient heuristics for the PTSP, we then proposed the usage of a random restart local search algorithm for the PTSPD. At first, we identified the 2.5-opt first improvement local search algorithm as the strongest local search algorithm for the PTSPD. We then use this local search algorithm within a random restart local search algorithm. In fact, this is the first heuristic for the PTSPD that is completely independent of the exact objective function. An extensive computational study regarding this heuristic, including comparisons to other approaches, confirms the efficiency of this approach. Although a random restart local search algorithm combined with the sampling-based approximation of the objective function is currently the strongest approach for the PTSP, we do not know if a similar situation holds for the PTSPD. Nonetheless, our approach is the first algorithm meant to obtain solutions of good quality in a reasonable runtime and with our computational studies we provide the basis for future comparisons to more sophisticated methods. In fact, the absolute computational times for our approach are quite low in the context of a priori optimization. This results mainly from the efficient approximation of the objective function using the quasi-parallel evaluation of samples. Therefore, it seems promising to use more sophisticated heuristics based on our approximation to obtain better solutions for the PTSPD.

Chapter 7

Stochastic Vehicle Routing Problems and GPGPU

In this chapter we introduce a metaheuristic framework based on general-purpose computing on graphics processing units (GPGPU) for the optimization of stochastic combinatorial optimization problems. More in detail, we propose to use metaheuristics in combination with the *Monte Carlo sampling approximation* of the objective function. The actual evaluation of the solutions is then parallelized using the graphics processing unit (GPU) on the level of a single sample. We verify the efficiency of our approach with a case-study on the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. The underlying publication for this chapter is Weyland et al. [2012c].

The remaining part of this chapter is organized as follows. We first give an overview about applications of GPGPU for solving combinatorial optimization problems with metaheuristics. Then we discuss a general metaheuristic framework based on GPGPU for combinatorial optimization problems. After that we refine the metaheuristic framework and adapt it for stochastic combinatorial optimization problems. Finally, we apply the new approach to the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES and perform extensive computational studies. Here we investigate the computational time for the evaluation of solutions using the new approach. After that, we compare the efficiency of a metaheuristic using the new approach with state-of-the-art methods. We finish the chapter with discussions and conclusions.

7.1 Applications of GPGPU for Solving Combinatorial Optimization Problems with Metaheuristics

The application of GPGPU for solving combinatorial optimization problems with metaheuristics is an emerging and fast growing field. Since the evaluation of solutions is usually the most time-consuming task for heuristics, many of the approaches evaluate solutions in parallel on the GPU. Some approaches go further and outsource the computations for the most part or even completely on the GPU. In the following we give an overview about successful applications of metaheuristics using GPGPU. In Van Luong et al. [2009] a parallel local search algorithm using the GPU is introduced. Computational studies on the QUADRATIC ASSIGNMENT PROBLEM, the PERMUTED PERCEPTRON PROBLEM and the TRAVELING SALESMAN PROBLEM show that significant speed-ups are possible compared to a serial local search. Another work of the same authors focuses on the parallelization of large neighborhood local search algorithms for binary problems (Van Luong et al. [2010]). Czapinski and Barnes [2011] deals with a parallel tabu search algorithm. Here the authors show for the PERMUTATION FLOWSHOP PROBLEM that significant speed-ups can be achieved with the proposed approach. In Choong et al. [2010] a parallel variant of the simulated annealing algorithm for FPGA placement is presented. Compared to serial approaches an average speed-up of a factor of 10 could be obtained. A parallel ant colony optimization algorithm for the optimization of nonlinear functions is introduced in Zhu and Curry [2009]. This parallel approach is orders of magnitude faster than the same approach on the CPU. Similar results could be obtained for another ant colony optimization algorithm (Li et al. [2009]) and for several parallel evolutionary algorithms (Tsutsui and Fujimoto [2009]; Chitty [2007]; Harding and Banzhaf [2007]; Wong and Wong [2006]; Fok et al. [2007]). Recently, a differential evolution algorithm for the optimization of continuous optimization problems has been proposed that uses the GPU for the evaluation of the population. As in the other cases, major speed-ups can be obtained compared to the serial implementation.

7.2 A Metaheuristic Framework for Solving Stochastic Combinatorial Optimization Problems on the GPU

In this section we present a general metaheuristic framework for solving a large class of stochastic combinatorial optimization problems on GPUs. For a wide

variety of metaheuristics the computationally most expensive task is the evaluation of the objective function and constraints for the candidate solutions. For example, profiling results show that the state-of-the-art methods for the PTSPD (Weyland et al. [2011b, 2012b]) spend more than 90% of the computation time for those two tasks in many of the common benchmark instances. Since a lot of metaheuristics create in each iteration a set of different solutions (or can be adapted in a way that a set of different solutions is created), the evaluation of the objective function and constraints for the solutions can be performed in parallel on the GPU. This approach is especially very promising for metaheuristics creating a large number of solutions in each iteration and for combinatorial optimization problems where the computation of the objective function and of the constraints has an easy control flow without many branches and jumps. There exist many well known and established metaheuristics which can be used in this way. Among them are evolutionary algorithms (Goldberg [1989]; Beyer and Schwefel [2002]), where in each iteration all the new offspring solutions have to be evaluated, best improvement local search (Aarts and Lenstra [2003]) and tabu search (Glover and Laguna [1998]; Jin et al. [2012]), where in one iteration all solutions in a neighborhood of the current solution have to be evaluated, ant colony optimization (Dorigo et al. [1999]; Dorigo and Stützle [2003]), where in every iteration for each ant a solution is constructed that has to be evaluated and particle swarm optimization (Kennedy and Eberhart [1995]; Poli et al. [2007]), where in each iteration each particle creates a solutions which needs to be evaluated. Other metaheuristics can usually be adapted for this purpose. For example, it is possible for a first improvement local search algorithm (Aarts and Lenstra [2003]) or for a simulated annealing algorithm (Aarts and Korst [1990]; Laarhoven and Aarts [1987]) to create and evaluate in parallel a large number of solutions in the neighborhood of the current solution. Using the results from these evaluations the original metaheuristic can be simulated afterwards. Following this approach, in some cases solutions are created and evaluated in advance which would not have been created and evaluated in the original heuristic. But as long as this overhead is dominated by the computational speed-up due to the parallel evaluations, an overall speed-up is obtained. The metaheuristic framework for solving general combinatorial optimization problems on GPUs is depicted in algorithm 6. Here the tasks performed by the GPU are printed bold italic.

There are mainly two drawbacks of this approach. First, the number of solutions generated in each iteration has to be quite large to use the GPU efficiently. Second, the evaluation of the objective function and constraints does not allow for a straight control flow without branches and jumps in general. In the re-

Algorithm 6 Metaheuristic framework for combinatorial optimization problems

1: Initialize the metaheuristic (parameters, initial solutions, ...)
2: **while** The termination criterion is not fulfilled **do**
3: Create a preferably large number of solutions
4: ***Evaluate the objective function for the new solutions in parallel on the GPU***
5: ***Evaluate the constraints for the new solutions in parallel on the GPU***
6: Update the metaheuristic (solutions, parameters, ...)
7: **end while**
8: **return** The solution obtained by the metaheuristic

maining part of this section we show how to overcome these two drawbacks for a large class of stochastic combinatorial optimization problems. We focus on stochastic combinatorial optimization problems for which the objective function and the constraints can be approximately evaluated using Monte Carlo sampling. There is a huge variety of stochastic combinatorial optimization problems belonging to this class, ranging from chance constrained problems (Dror et al. [1993]) to a priori optimization problems (Bertsimas et al. [1990]). More in detail, the overall idea is to create a set of samples due to the given probability distributions. These samples are then used for an approximative evaluation of the objective function and the constraints. In this case it is even possible to parallelize the evaluations on the level of a single sample instead of a single solution. In other words, it is possible to parallelize the task of evaluating exactly one sample on one solution, which leads to a much better grade of parallelism. As a result, the number of solutions that have to be generated in each iteration of the metaheuristic does not need to be as large as in the general case to guarantee that the GPU is used efficiently. Another advantage is that in most cases the evaluation of exactly one sample on one solution has an easier control flow with less branches and jumps compared to the full evaluation of the objective function and the constraints.

All in all, it is possible to overcome both of the drawbacks mentioned before for this class of stochastic combinatorial optimization problems. Following this approach the metaheuristic needs to generate only a reasonable number of solutions and the evaluations of the objective function and the constraints on the sample level have a straight control flow. The resulting metaheuristic framework is shown in algorithm 7. Again, the tasks performed by the GPU are printed bold italic.

Algorithm 7 Metaheuristic framework for stochastic combinatorial optimization problems

1: Initialize the metaheuristic (parameters, initial solutions, ...)
2: Generate a set of samples due to the given probability distributions
3: **while** The termination criterion is not fulfilled **do**
4: Create new solutions
5: ***Approximately evaluate the objective function for the new solutions in parallel on the GPU (on the sample level)***
6: ***Approximately evaluate the constraints for the new solutions in parallel on the GPU (on the sample level)***
7: Update the metaheuristic (solutions, parameters, ...)
8: **end while**
9: **return** The solution obtained by the metaheuristic

The framework is very general and applies to the class of problems under investigation. In sections 7.3 and 7.4 we show in detail how the framework can be used for the optimization of the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. The results of this case study demonstrate the huge potential of the proposed framework.

7.3 Solution Evaluation for the PTSPD on the GPU

Before we use the metaheuristic framework discussed in section 7.2 to tackle the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES, we perform experiments to determine the number of solutions and samples that have to be used to utilize the GPU in an efficient way. Although it is quite easy to theoretically compute a lower bound for the number of solutions and samples, the optimal number for these parameters also depends on factors like memory latencies, cache misses and the number of registers used. Those factors highly depend on the actual GPU that is used and it is very hard to incorporate those factors into a model to theoretically compute the optimal number of solutions and samples that have to be evaluated in parallel. Therefore, we perform experiments with different numbers of solutions to empirically determine those parameters for later usage within the metaheuristic framework.

As we have seen in chapter 6, the most efficient heuristics for the PTSPD are based on a local search algorithm using Monte Carlo sampling combined with

statistical tests to evaluate solutions. For being able to use statistical tests efficiently it is very important that only a small number of samples is used for the evaluation of a solution in each iteration. This allows to perform statistical tests in regular intervals to sort out solution that are worse than the current solution or to accept an improving solution as soon as sufficient statistical evidence is available. Therefore, we evaluate solutions always in packages of 32 samples in each iteration. There are two different reasons to use this number. First, it has been shown that performing statistical tests in regular intervals each time after between 20 and 50 additional samples are used for the evaluation of a solution leads to a very efficient usage of the statistical tests (Balaprakash et al. [2009a]). Second, this number corresponds exactly to the number of different threads that can be executed at the same time on one of the GPU processors. Note that this number is the same for all GPUs that are compatible with the CUDA API (`www.nvidia.com/object/cuda_home_new.html`). This means that we have to determine with our experiments the number of solutions which should be evaluated in parallel on 32 samples each.

7.3.1 Experimental Setup

For our experiments we iteratively evaluate a number of solutions on (not necessarily identical) packages of 32 samples and measure the average number of samples that can be evaluated in one second. We have implemented this evaluation mechanism in C using common compiler optimizations for the experiments on the CPU. Here the solutions are evaluated one after another on the same samples. Additionally, we have implemented the quasi-parallel evaluation of samples which has been introduced in chapter 6. This method is currently the state-of-the-art method to evaluate samples for the PTSPD. With this method samples can be evaluated much faster on instances with low customer probabilities. Note that the term quasi-parallel is somehow misleading in the context of this chapter. Instead of computing the costs for the different samples after each other, for the quasi-parallel evaluation the customers are processed in the order given by the solution and for each customer the costs of the samples in which this customer requires a visit are updated. This means that the samples are not evaluated strictly after each other, but this approach is not a parallel one and the computations are performed serially on a CPU. Moreover, we have implemented the standard evaluation mechanism in C using CUDA and the same compiler optimizations as before for the experiments on the GPU. Here the number of solutions that are evaluated in parallel is the parameter we want to optimize. More in detail, we want to find the number of solutions that have to be evalu-

ated in parallel on the GPU, such that the ratio between the average number of solutions evaluated per second on the GPU and the average number of solutions evaluated per second on the CPU is maximized. For the number of solutions that are evaluated in parallel, we have selected values between 256 and 16384 in steps of 256.

Since the computational effort to evaluate solutions does not depend significantly on the actual solution we use randomly generated solutions for those experiments. The computational effort also does not depend significantly on the customer locations, the customer deadlines and the penalty values. Therefore we use randomly generated instances of sizes 40, 60, 100, 150 and 200. For the customers' probabilities we use values of 0.1 for all customers, 0.9 for all customers, values that are chosen uniformly at random from the set $\{0.1, 1.0\}$ (referred to as mixed probabilities) or values chosen uniformly at random in $[0, 1]$ (referred to as range probabilities). That means we have in total 20 different classes of instances for these experiments. We have chosen these instance sizes and probabilities, since they are also used in the benchmark instances for the PTSPD and in the experiments in section 7.4. These benchmark instances were already discussed in detail in chapter 6 and for further details we refer to this chapter. For each instance class we evaluate the solutions iteratively on a Quad-Core AMD Opteron system running at 2GHz (for the serial and quasi-parallel approaches) and on a system with a GeForce GTX 580 graphics card, which has 16 streaming multiprocessors, each with 32 CUDA cores (for the parallel approach). The total computational time for each experiment was limited to 60 seconds. At the end we computed the average number of solutions that were evaluated in one second.

7.3.2 Results

Figure 7.1 contains a graph showing the number of evaluations that are performed per second (in millions) for the three different approaches on instances of size 100 with customer probabilities of 0.1. For the parallel evaluation on the GPU we have fitted a curve through the available data points. Although the quasi-parallel evaluation is more powerful than the serial evaluation, those differences vanish if they are compared to the parallel approach. The curve representing the parallel approach has a very steep trajectory in the area around 2000 solution evaluations and slowly approaches a limit for larger numbers of solution evaluations. The graphs for instances of different sizes and with different probability types are similar to those presented in figure 7.1. They are available at the author's homepage (`www.idsia.ch/~weyland/`).

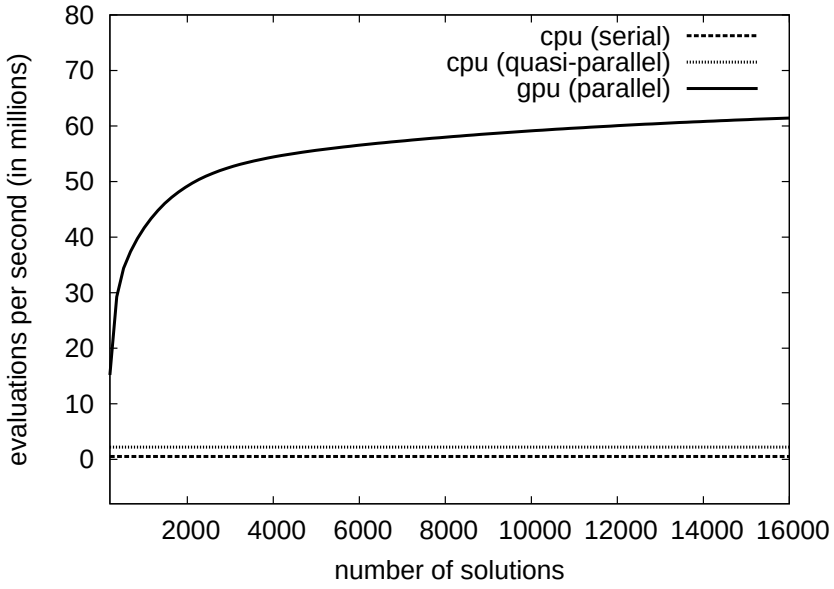


Figure 7.1. The graphs show the average number of solution evaluations (in millions) per second for the serial approach, the quasi-parallel approach and the parallel approach on instances of size 100 with customer probabilities of 0.1.

In table 7.1 the average numbers of solution evaluations per second (in millions) for the serial approach, the quasi-parallel approach, the parallel approach evaluating 512 solutions in parallel and the parallel approach evaluating 2048 solutions in parallel on the 20 different instance classes are shown. These results are additionally visualized in figure 7.2 for the four different instance classes of size 100. Significant speed-ups are obtained using the parallel approach instead of the serial or the quasi-serial approach. In table 7.2 the speed-ups obtained by the parallel approach evaluating 2048 solutions in parallel with respect to the serial and the quasi-serial approach is shown. The parallel approach can evaluate between 60 and 160 times more solutions than the serial approach on those instance classes. Similar speed-ups are obtained for the parallel approach in comparison with the quasi-parallel approach, except for the instance classes using probabilities of 0.1. But even for those instance classes the parallel approach can evaluate between 10 and 30 times more solutions than the quasi-parallel approach in the same time. The speed-up is even better if more solutions are evaluated in parallel, but for some metaheuristics it could be difficult to provide such large numbers of solutions in each iteration. Therefore, we omitted this option in our experiments to obtain fair and realistic results.

All in all, the parallel evaluation of samples is a very promising alternative to the serial and quasi-parallel evaluation for the PTSPD. Using a heuristic for the PTSPD which can generate between 200 and 4000 solutions in each iteration, the computational time for the solution evaluation can be decreased by a factor of between 15 and 175, depending on the instance size and the customer probabilities.

7.4 Heuristics for the PTSPD on the GPU

In this section we show how the metaheuristic framework of section 7.2 can be used to speed up the state-of-the-art heuristics for the PTSPD. The current best heuristics for the PTSPD were discussed in chapter 6. Those heuristics are random restart local search algorithms using a first improvement local search algorithm with a sampling-based approximation of the objective function. Extensive profiling revealed that a huge amount of the computational time for those approaches is used for the evaluation of solutions. Therefore, it is very promising to use the parallel evaluation of solutions on the sample level to speed up the computations. Nonetheless, the algorithms have to be adapted for this purpose. We start this section with a detailed description of the parallel random restart local search algorithm. After that we describe the experimental setup and present the results.

instance size	probability type	serial evaluation	quasi-parallel evaluation	parallel (512)	parallel (2048)
40	0.1	1.325	5.488	46.255	79.246
	0.9	1.088	1.054	45.184	77.444
	ranged	1.021	1.624	45.385	77.752
	mixed	1.107	1.734	45.923	78.525
60	0.1	0.881	3.712	41.715	66.565
	0.9	0.751	0.701	40.303	64.101
	ranged	0.678	1.185	40.270	63.583
	mixed	0.739	1.060	41.029	65.497
100	0.1	0.514	2.190	34.254	49.925
	0.9	0.413	0.382	32.668	47.468
	ranged	0.382	0.679	32.489	46.825
	mixed	0.453	0.629	33.223	48.615
150	0.1	0.337	1.404	27.764	37.548
	0.9	0.269	0.232	26.359	35.559
	ranged	0.240	0.433	26.304	35.022
	mixed	0.261	0.391	26.903	36.763
200	0.1	0.243	1.036	23.448	30.002
	0.9	0.180	0.162	21.997	28.246
	ranged	0.181	0.274	21.925	27.805
	mixed	0.184	0.279	22.646	29.301

Table 7.1. This table contains the average number of solution evaluations (in millions) per second on the 20 instance classes for the serial approach, the quasi-parallel approach, the parallel approach evaluating 512 solutions in parallel and the parallel approach evaluating 2048 solutions in parallel.

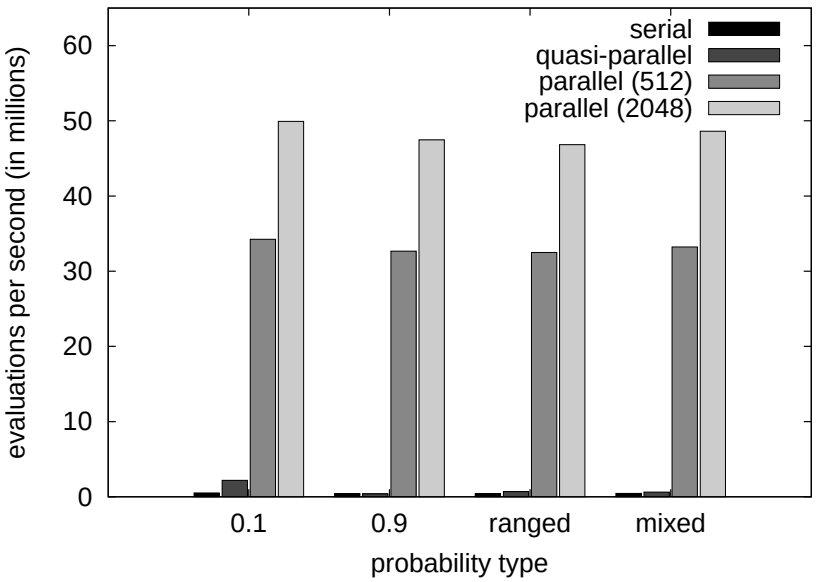


Figure 7.2. This figure shows the average number of solution evaluations (in millions) per second on instances of size 100 for the serial approach, the quasi-parallel approach, the parallel approach evaluating 512 solutions in parallel and the parallel approach evaluating 2048 solutions in parallel.

instance size	probability type	parallel / serial	parallel / quasi-parallel
40	0.1	59.81	14.44
	0.9	71.18	73.48
	ranged	76.15	47.88
	mixed	70.94	45.29
60	0.1	75.56	17.93
	0.9	85.35	91.44
	ranged	93.78	53.66
	mixed	88.63	61.79
100	0.1	97.13	22.80
	0.9	114.93	124.26
	ranged	122.58	68.96
	mixed	107.32	77.29
150	0.1	111.42	26.74
	0.9	132.19	153.27
	ranged	145.93	80.88
	mixed	140.85	94.02
200	0.1	123.47	28.96
	0.9	156.92	174.36
	ranged	153.62	101.48
	mixed	159.24	105.02

Table 7.2. This table contains the average speed-up for the evaluation of solutions on the 20 instance classes for the parallel approach evaluating 2048 solutions in parallel with respect to the serial and the quasi-serial approach.

7.4.1 The Parallel Random Restart Local Search Algorithm

In a random restart local search algorithm several local search runs starting from randomly generated solutions are performed after another. At the end the best local optimum is returned as the final solution. For our parallel implementation we initialize a certain number of local search algorithms at the beginning. Candidate solutions of all the different local search runs are then evaluated in parallel using the GPU. In this way it is easy to obtain between 200 and 4000 solutions in each iteration to use the GPU in an efficient way. For the underlying local search algorithm we use the same as for the heuristics used in chapter 6, namely a 2.5-opt first improvement local search using statistical tests. In each iteration the local search algorithm explores the neighborhood of the current solution in a random order. Each solution is evaluated in small packages of 32 samples. After each package of samples is evaluated a statistical test is performed to check whether there is enough statistical evidence that this solution is better/worse than the current solution. In case the solution is better, the current solution is replaced with this solution and a new iteration is started. In case the solution is worse, the algorithm continues with the next solution in the neighborhood. If there is no statistical evidence for the solution to be better/worse than the current solution and a maximum number of samples has been evaluated, the mean of the sample costs is used for the decision making. To be able to utilize the GPU efficiently, we have to adapt this local search algorithm slightly for our parallel approach. Since we want to benefit from the statistical tests, we evaluate multiple solutions for each local search run on one package of samples in parallel. Then we check for all those solutions, if they are improving over the current solution, if they are worse than the current solution or if they should be evaluated on more samples. Before we start to discuss some implementation issues, we would like to mention that the parallel adaption of the first improvement local search differs slightly from the local search it is based on. First of all the local search is not a strict first improvement local search anymore, since we evaluate multiple solutions in parallel and use all the available information immediately for decision making. Nonetheless, we expect both algorithms to perform in a similar way as long as the number of solutions that are evaluated in parallel is not too large compared to the size of the local search neighborhood. The second difference is that in case an improving solution has been found some other results that were already obtained in parallel have been computed for nothing. But since the ratio between the number of improving solutions found during a local search and the number of solutions that are evaluated during a local search is usually very small, this factor should be negligible.

In algorithm 8 a high level overview of the proposed algorithm is shown. The GPU related parts are printed bold italic. At the beginning the algorithm reads the instance file and initializes all the necessary data structures for the CPU and for the GPU. Then n local search algorithms are created and initialized. As long as there are active local search algorithms, the local search algorithms are requested to fill the evaluation slots. For this purpose a solution and the indices of the samples which should be evaluated on this solution are provided. If all evaluation slots are filled, the data is transferred to the GPU, which evaluates the solutions on the corresponding samples in parallel. After the evaluation of the solutions, the result data is transferred back from the GPU and the local search algorithms process the results. At the end of the main loop all local search algorithms which have already obtained a local optimum are indicated as inactive and removed from the optimization process. When all local search algorithms have found a local optimum, the algorithm returns the best local optimum found as the final solution.

Algorithm 8 High level description of the proposed parallel random restart local search algorithm on the GPU

```
1: Read instance file and initialize data structures
2: ***Initialize the GPU related data structures***
3: Initialize n local search algorithms and store them in the set L
4: while L is not empty do
5:    while there are free slots for the evaluation of solutions do
6:       Select an algorithm from L
7:       Request the algorithm to provide a solution for evaluation
8:    end while
9:    ***Transfer the relevant data to the GPU***
10:   ***Evaluate the solutions in parallel on the GPU***
11:   ***Transfer the result data from the GPU***
12:   for each algorithm in L do
13:      Process the results obtained from the GPU
14:   end for
15:   Remove algorithms which have obtained a local optimum from L
16: end while
17: return The best local optimum
```

There are three important tasks for the local search algorithms: initializing, providing solutions for the evaluation and processing the results from the evaluation. In the following we discuss these tasks more in detail.

In algorithm 9 the initialization process is depicted. At first m samples are created according to the probability distributions given in the input instance. Here m is the maximum number of samples that are used for the evaluation of one solution. Then an initial solution is created uniformly at random. After that two data structures are created, a stack and a queue. These data structures play an important role in providing solutions for the evaluation and in processing the results obtained from the GPU. The queue is initially empty and the stack is filled with all the moves that lead to solutions in the neighborhood of the initial solution. The order in which these moves are stored on the stack is according to a uniform distribution. Additionally, for each of these moves the number of samples which have already been used for the evaluation of this move is stored. This number is initially set to 0.

Algorithm 10 shows how the local search algorithms provide the solutions for the evaluation on the GPU. As long as the stack is not empty the solution on top of the stack is removed from the stack and returned by this procedure. In case this solution could be evaluated on other samples afterwards, it is additionally stored at the end of the queue. In case the stack is already empty, but the queue is not empty, the solution at the beginning of the queue is removed from the queue and returned. Again, if this solution could be evaluated on other samples afterwards, it is additionally reinserted at the end of the queue. In the rare case where the stack and the queue are both empty, a dummy solution is returned. The case in which both, the stack and the queue, are empty corresponds to the rare case in which no more evaluations are required for the current neighborhood of the local search algorithm. If the stack is empty, but there are still moves stored in the queue, it means that there are more evaluation slots available for the local search algorithms than the number of solutions in the current neighborhood which have not been fully evaluated yet. In that case the solutions are evaluated on more than one package of samples. The idea behind the stack is that solutions which have already been evaluated on some packages of samples are used for further evaluation with a higher priority than solutions which have not been evaluated at all. Therefore, solutions that have already been partially evaluated should always be on top of the stack.

This property is maintained in the last important task of the local search algorithm, namely the processing of the results obtained by the GPU. This task is described in algorithm 11. At the beginning of this procedure the queue is emptied, since it is only used for providing solutions. Then the solutions are

processed in the same order in which they were provided for the evaluation. In case a solution has been evaluated on the maximum number of available samples, the average sample costs are compared to the average sample costs of the current solution. If the new solution is better it replaces the current solution, the stack is reinitialized with the moves according to the neighborhood of the new solution and the procedure terminates. Otherwise, the new solution has been completely evaluated and is worse than the current solutions. Therefore, it is not considered anymore. In case the solution has not been evaluated on the maximum number of available samples, a statistical test is performed. If there is sufficient statistical evidence for the fact that the new solution is better than the current solution, the current solution is replaced by the new one, the stack is reinitialized with the moves according to the neighborhood of the new solution and the procedure terminates. If there is sufficient statistical evidence for the fact that the new solution is worse than the current solution, it is not considered anymore. If both statistical tests are negative the solution is reinserted on top of the stack with the corresponding number of samples already used for the evaluation of this solution. In this way solutions that have been evaluated partially are on top of the stack and therefore those solutions have a higher priority to be evaluated in the next iteration.

All in all, we use several mechanisms to create an efficient adaptation of a first improvement local search algorithm. Improving solutions are accepted as soon as there is sufficient statistical evidence. The stack guarantees that solutions that have already been evaluated on some samples have a higher priority in the following iterations than solutions which have not been evaluated at all. Additionally, the queue is used to occupy the available evaluation slots for the local search algorithm in an efficient way.

Algorithm 9 Local Search: Initialization

1: Create m samples according to the probability distributions given in the input instance
2: Create an initial solution x uniformly at random
3: Create a stack S
4: Create a queue Q
5: Fill the stack S with all moves that lead to solutions in the neighborhood of x in an order uniformly at random

Algorithm 10 Local Search: Provide Solution

```
1: if the stack S is not empty then
2:    Obtain the solution x' on top of S and remove it from S
3:    if x' could be evaluated on other samples afterwards then
4:       Put x' at the end of the queue
5:    end if
6:    return x' and the indices of the corresponding samples
7: else if the queue Q is not empty then
8:    Obtain the solution x' from the beginning of Q and remove it from Q
9:    if x' could be evaluated on other samples afterwards then
10:      Put x' at the end of the queue
11:   end if
12:   return x' and the indices of the corresponding samples
13: else
14:   return a dummy solution
15: end if
```

7.4.2 Experimental Setup

For the experiments we implemented the parallel random restart local search algorithm described above in C using common compiler optimizations. For the number of parallel local searches that are used within the proposed approach we use values of 3, 10 and 30. For the maximum number of samples we use a value of 1024. The samples are evaluated in packages of 32 samples, as described in section 7.3. We use the same statistical test as in the serial version of this algorithm (Weyland et al. [2011b]), namely a student t-test (Sheskin [2004]) with a significance level of 98%. In this way we are able to perform a fair comparison between the parallel and the serial version of the algorithm. For most of the experiments we setup the GPU to evaluate 512 solutions in parallel. For larger instances, especially in combination with the variant using 30 local searches, we setup the GPU to evaluate up to 16384 solutions in parallel. The exact number of parallel solution evaluations for the different experiments is available at the author's website (`www.idsia.ch/~weyland/`). For the benchmark instances we use the same instances as in chapter 6. We additionally extended the benchmark set for the PTSPD with larger instances. These new instances are constructed in the same way as the others (Campbell and Thomas [2008b]). In total we have

Algorithm 11 Local Search: Process Results

```
1: Empty the queue Q
2: for each solution x' provided for evaluation (in the order they were provided) do
3:   if the solution has been evaluated on the maximum number of samples then
4:     if x' is better than x then
5:       x := x'
6:       Fill the stack S with all moves that lead to solutions in the neighborhood of x in an order uniformly at random
7:       return
8:     end if
9:   else
10:    if x' is statistically better than x then
11:      x := x'
12:      Fill the stack S with all moves that lead to solutions in the neighborhood of x in an order uniformly at random
13:      return
14:    else if x' is statistically worse than x then
15:      Do not consider x' as a possible improving solution for x anymore
16:    else
17:      Put the move leading to x' on the stack and update the corresponding number of already evaluated samples for this solution
18:    end if
19:  end if
20: end for
```

80 different instance classes (5 different instance sizes, 4 probability types, 2 penalty values, 2 deadline types) consisting of 5 instances each. For our experiments we have performed 20 runs on each of the 5 instances for every instance class, resulting in 100 runs per instance class. We finally measured the average computational time and the average solution costs for all the 100 runs. Due to higher computational times for larger instances, we restricted the experiments for the instances of size 150 to 25 runs per instance class and for the instances of size 200 to 10 runs per instance class. All the experiments in this section were performed on a system with a GeForce GTX 580 graphics card.

7.4.3 Results

In table 7.3 and in figure 7.3 we have shown the performance of the new algorithms relative to that of the 2.5-opt random restart local search algorithm using statistical tests and the sampling-based evaluation of solutions (Weyland et al. [2011b]). This algorithm is evaluating the solutions in packages of 25 samples with a maximum number of 1000 samples and performs 3 iterations in total. The statistical test is the same which is used for our experiments. Note that this algorithm is currently the state-of-the-art method for the PTSPD. In the table the average relative computational time (average over $\text{time}_{\text{parallel}}$ / $\text{time}_{\text{serial}}$) and the average relative solution cost (average over $\text{cost}_{\text{parallel}}$ / $\text{cost}_{\text{serial}}$) over all instances and over some classes of instances characterized by different parameters is shown. A value of 1.0 indicates that the algorithms perform in average in the same way. Smaller values for the computational time indicate that the new approach is faster in average, larger values for the computational time indicate that the new approach is slower in average. For the solution costs, smaller values indicate that the new approach finds better solutions in average, whereas larger values indicate that the new approach finds worse solutions in average. With RRLS_3, RRLS_{10} and RRLS_{30} we depict the new algorithm running 3, 10 and 30 local searches in parallel.

What we can see in table 7.3 and in figure 7.3 is the following. The parallel random restart local search requires in average 24.6% of the computational time required by the state-of-the-art algorithm using the same number of local searches, while the quality of the solutions obtained is the same. The other parallel approaches with 10 and 30 local searches require 58.5% and 151.8% of the computational time of the serial random restart local search. Here the solutions are in average 1% and 1.4% better. Note that there is a sublinear increase in computational time with respect to the number of local searches for most of the benchmark instances. This is mainly due to the fact that the graphics card can be occupied more efficiently if a larger number of local searches is used. Therefore, the computational time of the approach with 30 local searches is in average only a factor of about 7 larger than the computational time for the approach with 3 loacl searches, whereas the expected difference in computational time is a factor of 10. Although the average results are quite good, we can see that the performance of the parallel approach is not that good for the very small instances of size 40, for the large instances of size 200 and for instances with customer probabilities of 0.1. For instances with low customer probabilities the quasi-parallel evaluation of samples leads to a huge speed-up for the serial approach. This explains the observations for the instances with customer probabilities of 0.1 and

will become more clear later in this section when we compare absolute computational times. The explanations for the behavior of the instances with sizes 40 and 200 can be given based on extensive profiling. In both cases the fraction of the computational time spend for the evaluation of solutions is much smaller than for the other instance classes and therefore the overall performance is not affected that much by the parallel evaluation of the solutions. For the instances of size 40 this originates from the algorithmic overhead and from the fact that the neighborhood of the current solution is rather small. For the instances of size 200 it is caused mainly by the creation of neighbor solutions which require much more computational time for those instances. In fact, the computational time for creating solutions in the 2-opt neighborhood grows quadratically with the size of the instance. On the other hand, the performance of the new approach is much better on the other instance classes. For example, the parallel random restart local search with 3 local searches is in average more than a factor of 12 faster on instances of size 150 compared to the serial approach using the same number of local searches. As long as instances with customer probabilities of 0.1 are not considered, the parallel random restart local search with 30 local searches is in average even faster than the serial approach with 3 local searches.

Tables 7.4 to 7.6 show the absolute average computational times and the absolute average solution costs obtained by the new parallel random restart local search with 3, 10 and 30 local searches and the serial random restart local search for the instances of size 100, 150 and 200. The serial random restart local search algorithm is denoted with RRLS. For the parallel approaches we use the same notation as in table 7.3. Similar tables for instances of 40 and 60 are available at the author's website (`www.idsia.ch/~weyland/`). We can see that compared to the algorithm RRLS the new parallel approach RRLS_3 achieves a speed-up by a factor of around 10 in most of the cases while obtaining solutions of the same quality. Note that the algorithmic overhead, which is still computed on the CPU, is the same for the serial and the parallel approach. Therefore, we cannot expect speed-up factors as huge as in section 7.3 for the pure evaluation of solutions. Since the algorithmic overhead is responsible for between 2% and 10% of the total computational time, the speed-up values depicted in tables 7.4 to 7.6 are quite close to the theoretical upper bound. Furthermore, we can see that additional iterations of the parallel random restart local search algorithm (RRLS_{10} and RRLS_{30}) can be used to obtain better solutions while the absolute computational times are still in the range of the serial approach with 3 iterations (except for instances of size 200 and with customer probabilities of 0.1).

instances	parallel $RRLS_3$ rel. costs	/ time	parallel $RRLS_{10}$ rel. costs	/ time	parallel $RRLS_{30}$ rel. costs	/ time
all	1.004	**0.246**	**0.991**	**0.585**	**0.986**	1.518
size 40	1.007	**0.514**	**0.997**	**0.898**	**0.997**	2.138
size 60	1.000	**0.286**	**0.996**	**0.640**	**0.994**	1.537
size 100	1.001	**0.128**	**0.985**	**0.261**	**0.975**	**0.651**
size 150	1.001	**0.072**	**0.990**	**0.194**	**0.983**	**0.487**
size 200	1.012	**0.230**	**0.989**	**0.930**	**0.979**	2.775
prob. 0.1	**0.999**	**0.667**	**0.997**	1.590	**0.996**	4.089
prob. 0.9	1.006	**0.051**	**0.985**	**0.149**	**0.979**	**0.411**
prob. ranged	1.001	**0.126**	**0.991**	**0.291**	**0.984**	**0.769**
prob. mixed	1.009	**0.139**	**0.993**	**0.308**	**0.985**	**0.801**
deadl. early	1.002	**0.238**	**0.991**	**0.586**	**0.987**	1.512
deadl. late	1.007	**0.254**	**0.992**	**0.583**	**0.984**	1.523
penalty 5	1.002	**0.227**	**0.992**	**0.556**	**0.985**	1.448
penalty 50	1.006	**0.265**	**0.990**	**0.613**	**0.986**	1.587

Table 7.3. The average performance of the new approach relative to the 2.5-opt random restart local search algorithm using statistical tests and the sampling-based evaluation of solutions. Improvements are highlighted in boldface.

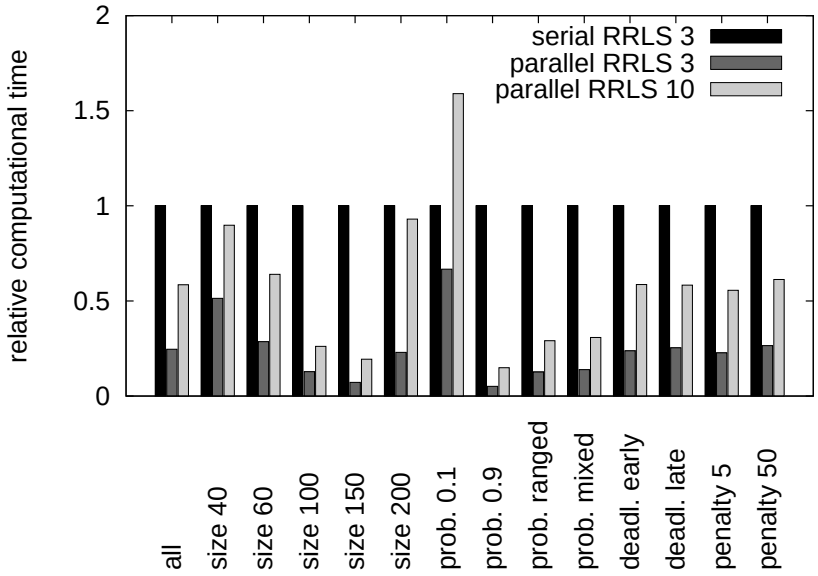


Figure 7.3. The average computational time of the new approach relative to the 2.5-opt random restart local search algorithm using statistical tests and the sampling-based evaluation of solutions.

prob.	deadl.	pen.	RRLS	par. $RRLS_3$	par. $RRLS_{10}$	par. $RRLS_{30}$
ranged	early	5	418.51	420.06	413.60	**408.39**
			(33.36)	(**1.94**)	(4.45)	(11.59)
ranged	early	50	1021.95	1010.87	1006.43	**1001.58**
			(34.86)	(**2.28**)	(5.29)	(14.15)
ranged	late	5	331.14	330.59	323.45	**315.47**
			(31.92)	(**1.96**)	(4.41)	(11.46)
ranged	late	50	339.12	338.19	329.84	**321.16**
			(34.36)	(**2.30**)	(5.19)	(13.30)
0.1	early	5	165.19	165.08	164.89	**164.79**
			(21.43)	(**6.41**)	(14.02)	(35.52)
0.1	early	50	239.39	239.30	238.49	**238.17**
			(45.69)	(**21.44**)	(31.82)	(75.35)
0.1	late	5	153.53	153.39	153.05	**152.81**
			(19.16)	(**5.66**)	(13.23)	(32.35)
0.1	late	50	157.37	157.10	156.60	**156.25**
			(31.67)	(**12.18**)	(21.84)	(56.66)
0.9	early	5	779.68	778.72	756.83	**749.76**
			(51.87)	(**1.71**)	(3.89)	(10.33)
0.9	early	50	3518.86	3524.21	3387.35	**3380.97**
			(54.45)	(**1.75**)	(4.15)	(11.21)
0.9	late	5	426.65	427.07	423.09	**419.63**
			(48.27)	(**1.64**)	(3.73)	(9.98)
0.9	late	50	429.40	426.91	423.68	**421.25**
			(53.01)	(**1.67**)	(6.80)	(10.53)
mixed	early	5	433.93	435.30	423.91	**419.28**
			(39.29)	(**2.06**)	(4.71)	(12.31)
mixed	early	50	1035.91	1048.63	1023.94	**1018.01**
			(41.89)	(**2.47**)	(5.59)	(14.53)
mixed	late	5	339.42	342.16	334.21	**327.67**
			(37.20)	(**1.99**)	(4.50)	(11.73)
mixed	late	50	341.86	344.24	337.00	**331.56**
			(40.52)	(**2.41**)	(5.34)	(13.73)

Table 7.4. Absolute solution costs and absolute computational times (in seconds) on instances of size 100. The values without brackets are the solution costs and the values within the brackets are the computational times. Best values for each instance are highlighted in boldface.

prob.	deadl.	pen.	RRLS	par. $RRLS_3$	par. $RRLS_{10}$	par. $RRLS_{30}$
ranged	early	5	448.19	447.88	445.11	**441.07**
			(142.45)	(**5.78**)	(14.31)	(37.16)
ranged	early	50	1082.84	1082.94	1079.72	**1076.54**
			(156.51)	(**6.84**)	(17.09)	(44.87)
ranged	late	5	369.18	371.28	365.32	**361.99**
			(147.20)	(**5.51**)	(14.07)	(35.88)
ranged	late	50	372.97	368.63	367.59	**362.57**
			(157.11)	(**6.35**)	(16.40)	(41.74)
0.1	early	5	186.52	186.40	186.22	**186.17**
			(80.32)	(**14.20**)	(40.96)	(86.69)
0.1	early	50	232.81	232.49	231.61	**231.20**
			(134.61)	(**25.91**)	(75.87)	(216.06)
0.1	late	5	178.48	178.28	178.14	**178.05**
			(77.01)	(**13.67**)	(37.36)	(82.91)
0.1	late	50	181.28	180.68	180.33	**180.05**
			(102.83)	(**22.27**)	(52.37)	(147.37)
0.9	early	5	792.12	798.92	782.94	**778.61**
			(236.33)	(**5.03**)	(12.57)	(34.62)
0.9	early	50	3357.15	3348.59	3339.74	**3337.40**
			(237.82)	(**4.89**)	(12.93)	(36.22)
0.9	late	5	488.04	487.70	477.46	**472.43**
			(233.24)	(**4.72**)	(12.05)	(33.68)
0.9	late	50	486.21	489.68	478.63	**474.67**
			(230.72)	(**5.01**)	(24.83)	(34.40)
mixed	early	5	503.41	503.60	494.06	**487.18**
			(172.80)	(**5.71**)	(14.61)	(38.65)
mixed	early	50	1375.04	1370.02	1363.91	**1360.17**
			(178.85)	(**6.58**)	(17.34)	(46.22)
mixed	late	5	389.60	393.62	385.95	**379.04**
			(162.08)	(**5.46**)	(14.06)	(36.55)
mixed	late	50	393.04	395.29	387.30	**381.84**
			(177.43)	(**6.49**)	(16.53)	(42.71)

Table 7.5. Absolute solution costs and absolute computational times (in seconds) on instances of size 150. The values without brackets are the solution costs and the values within the brackets are the computational times. Best values for each instance are highlighted in boldface.

prob.	deadl.	pen.	RRLS	par. $RRLS_3$	par. $RRLS_{10}$	par. $RRLS_{30}$
ranged	early	5	518.41	524.20	513.00	**510.03**
			(412.56)	(**39.37**)	(122.93)	(361.43)
ranged	early	50	1117.37	1120.26	1113.17	**1104.71**
			(407.26)	(**48.27**)	(140.90)	(423.59)
ranged	late	5	438.22	437.85	431.56	**425.50**
			(390.68)	(**38.00**)	(121.77)	(346.74)
ranged	late	50	443.19	450.85	436.15	**434.00**
			(402.65)	(**50.16**)	(151.43)	(444.49)
0.1	early	5	212.29	212.85	212.42	**211.87**
			(206.49)	(**102.43**)	(470.96)	(1486.77)
0.1	early	50	264.65	264.32	263.31	**262.58**
			(310.64)	(**193.35**)	(1053.84)	(2913.29)
0.1	late	5	202.63	203.84	202.60	**202.35**
			(180.09)	(**102.95**)	(455.20)	(1433.29)
0.1	late	50	205.20	206.24	206.10	**204.77**
			(262.98)	(**262.12**)	(923.43)	(2764.23)
0.9	early	5	900.94	916.64	883.88	**863.71**
			(668.04)	(**23.63**)	(88.38)	(251.05)
0.9	early	50	3711.44	3666.13	3481.28	**3468.07**
			(656.23)	(**26.81**)	(97.61)	(272.10)
0.9	late	5	565.52	565.09	555.35	**542.73**
			(620.73)	(**24.14**)	(89.25)	(249.45)
0.9	late	50	566.52	567.35	553.41	**547.83**
			(616.40)	(**26.21**)	(88.94)	(271.95)
mixed	early	5	526.45	529.32	524.72	**513.28**
			(469.58)	(**40.92**)	(137.55)	(392.51)
mixed	early	50	1127.78	1158.12	1116.48	**1113.12**
			(487.55)	(**53.04**)	(158.28)	(485.75)
mixed	late	5	442.94	452.21	447.83	**437.67**
			(463.24)	(**38.80**)	(133.08)	(385.47)
mixed	late	50	447.93	486.55	444.09	**442.18**
			(467.51)	(**53.02**)	(166.05)	(537.57)

Table 7.6. Absolute solution costs and absolute computational times (in seconds) on instances of size 200. The values without brackets are the solution costs and the values within the brackets are the computational times. Best values for each instance are highlighted in boldface.

7.5 Discussion and Conclusions

In this chapter we have introduced a general metaheuristic framework for solving stochastic combinatorial optimization problems based on general-purpose computing on graphics processing units. This framework can be applied to all stochastic combinatorial optimization problems for which the objective function and the constraints can be efficiently approximated with Monte Carlo sampling. In this way the evaluation of the objective function and the constraints can be parallelized on a sample level, which allows an efficient utilization of the GPU. Many existing metaheuristics can be used easily within the proposed framework to benefit from the computational power provided by modern GPUs.

As a case study we applied the metaheuristic framework to a random restart local search algorithm for the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. The resulting parallel heuristic leads to significant improvements over state-of-the-art methods for the PTSPD. For most of the instances used in this chapter the computational time could be reduced by more than an order of magnitude. Additionally, we showed that the savings in computational time could be used to obtain solutions with a better quality. Compared to the best heuristics for the PTSPD using the exact objective function or analytical approximations, the computational time could be reduced even by several orders of magnitude. This reveals once again the power of sampling-based approaches for the optimization of stochastic combinatorial optimization problems.

With the new framework sampling-based heuristics for stochastic combinatorial optimization problems become even more powerful and it is very promising that this metaheuristic framework can also improve state-of-the-art methods for other stochastic vehicle routing problems and for other stochastic combinatorial optimization problems.

Chapter 8

A Vehicle Routing Problem for the Collection of Exhausted Oil

In this chapter we focus on a real world application related to vehicle routing problems. This real world application is part of a project for the recycling of exhausted cooking oil into bio diesel fuel that has been initiated in 2010 by Caritas Suisse. Several tasks are crucial for a successful outcome of this project. Among them is the very important task of organizing an efficient and fairly balanced transportation of the old cooking oil from various locations to the processing plant. In this context fairly balanced means that the workload is distributed almost evenly among the different drivers and working days. We show in this chapter how we can use our experience with (stochastic) vehicle routing problems to contribute to this project. Additionally, it is very important for us to show that the theoretical and practical results from the previous chapters on academic problems can be transferred towards a real world problem. The project is currently in an early stage, and so far we were able to successfully support some decisions at this early stage. There is still a lot of work that has to be done to achieve our goal of an efficient and fair transportation of the old cooking oil. Therefore, we will also discuss our ideas and plans regarding the next steps in this project. Please note that so far stochasticity is not explicitly used in the model of this problem. We plan to include it in the near future and our model is constructed in a way that easily supports this. See the discussion at the end of this chapter for more details. The underlying publication for this chapter is Weyland et al. [2013b].

The remaining part of this chapter is organized in the following way. We start with a description of the real world project. After that we will show how

the transportation component of our real world problem can be modeled in terms of a vehicle routing problem. Using our experience from the optimization of various (stochastic) vehicle routing problems we then propose a heuristic to tackle this problem. We continue with computational studies, applying our heuristic on the original data provided by Caritas Suisse and varying important parameters like the location of the processing plant and the number of vehicles used. Finally, we finish with a discussion and with conclusions.

8.1 The Project Description

In 2010 Caritas Suisse initiated a project for the recycling of exhausted cooking oil into bio diesel fuel in Bali. In this region large quantities of exhausted cooking oil are accumulated in restaurants and hotels. The appropriate disposal of this cooking oil is still an unsolved problem and an inappropriate disposal could lead to some serious problems, ranging from the pollution of water up to health risks caused by reusing old cooking oil. The basic idea of the project is to organize and perform an appropriate disposal of the exhausted cooking oil by recycling it into bio diesel fuel. The quantities of oil that are accumulated in cooperating hotels and restaurants are regularly collected and brought to a processing plant. At that plant the oil is transformed into biofuel. In this way an appropriate disposal of the old cooking oil is ensured. An overview about the region of Bali involved and the cooperating hotels and restaurants (in gray) is given in figure 8.1, together with the three potential locations for the processing plant that were identified by Caritas Suisse (in red).

There are a lot of other important benefits resulting from this project. By ensuring an appropriate disposal of the oil, the environment in Bali is protected and risks for the health that are caused by reused oil are removed. Furthermore, the project helps the people living in Bali by creating new job opportunities. Finally, a contribution for the combat against climate change is made. Conventional fuel can be substituted by the bio diesel fuel obtained during the recycling process. Summarizing the benefits, it is a project with a huge potential for helping the local people and protecting the environment.

8.2 The Formal Model

As stated in section 8.1, the goal of the project is to organize and perform an appropriate disposal of exhausted cooking oil by recycling it into bio diesel fuel.

Figure 8.1. Bali, Indonesia. The region where the project is run, with cooperating hotels and restaurants highlighted in gray and potential locations for processing plants in red.

This recycling process is performed at a plant and one important component of the project is to provide an efficient transportation system for the collection of the oil. In this section we discuss the process of deriving a formal model for this transportation component.

More abstract we can see the transportation problem in the following way. We have been given a set of 308 different locations for the collection of oil. For each location we know the expected amount of oil that accumulates within one week (ranging from 1 to 22 cans of 25 liters each). Additionally, we assume that the location for the processing plant is given. In fact, there are only three possible locations for the processing plant and therefore we do not include the selection of the location into our model. The travel times between any two of the locations are known. The task is to collect the oil and to transport it to the location of the processing plant. For this purpose cans with a capacity of 25 liters are used. The vehicles start at the processing plant, filled with empty cans. Then they proceed to the different locations and exchange the empty cans with full cans, which are at the end delivered to the processing plant. The total capacity is 18 cans for each vehicle. The goal is to obtain a collection plan for a time horizon of two weeks (10 working days), which is then executed iteratively. For each vehicle and each working day routes have to be computed, such that all the accumulated oil is collected from the different locations and such that the workload between the single routes is balanced. The travel time for each route is not allowed to exceed eight hours due to working time constraints. Here the main optimization objective is to compute fairly balanced routes, the minimization of the total travel time is "only" a secondary objective. Nonetheless, the total travel time should be in a reasonable domain. Therefore, the objective function is a weighted sum of the travel time standard deviation among the different routes and the total travel time, where more emphasis is given for the travel time standard deviation. Our problem can then be modeled as a variant of the PERIODIC VEHICLE ROUTING PROBLEM (PVRP, see Angelelli and Speranza [2002]; Cordeau et al. [1997]; Gaudioso and Paletta [1992]) and in the remaining part of this section we will derive such a model.

First of all, for our real world problem we have to chose among three potential locations for the processing plant and to specify the number of vehicles used. Since we only have very few choices for those parameters, we do not model them as decision variables. Instead, we assume that the location of the processing plant and the number of vehicles is known. With some experiments a proper choice can be made for those two parameters, which are then used as fixed parameters for subsequent experiments. Another difference to the PVRP is that we do not have the frequencies with which the different locations have

to be visited. Therefore, we derive the number of visits for each location from the given amount of oil that accumulates at that location within one week. With these modifications our model can now be formulated in the following way.

We have been given a time horizon of 10 working days, a depot and a set of locations together with the travel times between all the locations. For each location we additionally know the required number of visits within the given time horizon as well as the total amount of goods that have to be collected. For this collection a fixed number of vehicles with a fixed capacity are available at the depot. The optimization goal is to compute routes (starting at the depot, visiting the locations and finishing at the depot) for all the vehicles in the given time horizon, which minimize the standard deviation of travel times between the different vehicles and working days, such that all goods are collected from the different locations and such that the locations are visited exactly for the specified number of times. Additionally, we want to assure that the total travel time is within a reasonable domain. Therefore, we also include the total travel time into the optimization objective. For the total travel time we use a much smaller weight compared to the standard deviation of travel times between the different vehicles and working days.

More formal, we have given a set V of locations, including a special element v_0, the depot. The function $d : V \times V \to \mathbb{R}^+$ is representing the travel times in minutes between any two locations. We include in these travel times the time required to load and unload the vehicle. The function $s : V \setminus \{v_0\} \to \mathbb{N}$ specifies the number of visits for each customer and the function $g : V \setminus \{v_0\} \to \mathbb{R}^+$ specifies the demand for each customer in liters. We further assume that the demand accumulates uniformly over the time horizon (respecting weekends). The time horizon is 10 working days (2 weeks), the capacity of the cans c, the capacity of the vehicle C and the number of vehicles k are given. A solution τ to this problem is a set of routes $\tau_{i,j}$ ($i \in \{1,\dots,k\}, j \in \{1,\dots,10\}$), starting at the depot, finishing at the depot, visiting a customer at most once and respecting the following constraints. Let $t(\tau_{i,j})$ denote the total travel time of route $\tau_{i,j}$ and let $r(v,j)$ denote the demand of customer $v \in V \setminus \{v_0\}$ accumulated until the customer gets visited at working day j.

(i) For all $i \in \{1,\dots,k\}, j \in \{1,\dots,10\}$: $t(\tau_{i,j}) \leq 480$

(ii) For all $i \in \{1,\dots,k\}, j \in \{1,\dots,10\}$: $\sum_{v \in \tau_{i,j}, v \neq v_0} \lfloor r(v,j)/c \rfloor \leq C$

(iii) For all $v \in V \setminus \{v_0\}$: $\left|\{\tau_{i,j} : i \in \{1,\dots,k\}, j \in \{1,\dots,10\}, v \in \tau_{i,j}\}\right| = s(v)$

(iv) For all $v \in V \setminus \{v_0\}, j \in \{1,\dots,10\}$: $\left|\{\tau_{i,j} : i \in \{1,\dots,k\}, v \in \tau_{i,j}\}\right| \leq 1$

Constraint (i) ensures that the total travel time for each route does not exceed the working time limit of eight hours and constraint (ii) ensures that the vehicle capacity is respected. The third constraint specifies that each customer is visited the required number of times during the time horizon. Constraint (iv) further ensures that a customer is not visited twice per day. The objective function now is a weighted sum of the total travel time,

$$f_1(\tau) = \sum_{i\in\{1,\dots,k\},j\in\{1,\dots,10\}} t(\tau_{i,j}),$$

and the standard deviation of the travel time,

$$f_2(\tau) = \sqrt{\frac{1}{10k} \sum_{i\in\{1,\dots,k\},j\in\{1,\dots,10\}} \left(t(\tau_{i,j}) - f_1(\tau)/10k\right)^2}.$$

Using a weight parameter α, the objective function can then be written as

$$f(\tau) = \alpha f_1(\tau) + f_2(\tau).$$

This allows us to formally state the problem.

Problem 9. *Given a set V with a special element $v_0 \in V$, a function $d : V \times V \to \mathbb{R}^+$, a function $s : V \setminus \{v_0\} \to \mathbb{N}$, a function $g : V \setminus \{v_0\} \to \mathbb{R}^+$ and values $c \in \mathbb{N}, C \in \mathbb{N}$ and $k \in \mathbb{N}$, the problem is to compute a set of routes $\tau^\star$, such that $f(\tau^\star) \leq f(\tau)$ for any other set of routes τe.*

8.3 A Heuristic Approach

Our background in the development of efficient heuristics for different vehicle routing problems is strong and the considerations for obtaining a heuristic for the problem in this work are based on this experience. Various such vehicle routing problems are tackled for example in Donati et al. [2008]; Gambardella et al. [2003a,b]; Montemanni et al. [2005, 2007]; Rizzoli et al. [2003, 2007]. At the end we decided to use a local search algorithm to tackle the problem in this work. Different variants of local search algorithms could be applied successfully for many vehicle routing problems (see Weyland et al. [2009a,b] for representative examples) and considering the similarities to our problem, such an approach seems very promising.

Preliminary experiments have shown that the number of locations that are visited by the same vehicle at the same working day are usually quite small

and mainly in the range between 6 and 12. Having an assignment of between 6 and 12 locations for one vehicle, the optimal route can be computed in a straightforward way. For this reason we propose a two level approach similar to the assign first strategy used for PERIODIC VEHICLE ROUTING PROBLEMS (Baptista et al. [2002]). The idea is to optimize the assignment of customers to vehicles and working days on the first level using the local search algorithm. An inner optimization mechanism is then used on the second level to optimize the routes of the different vehicles and working days. An additional benefit of the hybridization between the local search algorithm and an inner optimization mechanism is the following. The changes imposed by the local search algorithm affect a solution locally and this can be exploited by the second level optimization mechanism, which only has to re-optimize the parts that have changed. In the following we discuss our approach more in detail. We additionally give a high level description of the approach in algorithm 12.

Algorithm 12 High level description of the heuristic.

```
1: Create a random assignment of locations to vehicles / working days
2: for each vehicle and working day do
3:    Compute the optimal route for the corresponding assignment using the
      second level optimization mechanism
4: end for
5: Use the resulting solution to start the local search algorithm
6: while the current solution is not a local optimal solution do
7:    for each neighbor solution (explored in a random order) do
8:       Perform the local search move
9:       Re-optimize the routes which were changed
10:      if the neighbor solution improves over the current solution then
11:         Replace the current solution with the neighbor solution
12:         Start a new iteration (go to 7)
13:      end if
14:   end for
15: end while
16: return The local optimum obtained by the local search algorithm
```

The heuristic starts by randomly assigning the different locations to vehicles and working days. Then for each vehicle and each working day an optimal route

is computed using the second level optimization mechanism. The resulting solution is the starting solution for the local search algorithm. For the local search neighborhoods we use a shift operator similar to the 1-shift operator which has been introduced in chapter 5 and a swap operator. The shift operator takes a location assigned to a specific vehicle and a specific working day and shifts it to another vehicle and/or another working day. The swap operator considers two different locations and interchanges their assignments to vehicles and working days. In each iteration the local search algorithm explores all the solutions in the local search neighborhoods induced by those two operators in a random order. Note that both operators change only the assignment of locations to vehicles and working days for exactly two pairs of vehicles and working days. Therefore, the second level optimization mechanism is used to re-optimize the routes for only those two assignments that differ from the current solution. Here the second level optimization mechanism computes an optimal route by complete enumeration in the case where at most 8 locations are assigned to a specific vehicle and working day, whereas due to limitations of the computational time a fast heuristic approach is used if more than 8 locations are involved. This heuristic approach tries to insert the new location in the best possible way without changing the order of the locations that are already visited by that vehicle on that working day. The resulting solutions are then evaluated and compared to the current solution. If an improving solution is found, this solution replaces the current solution and a new iteration is started immediately. If there is no improving solution in the neighborhood of the current solution, the local search algorithm terminates and the local optimal solution is returned by our approach as the final solution.

8.4 Computational Studies

In this section we show the results of a computational study performed with our heuristic. For this purpose we use the original data of the problem which has been provided by Caritas Suisse. There are still two very important properties that need to be decided. One of them is the location of the processing plant. Here we can chose between three different locations: *Serangan*, *Jl. Kargo Permai* and *Green School* (see figure 8.1). The other one is the number of vehicles used. The main purpose of the computational study is to support the decision process about the location of the plant and the number of vehicles used. Therefore, we run the algorithm several times with different locations for the processing plant and with different numbers of vehicles.

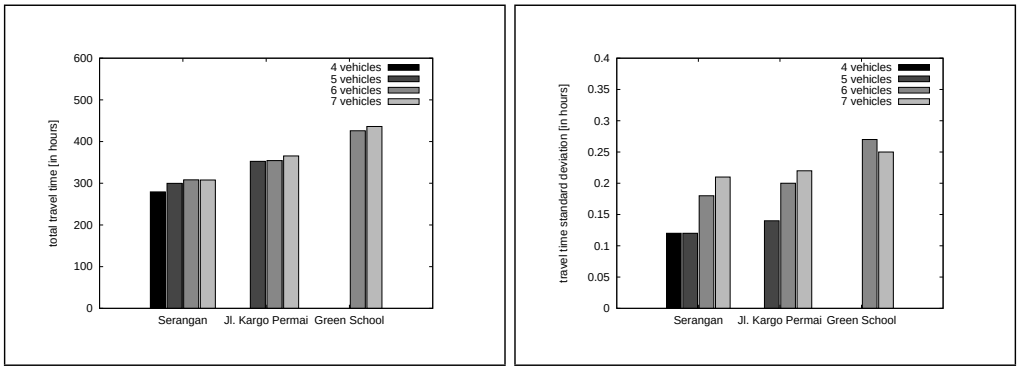
We have implemented the algorithm in ANSI C using the gcc compiler with common optimization flags. For the experiments an Intel Core2 Duo machine running at 2.53GHz was used. Due to feasibility constraints, we have used values between 4 and 7 for the number of vehicles. Together with the three different locations for the processing plant this results in a total of 12 experiments. Each experiment was performed for a total of 10 runs. The results are summarized in table 8.1. Here we report for each combination of the number of vehicles and the plant location the average values obtained in the 10 different runs for the total travel time, the travel time standard deviation and the minimum/maximum travel time over all vehicles and working days (all values in hours). Notice that feasible solutions, where the travel times of all routes do not exceed eight hours, are printed in boldface. Additionally, we have visualized the total travel times and the travel time standard deviations for different vehicle numbers and different plant locations in figures 8.2 (a) and 8.2 (b). Note that this visualization contains only the feasible solutions.

First of all, we see that not all of the experiments result in feasible solutions. Using 4 vehicles for the location *Jl. Kargo Permai* and 4 or 5 vehicles for the location *Green School* leads to infeasible solutions, since the travel time of some routes exceeds eight hours. One of the main goals in our work is to balance the workload while maintaining efficient routes in terms of travel times. This goal has been reached for the solutions using 4 or 5 vehicles for the location *Serangan* and 5 vehicles for the location *Jl. Kargo Permai*. In terms of the total travel time, solutions using the location *Serangan* seem to be slightly better than solutions using the location *Jl. Kargo Permai*. Nonetheless, for a final decision additional economical considerations and constraints (which are not part of our study) have to be respected. Based on our results we are able to exclude the location *Green School* for further considerations and to support the decision process in this way. It is interesting to report that following our study, as well as additional economic considerations, the location finally selected by Caritas Suisse has been in fact *Jl. Kargo Permai*.

We want to finish this section with a short discussion about the performance of our heuristic. The computational time for a single run is in the range of between 3 and 10 minutes, depending on the number of vehicles and also on the plant location. Additionally, our heuristic is able to fairly balance the workload between single routes, especially if the number of vehicles is 4 or 5. In those cases the difference between the longest and the shortest route over all vehicles and working days is less than one hour. This is a remarkable achievement, in particular because the single routes consist only of very few locations.

plant location	vehicles	travel time			
		total	standard deviation	minimum	maximum
Serangan	4	**279.40**	**0.12**	**6.67**	**7.39**
	5	**299.85**	**0.12**	**5.64**	**6.37**
	6	**308.32**	**0.18**	**4.72**	**5.97**
	7	**307.79**	**0.21**	**4.05**	**5.92**
Jl. Kargo Permai	4	339.30	0.11	8.13	8.87
	5	**352.60**	**0.14**	**6.68**	**7.62**
	6	**354.35**	**0.20**	**5.52**	**7.04**
	7	**365.49**	**0.22**	**4.88**	**6.76**
Green School	4	411.68	0.14	9.95	10.77
	5	418.67	0.18	7.78	9.09
	6	**425.95**	**0.27**	**6.36**	**7.79**
	7	**436.24**	**0.25**	**5.80**	**7.19**

Table 8.1. Computational results (absolute values in hours). Feasible solutions are printed in boldface.



(a) total travel times (b) standard deviations

Figure 8.2. The total travel times and the travel time standard deviations for feasible solutions using different numbers of vehicles and different locations for the plant. The results are given in hours, grouped by plant locations.

8.5 Discussion and Conclusions

In this chapter we showed how to model the transportation component of a real world oil collection problem as a vehicle routing problem. This problem is an integral part of a recycling project in Bali, helping the people in this region, protecting the environment and making a contribution to combat climate change. Based on our experience with vehicle routing problems we proposed a local search algorithm for this problem. This algorithm assigns the different collection points to the vehicles and the working days. An inner optimization mechanism is then used to optimize the routes for the different vehicles and working days. With this approach it is possible to obtain solutions in which the workload among the vehicles and working days is fairly balanced. Additionally, we could gain useful information for the decision process regarding the number of vehicles and the location of the processing plant.

We want to finish the chapter with a brief outlook about our ideas and plans for the future of this project. An important issue is that we have only very limited information about the actual travel times between the different locations at the moment. Here the idea is to gather additional and more reliable information in a test phase of the project where the vehicles are used on computed routes to verify our results. Depending on the results of this phase we could also decide to model the travel times time-dependently and/or in a stochastic way. The corresponding modifications to our current algorithm would be straightforward. The other important issue is the modeling of the amount of oil that has to be collected. So far we are using fixed values which are basically estimates of the demands using average values. In the future we want to extend our model and to incorporate also stochastic demands. By combining our existing algorithm with an approach based on Monte Carlo sampling, as it has been used in chapters 5 and 6, only slight modifications are necessary in this case. All in all, we want to use the available information in the best possible way to create a realistic model for the transportation component and to obtain high quality solutions for this problem. Based on the results of the other chapters and on our experience we believe that using stochastic information for the demands (and maybe also for the travel times) is the right choice here.

Chapter 9

Conclusions

At the beginning of the thesis we have presented a convergence result for vehicle routing problems with stochastic demands. Here we have shown a kind of asymptotic equivalence between the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS and the TRAVELING SALESMAN PROBLEM, as well as between the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS and the PROBABILISTIC TRAVELING SALESMAN PROBLEM. With this result we were able to give the first theoretical explanations for some phenomena that were observed regarding to those problems. We could further show that in certain situations it makes sense to treat a vehicle routing problem with stochastic demands like a conventional (non-stochastic) vehicle routing problem. Those results require that only a single vehicle (or a very small number of vehicles) is used and that a lot of restocking actions are performed. Additionally, the number of customers that have to be served should be quite large. Here we also assume that the basic restocking strategy is used. The cases in which those conditions are not fulfilled are worth for further investigations. In fact, it is of high interest to characterize those variants of vehicle routing problems with stochastic demands that differ significantly from their non-stochastic counterparts.

We then continued with investigations of (*substantially*) stochastic instances of stochastic vehicle routing problems. These problems also contain non-stochastic instances as special cases. With (*substantially*) stochastic instances we refer to instances which make extensive use of modeling input in a stochastic way and which differ significantly from the non-stochastic instances. Here we could show several hardness results for the exact optimization and the approximation of the PROBABILISTIC TRAVELING SALESMAN PROBLEM, the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS and the VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS AND CUSTOMERS. All in all, those problems remain difficult to solve (or approximate), even if we consider only (*substantially*) stochastic in-

stances. We think that it is of great importance to show the hardness for such stochastic instances, since those instances are the reason why stochastic vehicle routing problems are used instead of conventional (non-stochastic) vehicle routing problems. We hope that similar results will be derived for other stochastic combinatorial optimization problems in the future.

After that we focused on the complexity of several computational tasks related to the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. We could show that the following computational tasks are #P-hard for Euclidean instances of the PTSPD: the computation of the probabilities for deadline violations, the evaluation of solutions, delta evaluation in reasonable local search neighborhoods, the decision variant and the optimization variant. To our knowledge, it has been shown for the first time that the evaluation of solutions for a practically relevant stochastic vehicle routing problem is a difficult computational task itself. Additionally, we could show that the PTSPD is more difficult to solve from a complexity point of view than its non-stochastic counterpart, the TRAVELING SALESMAN PROBLEM WITH DEADLINES. In our opinion, similar results can be derived for a large number of other stochastic vehicle routing problems and stochastic combinatorial optimization problems. This shows that many stochastic combinatorial optimization problems are much harder to solve from a complexity point of view, a fact that has already been experienced by many practitioners. Finally, we were able to reveal an interesting connection between stochastic vehicle routing problems, or more general stochastic combinatorial optimization problems, and the class #P of counting problems. We believe that there is a very deep relationship between these two classes of problems. Recently, we could also show that the complexity of various computational tasks for counting problems are upper bounds for the complexity of several computational tasks for stochastic combinatorial optimization problems. Future research in this direction seems to be of great importance.

In our first chapter regarding practical applications of stochastic vehicle routing problems, we developed efficient local search algorithms and efficient heuristics for the PROBABILISTIC TRAVELING SALESMAN PROBLEM. The local search algorithms are based on the former state-of-the-art local search algorithm and are using a combination of an analytical approximation of the objective function and an approximation based on Monte Carlo sampling. Additionally, many advanced local search techniques like *don't look bits*, *neighborhood lists* and *delta evaluation* were used for this algorithm. This local search algorithm was then used within a random restart local search algorithm and an iterated local search algorithm. These algorithms are currently the state-of-the-art methods for the PTSP.

We then continued with the development of efficient problem solving methods for the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. As we have seen in earlier chapters this is a computationally very demanding problem and even the evaluation of the solutions is a computationally difficult task. Therefore, the usage of heuristics, in particular in combination with approximations of the objective function, is very well motivated in this case. Approximations for the objective function introduced in literature do not lead to a satisfactory approximation behavior and therefore we proposed an approximation based on Monte Carlo sampling. Theoretically, this approximation also does not lead to a satisfactory approximation behavior, but in practical applications it outperforms the other methods by far. We used this approximation within a local search algorithm and obtained very promising results. We then used this local search algorithm within a random restart local search algorithm. This approach achieves very good results and is the current state-of-the-art method for the PTSPD.

During our research for the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES we were able to identify that sampling based methods for the optimization of stochastic combinatorial optimization problems allow for an efficient parallelization on a very low level. In fact, for many such problems it is possible to parallelize the evaluation of solutions on the level of a single sample, that means we perform the evaluation of one sample on one solution many times in parallel. We proposed this approach as a framework for the optimization of stochastic combinatorial optimization problems and applied this framework to the PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH DEADLINES. In this way we could demonstrate the efficiency of our framework. The results obtained are very promising, in fact the computational time for the state-of-the-art heuristics for the PTSPD could be reduced by up to two orders of magnitude. We are sure that these results also generalize to sampling based heuristics for other stochastic vehicle routing problems and other stochastic combinatorial optimization problems.

We then turned to a real world vehicle routing problem, finishing our journey “from theory to practice”. This project has been initiated by the Caritas Suisse and is still in an early stage. Here we were able to use our knowledge and experience with the optimization of vehicle routing problems and stochastic vehicle routing problems to support the early stage decision process for this project. More in detail, our results could help in selecting a location for the depot and in determining an appropriate number of vehicles that were purchased for the project. For the future our model has to be changed in many different ways. In particular, we have to consider better estimates for the travel times, the

actual stochastic demands and time dependent travel times, just to mention the most important changes. For us it is very important that we could show that our theoretical and practical results obtained in this thesis are not only of academic interest. They can be easily adapted for tackling real world problems. So far we have successfully supported an early stage decision process for the project and we feel certain that we are also able to contribute to the project in other fundamental ways.

At this point, let us come to a final conclusion. All in all, we strongly believe that with our work we could significantly contribute to both the theoretical understanding and the practical applications of stochastic vehicle routing problems. Our results open a lot of new directions for further research. In our opinion the most promising ones are the following. First of all, we believe that sampling based methods have a huge potential for solving stochastic vehicle routing problems and stochastic combinatorial optimizations in general. Surprisingly, there exist many problems for which such methods have not been used yet and we hope that with our work we contribute towards a change of this situation. Secondly, we think that it is important to prove hardness results for (*substantially*) stochastic instances of stochastic vehicle routing problems and stochastic combinatorial optimization problems. In this way we are able to give a strong motivation for the usage of heuristics for solving these problems. Last but not least, we are sure that there exists a deep relationship between stochastic combinatorial optimization problems and the class #P of counting problems. With our work we have shown a first connection between these two classes of problems. We believe that there is much more to show and future research in this direction seems to be extremely promising and of great importance.

Appendix A

Convergence Results for Vehicle Routing Problems with Stochastic Demands

In this chapter we present the material that has been omitted in chapter 2. It is basically the proof for a simple property of cyclic matrices and proofs regarding invariances of the gcd property.

A.1 Cyclic Matrices

Lemma 13. *Let $A = (a_{ij})$ be a $m \times m$ cyclic matrix and let $B = (b_{ij})$ be a $m \times m$ cyclic matrix. Then $C = AB$ is a $m \times m$ cyclic matrix.*

Proof. We have $\forall i \in \{0, 1, \ldots, m-1\}, \ \forall j \in \{0, 1, \ldots, m-1\}$:

$$\begin{aligned} c_{ij} &= \sum_{k=1}^{m} a_{ik} b_{kj} \\ &= \sum_{k=1}^{m} a_{i+1,k+1} \ b_{k+1,j+1} \\ &= \sum_{k=1}^{m} a_{i+1,k} \ b_{k,j+1} \\ &= c_{i+1,j+1}, \end{aligned}$$

where the indices are taken modulo m. □

A.2 Invariances of the gcd Property

First we show that the gcd property does not depend on the index of the strictly positive entry relative to which the differences are taken.

Theorem 22. *Let $A = (a_{ij})$ be a cyclic $m \times m$ matrix with only non-negative entries and with strictly positive entries $a_{i_1 0}, a_{i_2 0}, \ldots, a_{i_l 0}$, $i_1 < i_2 < \ldots < i_l$ in the first column. Then the following statements are equivalent:*

(i) A fulfills the gcd property.

(ii) There exists a $k \in \{1, 2, \ldots, l\}$, s.t. the greatest common divisor of the differences $i_k - i_j$, $j \in \{1, 2, \ldots, l\} \setminus \{k\}$, and m is 1.

(iii) For all $k \in \{1, 2, \ldots, l\}$ the greatest common divisor of the differences $i_k - i_j$, $j \in \{1, 2, \ldots, l\} \setminus \{k\}$, and m is 1.

Proof. The implications (i) $\Rightarrow$ (ii) and (iii) $\Rightarrow$ (i) are trivial. So it is sufficient to prove the implication (ii) $\Rightarrow$ (iii). Let $k \in \{1, 2, \ldots, l\}$ be a value fulfilling (ii). We show that the property also holds for any other value $k' \in \{1, 2, \ldots, l\} \setminus \{k\}$. There exist values $c_1, c_2, \ldots, c_l, d \in \mathbb{Z}$, s.t. the following equivalences hold.

$$
\begin{aligned}
& \gcd(i_k - i_1, \ldots, i_k - i_{k-1}, i_k - i_{k+1}, \ldots, i_k - i_l, m) = 1 \\
\Leftrightarrow \quad & \sum_{t=1}^{l} c_t (i_k - i_t) + dm = 1 \\
\Leftrightarrow \quad & \sum_{t=1}^{l} c_t (i_k - i_{k'} + i_{k'} - i_t) + dm = 1 \\
\Leftrightarrow \quad & \sum_{t=1}^{l} c_t (i_{k'} - i_t) + \left(- \sum_{t=1}^{l} c_t \right) (i_{k'} - i_k) + dm = 1 \\
\Leftrightarrow \quad & \gcd(i_{k'} - i_1, \ldots, i_{k'} - i_{k'-1}, i_{k'} - i_{k'+1}, \ldots, i_{k'} - i_l, m) = 1
\end{aligned}
$$

Here we use the fact that $\mathbb{Z}$ is a principal ideal ring. For details about properties of principal ideal rings, see e.g. Baldoni et al. [2008]. □

Now we show that the property also does not depend on the column, from which we chose the indices of the strictly positive entries.

Theorem 23. *Let $A = (a_{ij})$ be a cyclic $m \times m$ matrix with only non-negative entries. Then the following statements are equivalent:*

(i) A fulfills the gcd property.

(ii) There exists a $k \in \{0, 1, \ldots, m-1\}$, s.t. the greatest common divisor of the differences $i_l^{(k)} - i_j^{(k)}$, $j \in \{1, 2, \ldots, l-1\}$, and m is 1, where $i_1^{(k)} < i_2^{(k)} < \ldots < i_l^{(k)}$ are the indices with strictly positive entries in column k.

(iii) For all $k \in \{0, 1, \ldots, m-1\}$ the greatest common divisor of the differences $i_l^{(k)} - i_j^{(k)}$, $j \in \{1, 2, \ldots, l-1\}$, and m is 1, where $i_1^{(k)} < i_2^{(k)} < \ldots < i_l^{(k)}$ are the indices with strictly positive entries in column k.

Proof. Again the implications (i) $\Rightarrow$ (ii) and (iii) $\Rightarrow$ (i) are trivial and we just have to show the implication (ii) $\Rightarrow$ (iii). For that purpose we show that if the property in (ii) is fulfilled for any $k > 0$ then it is also fulfilled for $k-1$ and if the property is fulfilled for any $k < m-1$ then it is also fulfilled for $k+1$. In this way we can prove (iii) by using this reasoning repeatedly. We show only the implication from k to $k-1$, since the implication from k to $k+1$ is analogue.

So let (ii) be fulfilled for some $k > 0$ and let $i_1^{(k)} < i_2^{(k)} < \ldots < i_l^{(k)}$ be the indices with strictly positive entries in column k. Now we distinguish the two cases, where the first entry in column k is zero or strictly positive, respectively.

In the first case we have $i_1^{(k)} > 0$. Then the strictly positive entries in column $k-1$ are

$$i_1^{(k-1)} = i_1^{(k)} - 1 < i_2^{(k-1)} = i_2^{(k)} - 1 < \ldots < i_l^{(k-1)} = i_l^{(k)} - 1$$

due to the cyclic structure of A. That means the corresponding differences are the same for both columns, since

$$i_l^{(k-1)} - i_j^{(k-1)} = \left(i_l^{(k)} - 1\right) - \left(i_j^{(k)} - 1\right) = i_l^{(k)} - i_j^{(k)}, \quad j \in \{1, 2, \ldots, l\},$$

and thus (ii) holds for $k-1$.

Now let $i_1^{(k)} = 0$. In this case we have strictly positive entries in column $k-1$ at the row indices

$$i_1^{(k-1)} = i_2^{(k)} - 1 < i_2^{(k-1)} = i_3^{(k)} - 1 < \ldots < i_{l-1}^{(k-1)} = i_l^{(k)} - 1 < i_l^{(k-1)} = m - 1,$$

again due to the cyclic structure of A. The differences with respect to the index $i_{l-1}^{(k-1)}$ (which is not the largest index of a strictly positive entry for column $k-1$) are

$$i_{l-1}^{(k-1)} - i_j^{(k-1)} = \left(i_l^{(k)} - 1\right) - \left(i_{j+1}^{(k)} - 1\right) = i_l^{(k)} - i_{j+1}^{(k)}, \quad j \in \{1, 2, \ldots, l-2\}$$

and

$$i_{l-1}^{(k-1)} - i_l^{(k-1)} = \left(i_l^{(k)} - 1\right) - (m-1) = i_l^{(k)} - m = i_l^{(k)} - i_1^{(k)} - m.$$

Every common divisor of the differences $i_{l-1}^{(k-1)} - i_j^{(k-1)}$, $j \in \{1, 2, \ldots, l-2, l\}$, and m would also be a common divisor of the differences $i_l^{(k)} - i_j^{(k)}$, $j \in \{1, 2, \ldots, l-1\}$, and m, and vice versa. Since the greatest common divisor of the latter differences and m is 1, the greatest common divisor of the former differences and m has to be 1, too. With theorem 22 we can conclude that the required property also holds for column $k-1$ in this case, which finishes the proof. □

It is also possible to chose a row instead of a column for the definition of the gcd property and to consider the indices of the strictly positive entries in that row. Due to the cyclic structure, the entries in the first row of the matrix are the same as in the last column, but in inverse order. We omit a formal proof for that statement, but a similar reasoning as in the proof of the last theorem can be used here.

Bibliography

E.H.L. Aarts and J. Korst. *Simulated annealing and Boltzmann machines*. Wiley Chichester, 1990.

E.H.L. Aarts and J.K. Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003. ISBN 0691115222.

Z. Adar and J.M. Griffin. Uncertainty and the choice of pollution control instruments. *Journal of Environmental Economics and Management*, 3(3):178–188, 1976.

C. Almeder and R.F. Hartl. A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer. *International Journal of Production Economics*, 2012.

E. Angelelli and M.G. Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247, 2002.

S. Arora and B. Barak. *Computational complexity: A modern approach*. Cambridge University Press, 2009.

P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Adaptive sample size and importance sampling in estimation-based local search for the probabilistic traveling salesman problem. *European Journal of Operational Research*, 199 (1):98–110, 2009a.

P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and M. Dorigo. Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem. *Swarm Intelligence*, 3(3):223–242, 2009b.

P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Estimation-based metaheuristics for the probabilistic traveling salesman problem. *Computers and Operations Research*, 37(11):1939–1951, 2010.

M.W. Baldoni, C. Ciliberto, and G.M.P. Cattaneo. *Elementary Number Theory, Cryptography and Codes*. Springer, 2008.

S. Baptista, R.C. Oliveira, and E. Zuquete. A period vehicle routing case study. *European Journal of Operational Research*, 139(2):220–229, 2002.

G. Barbarosoğlu and Y. Arda. A two-stage stochastic programming framework for transportation planning in disaster response. *Journal of the Operational Research Society*, 55(1):43–53, 2004.

C. Bastian and A.H.G. Rinnooy Kan. The stochastic vehicle routing problem revisited. *European Journal of Operational Research*, 56(3):407–412, 1992.

A. Ben-Tal and A. Nemirovski. Robust optimization–methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

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

D.J. Bertsimas and L.H. Howell. Further results on the probabilistic traveling salesman problem. *European Journal of Operational Research*, 65(1):68–95, 1993.

D.J. Bertsimas, P. Jaillet, and A.R. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

D.J. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic vehicle routing problems. *Transportation science*, 29(4):342–352, 1995.

H.G. Beyer and H.P. Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002. ISSN 1567-7818.

H.G. Beyer and B. Sendhoff. Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33):3190–3218, 2007.

L. Bianchi and L.M. Gambardella. Ant colony optimization and local search based on exact and estimated objective values for the probabilistic traveling salesman problem. Technical Report 06 - 07, IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, June 2007.

L. Bianchi, L.M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. *Parallel Problem Solving from Nature–PPSN VII*, pages 883–892, 2002a.

L. Bianchi, L.M. Gambardella, and M. Dorigo. Solving the homogeneous probabilistic traveling salesman problem by the ACO metaheuristic. *Ant Algorithms*, pages 25–38, 2002b.

L. Bianchi, L.M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. *Lecture Notes in Computer Science*, pages 883–892, 2003.

L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Metaheuristics for the vehicle routing problem with stochastic demands. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 450–460. Springer, 2004.

L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms*, 5(1):91–110, 2006.

L. Bianchi, M. Dorigo, L.M. Gambardella, and W.J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2): 239–287, 2009.

M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Estimation-based local search for stochastic combinatorial optimization using delta evaluations: A case study on the probabilistic traveling salesman problem. *INFORMS Journal on Computing*, 20(4):644–658, 2008a.

M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Estimation-based local search for stochastic combinatorial optimization using delta evaluations: A case study on the probabilistic traveling salesman problem. *INFORMS Journal on Computing*, 20(4):644–658, 2008b.

J.R. Birge and F.V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988.

H.J. Böckenhauer, J. Hromkovič, T. Mömke, and P. Widmayer. On the hardness of reoptimization. *SOFSEM 2008: Theory and Practice of Computer Science*, pages 50–65, 2008.

J. Branke and M. Guntsch. New ideas for applying ant colony optimization to the probabilistic tsp. *Applications of Evolutionary Computing*, pages 127–134, 2003.

J. Branke and M. Guntsch. Solving the probabilistic tsp with ant colony optimization. *Journal of Mathematical Modelling and Algorithms*, 3(4):403–425, 2004.

A.M. Campbell. Aggregation for the probabilistic traveling salesman problem. *Computers and Operations Research*, 33(9):2703–2724, 2006.

A.M. Campbell and B.W. Thomas. Challenges and advances in a priori routing. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 123–142, 2008a.

A.M. Campbell and B.W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42(1):1–21, 2008b.

A.M. Campbell and B.W. Thomas. Runtime reduction techniques for the probabilistic traveling salesman problem with deadlines. *Computers and Operations Research*, 36(4):1231–1248, 2009.

C.C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1):451–464, 1998.

R.L. Carraway, T.L. Morin, and H. Moskowitz. Generalized dynamic programming for stochastic combinatorial optimization. *Operations Research*, 37(5): 819–829, 1989.

C. Cervellera, V.C.P. Chen, and A. Wen. Optimization of a large-scale water reservoir network by stochastic dynamic programming with efficient state space discretization. *European Journal of Operational Research*, 171(3):1139–1151, 2006.

M.S. Chang, Y.L. Tseng, and J.W. Chen. A scenario planning approach for the flood emergency logistics preparation problem under uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):737–754, 2007.

G. Chen, M.S. Daskin, Z.J.M. Shen, and S. Uryasev. The α-reliable mean-excess regret model for stochastic facility location modeling. *Naval Research Logistics (NRL)*, 53(7):617–626, 2006.

K. Chepuri and T. Homem-de Mello. Solving the vehicle routing problem with stochastic demands using the cross-entropy method. *Annals of Operations Research*, 134(1):153–181, 2005.

R.K. Cheung and C.Y. Chen. A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. *Transportation Science*, 32(2):142–162, 1998.

D.M. Chitty. A data parallel approach to genetic programming using programmable graphics hardware. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1566–1573. ACM, 2007.

A. Choong, R. Beidas, and J. Zhu. Parallelizing simulated annealing-based placement using GPGPU. In *2010 International Conference on Field Programmable Logic and Applications*, pages 31–34. IEEE, 2010.

C.H. Christiansen and J. Lysgaard. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35(6):773–781, 2007.

L. Cooper and L.J. Leblanc. Stochastic transportation problems and other newtork related convex problems. *Naval Research Logistics Quarterly*, 24(2):327–337, 1977.

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

M. Czapinski and S. Barnes. Tabu search with two approaches to parallel flowshop evaluation on CUDA platform. *Journal of Parallel and Distributed Computing*, 71(6):802–811, 2011. ISSN 0743-7315.

E. Delage. Re-optimization of technician tours in dynamic environments with stochastic service time. *Rapport de stage du Master ORO*, 2010.

M. Delgado, M.A. Vila, J. Kaprzyk, and J.L. Verdegay. *Fuzzy optimization: Recent advances*. Springer-Verlag New York, Inc., 1994.

D. Dentcheva and W. Römisch. Optimal power generation under uncertainty via stochastic programming. 1998.

K. Dhamdhere, R. Ravi, and M. Singh. On two-stage stochastic minimum spanning trees. *Integer Programming and Combinatorial Optimization*, pages 189–199, 2005.

U. Diwekar. Optimization under uncertainty. *Introduction to Applied Optimization*, pages 1–54, 2008.

A.V. Donati, R. Montemanni, N. Casagrande, A.E. Rizzoli, and L.M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185(3):1174–1191, 2008.

M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. *Handbook of metaheuristics*, pages 250–285, 2003.

M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172, 1999. ISSN 1064-5462.

M. Dror and P. Trudeau. Stochastic vehicle routing with modified savings algorithm. *European Journal of Operational Research*, 23(2):228–235, 1986.

M. Dror, G. Laporte, and F.V. Louveaux. Vehicle routing with stochastic demands and restricted failures. *Mathematical Methods of Operations Research*, 37(3): 273–283, 1993. ISSN 1432-2994.

Y. Dumas, J. Desrosiers, E. Gelinas, and M.M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432, 2006.

Y.M. Ermoliev and G. Leonardi. Some proposals for stochastic facility location models. *Mathematical Modelling*, 3(5):407–420, 1982.

K.L. Fok, T.T. Wong, and M.L. Wong. Evolutionary computing on consumer graphics hardware. *Intelligent systems*, 22(2):69–78, 2007. ISSN 1541-1672.

T.D. Frank, P.J. Beek, and R. Friedrich. Fokker-Planck perspective on stochastic delay systems: Exact solutions and data analysis of biological systems. *Physical Review E*, 68(2):021912, 2003.

R.M. Freund. Optimization under uncertainty. *Massachusetts Institute of Technology*, pages 18–27, 2004.

F. FuCe, W. Hui, and Z.L. Ying. Solving the vehicle routing problem with stochastic demands and customers. In *Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2005.*, pages 736–739, 2005.

L.M. Gambardella, N. Casagrande, R. Montemanni, F. Oliverio, and A.E. Rizzoli. Ant colony optimization applied to industrial vehicle routing problems. In *ECCO 2003, XVI Conference of the European Chapter on Combinatorial Optimisation*, pages 5–7, 2003a.

L.M. Gambardella, A.E. Rizzoli, F. Oliverio, N. Casagrande, A.V. Donati, R. Montemanni, and E. Lucibello. Ant colony optimization for vehicle routing in advanced logistic systems. In *Proceedings of MAS 2003 – International Workshop on Modeling and Applied Simulation, Bergeggi, Italy*, pages 3–9, 2003b.

M. Gaudioso and G. Paletta. A heuristic for the periodic vehicle routing problem. *Transportation Science*, 26(2):86–92, 1992.

M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29(2):143–155, 1995.

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

M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477, 1996b.

F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Pub, 1998. ISBN 0792381874.

D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989. ISBN 0201157675.

G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

H. Gordon. *Discrete probability*. Springer, 1997.

L. Gouveia and S. Voß. A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research*, 83(1): 69–82, 1995.

G. Grimmett and D. Welsh. *Probability: An introduction*. Clarendon Press, 1986.

W. Gutjahr. A converging ACO algorithm for stochastic combinatorial optimization. *Stochastic Algorithms: Foundations and Applications*, pages 10–25, 2003.

W. Gutjahr. S-ACO: An ant-based approach to combinatorial optimization under uncertainty. *Ant Colony Optimization and Swarm Intelligence*, pages 157–174, 2004.

M. Haimovich and A.H.G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, pages 527–542, 1985.

S. Harding and W. Banzhaf. Fast genetic programming on GPUs. *Genetic Programming*, pages 90–101, 2007.

M. Hariga and M. Ben-Daya. Some stochastic inventory models with deterministic variable lead time. *European Journal of Operational Research*, 113(1): 42–51, 1999.

M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

V.C. Hemmelmayr, K.F. Doerner, and R.F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

P.V. Hentenryck and R. Bent. *Online stochastic combinatorial optimization*. The MIT Press, 2009.

C. Hjorring and J. Holt. New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research*, 86:569–584, 1999.

A. Hoff, A.G. Lium, A. Løkketangen, and T.G. Crainic. A metaheuristic for stochastic service network design. *Journal of Heuristics*, 16(5):653–679, 2010.

R.D. Horan. Cost-effective and stochastic dominance approaches to stochastic pollution control. *Environmental and Resource Economics*, 18(4):373–389, 2001.

R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge University Press, 1990.

G.H. Huang and D.P. Loucks. An inexact two-stage stochastic programming model for water resources management under uncertainty. *Civil Engineering Systems*, 17(2):95–118, 2000.

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

L.M. Hvattum, A. Løkketangen, and G. Laporte. Scenario tree-based heuristics for stochastic inventory-routing problems. *INFORMS Journal on Computing*, 21(2):268–285, 2009.

N. Immorlica, D. Karger, M. Minkoff, and V.S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 691–700. Society for Industrial and Applied Mathematics, 2004.

M. Inuiguchi and J. Ramık. Possibilistic linear programming: A brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem. *Fuzzy sets and systems*, 111(1):3–28, 2000.

S.S. Isukapalli, A. Roy, and P.G. Georgopoulos. Stochastic response surface methods (SRSMs) for uncertainty propagation: Application to environmental and biological systems. *Risk analysis*, 18(3):351–363, 1998.

P. Jaillet. *Probabilistic traveling salesman problems*. PhD thesis, M. I. T., Dept. of Civil Engineering, 1985.

J. Jin, T.G. Crainic, and A. Løkketangen. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*, 222(3):441–451, 2012.

D.S. Johnson and L.A. McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, pages 215–310, 1997.

M. Karamouz and H.V. Vasiliadis. Bayesian stochastic optimization of reservoir operation using uncertain forecasts. *Water Resources Research*, 28(5):1221–1232, 1992.

R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

A.S. Kenyon and D.P. Morton. Stochastic vehicle routing with random travel times. *Transportation Science*, 37(1):69–82, 2003.

W.K. Klein Haneveld and M.H. van der Vlerk. Stochastic integer programming: General models and algorithms. *Annals of Operations Research*, 85:39–57, 1999.

P.J.M. Laarhoven and E.H.L. Aarts. *Simulated annealing: Theory and applications*. Springer, 1987. ISBN 9027725136.

G. Laporte, F.V. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations research*, 42(3):543–549, 1994.

G. Laporte, F.V. Louveaux, and L. Van Hamme. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, 2002.

G. Latouche and V. Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*, volume 5. Society for Industrial Mathematics, 1987.

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The traveling salesman problem: A guided tour of combinatorial optimization*. Wiley New York, 1985.

J. Li, X. Hu, Z. Pang, and K. Qian. A parallel ant colony optimization algorithm based on fine-grained model with GPU-acceleration. *International Journal of Innovative Computing, Information and Control*, 5(11):3707–3716, 2009.

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

B. Liu and K.K. Lai. Stochastic programming models for vehicle routing problems. In *Focus on computational neurobiology*, pages 13–27. Nova Science Publishers, Inc., 2004.

J. Liu. *Portfolio selection in stochastic environments*. PhD thesis, Stanford University, 1999.

Y.H. Liu. A hybrid scatter search for the probabilistic traveling salesman problem. *Computers and Operations Research*, 34(10):2949–2963, 2007.

Y.H. Liu. Diversified local search strategy under scatter search framework for the probabilistic traveling salesman problem. *European Journal of Operational Research*, 191(2):332–346, 2008a.

Y.H. Liu. A memetic algorithm for the probabilistic traveling salesman problem. In *IEEE Congress on Evolutionary Computation, CEC 2008*, pages 146–152, 2008b.

Y.H. Liu. Different initial solution generators in genetic algorithms for solving the probabilistic traveling salesman problem. *Applied mathematics and computation*, 216(1):125–137, 2010.

F.V. Louveaux and D. Peeters. A dual-based procedure for stochastic facility location. *Operations research*, 40(3):564–573, 1992.

J. Lu, D.W.C. Ho, and Z. Wang. Pinning stabilization of linearly coupled stochastic neural networks via minimum number of controllers. *IEEE Transactions on Neural Networks*, 20(10):1617–1629, 2009.

A. Lucena. Time-dependent traveling salesman problem–the deliveryman case. *Networks*, 20(6):753–763, 1990.

M.K. Luhandjula and M.M. Gupta. On fuzzy stochastic optimization. *Fuzzy Sets and Systems*, 81(1):47–55, 1996.

I. Maqsood and G.H. Huang. A two-stage interval-stochastic programming model for waste management under uncertainty. *Journal of the Air and Waste Management Association*, 53(5):540–552, 2003.

Y. Marinakis and M. Marinaki. A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problem. In *IEEE Congress on Evolutionary Computation, CEC 2009*, pages 1762–1769, 2009.

Y. Marinakis and M. Marinaki. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. *Computers and Operations Research*, 37(3):432–442, 2010.

Y. Marinakis, A. Migdalas, and P.M. Pardalos. Expanding neighborhood search–GRASP for the probabilistic traveling salesman problem. *Optimization Letters*, 2(3):351–361, 2008.

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

J.E. Mendoza, B. Castanier, C. Guéret, A.L. Medaglia, and N. Velasco. Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science*, 45(3):346–363, 2011.

E. Miller-Hooks and H. Mahmassani. Path comparisons for a priori and time-adaptive decisions in stochastic, time-varying networks. *European Journal of Operational Research*, 146(1):67–82, 2003.

R. Montemanni, L.M. Gambardella, A.E. Rizzoli, and A.V. Donati. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10:327–343, 2005.

R. Montemanni, J. Barta, M. Mastrolilli, and L.M. Gambardella. The robust traveling salesman problem with interval data. *Transportation Science*, 41(3): 366–381, 2007.

B. Morris and A. Sinclair. Random walks on truncated cubes and sampling 0-1 knapsack solutions. *SIAM Journal on Computing*, 34:195, 2004.

C. Murat and V.T. Paschos. A priori optimization for the probabilistic maximum independent set problem. *Theoretical Computer Science*, 270(1):561–590, 2002.

C.V. Negoita and D.A. Ralescu. On fuzzy optimization. *Kybernetes*, 6(3):193–195, 1977.

C. Novoa and R. Storer. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509–515, 2009.

M.P. Nowak and W. Römisch. Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *Annals of Operations Research*, 100(1):251–272, 2000.

T. Nuortio, J. Kytöjoki, H. Niska, and O. Bräysy. Improved route planning and scheduling of waste collection and transport. *Expert Systems with Applications*, 30(2):223–232, 2006.

C.H. Papadimitriou and S. Vempala. On the approximability of the traveling salesman problem. *Combinatorica*, 26(1):101–120, 2006.

R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007. ISSN 1935-3812.

E.L. Porteus. Stochastic inventory theory. *Handbooks in Operations Research and Management Science*, 2:605–652, 1990.

W.B. Powell and H. Topaloglu. Stochastic programming in transportation and logistics. *Handbooks in Operations Research and Management Science*, 10:555–635, 2003.

W. Rei, M. Gendreau, and P. Soriano. A hybrid Monte Carlo local branching algorithm for the single vehicle routing problem with stochastic demands. *Transportation Science*, 44(1):136–146, 2010.

D. Revuz. *Markov chains*. North Holland, 2005.

A.E. Rizzoli, N. Casagrande, A.V. Donati, L.M. Gambardella, D. Lepori, R. Montemanni, P. Pina, and M. Zaffalon. Planning and optimization of vehicle routes for fuel oil distribution. In *Proceedings of MODSIM 2003 – Integrative Modeling of Biophysical, Social and Economic Systems for Resource Management Solutions*, volume 4, pages 2024–2029, 2003.

A.E. Rizzoli, R. Montemanni, E. Lucibello, and L.M. Gambardella. Ant colony optimisation for real world vehicle routing problems: From theory to applications. *Swarm Intelligence*, 1(2):135–151, 2007.

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

H.M. Salkin and C.A. De Kluyver. The knapsack problem: A survey. *Naval Research Logistics Quarterly*, 22(1):127–144, 1975.

P.A. Samuelson. Lifetime portfolio selection by dynamic stochastic programming. *The Review of Economics and Statistics*, 51(3):239–246, 1969.

M. Schatzman and J. Taylor. *Numerical analysis: A mathematical introduction*. Clarendon Press Oxford, 2002.

M. Schilde, K.F. Doerner, and R.F. Hartl. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers and Operations Research*, 38(12):1719–1730, 2011.

R. Schultz, L. Stougie, and M.H. Vlerk. Two-stage stochastic integer programming: A survey. *Statistica Neerlandica*, 50(3):404–416, 2008.

N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

N. Secomandi and F. Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations research*, 57(1):214–230, 2009.

A. Shapiro. Monte Carlo sampling methods. *Handbooks in Operations Research and Management Science*, 10:353–425, 2003.

D.J. Sheskin. *Handbook of parametric and nonparametric statistical procedures. 2004*. Chapman & Hall / CRC, Boca Raton, FL, 2004.

J. Shu, C.P. Teo, and Z.J.M. Shen. Stochastic transportation-inventory network design problem. *Operations Research*, 53(1):48–60, 2005.

J.R. Stedinger, B.F. Sule, and D.P. Loucks. Stochastic dynamic programming models for reservoir operation optimization. *Water Resources Research*, 20(11): 1499–1505, 1984.

W.R. Stewart and B.L. Golden. Stochastic vehicle routing: A comprehensive approach. *European Journal of Operational Research*, 14(4):371–385, 1983.

I. Sungur, F. Ordóñez, and M. Dessouky. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, 40(5):509–523, 2008.

H. Tang and E. Miller-Hooks. Approximate procedures for probabilistic traveling salesperson problem. *Transportation Research Record: Journal of the Transportation Research Board*, 1882(-1):27–36, 2004.

L. Trevisan. When hamming meets Euclid: The approximability of geometric TSP and steiner tree. *SIAM Journal on Computing*, 30(2):475–485, 2000.

S. Tsutsui and N. Fujimoto. Solving quadratic assignment problems by genetic algorithms with GPU computation: A case study. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2523–2530. ACM, 2009.

S. Uryasev and P.M. Pardalos. *Stochastic optimization: Algorithms and applications*. Springer, 2001.

T. Van Luong, N. Melab, and E.G. Talbi. Parallel local search on GPU. Technical Report RR-6915, INRIA, Institut National de Recherche en Informatique et en Automatique, 2009.

T. Van Luong, N. Melab, and E.G. Talbi. Large neighborhood local search optimization on graphics processing units. In *IEEE International Symposium on Parallel and Distributed Processing, Workshops and PhD Forum (IPDPSW), 2010*, pages 1–8, 2010.

R.J. Vander Wiel and N.V. Sahinidis. An exact solution approach for the time-dependent traveling-salesman problem. *Naval Research Logistics (NRL)*, 43(6): 797–820, 1996.

Z. Wang, H. Qiao, and K.J. Burnham. On stabilization of bilinear uncertain time-delay stochastic systems with markovian jumping parameters. *IEEE Transactions on Automatic Control*, 47(4):640–646, 2002.

Z. Wang, Y. Liu, and X. Liu. Exponential stabilization of a class of stochastic system with markovian jump parameters and mode-dependent mixed time-delays. *IEEE Transactions on Automatic Control*, 55(7):1656–1662, 2010.

D. Weyland, L. Bianchi, and L.M. Gambardella. New approximation-based local search algorithms for the probabilistic traveling salesman problem. In *Proceedings of the 12th International Conference on Computer Aided Systems Theory (Eurocast 2009)*, 2009a.

D. Weyland, L. Bianchi, and L.M. Gambardella. New heuristics for the probabilistic traveling salesman problem. In *Proceedings of the VIII Metaheuristic International Conference (MIC 2009)*, 2009b.

D. Weyland, R. Montemanni, and L.M. Gambardella. Hardness results for substantially stochastic vehicle routing problems. submitted, 2011a.

D. Weyland, R. Montemanni, and L.M. Gambardella. Using statistical tests for improving state-of-the-art heuristics for the probabilistic traveling salesman problem with deadlines. In *Proceedings of the 13th International Conference on Computer Aided Systems Theory (Eurocast 2011)*, volume 6927, pages 448–455, 2011b.

D. Weyland, R. Montemanni, and L.M. Gambardella. On the computational complexity of the probabilistic traveling salesman problem with deadlines. submitted, 2012a.

D. Weyland, R. Montemanni, and L.M. Gambardella. Heuristics for the probabilistic traveling salesman problem with deadlines based on quasi-parallel Monte Carlo sampling. *Computers and Operations Research*, to appear, 2012b.

D. Weyland, R. Montemanni, and L.M. Gambardella. A metaheuristic framework for stochastic combinatorial optimization problems based on GPGPU with a case study on the probabilistic traveling salesman problem with deadlines. *Journal of Parallel and Distributed Computing*, 2012c.

D. Weyland, R. Montemanni, and L.M. Gambardella. Hardness results for the probabilistic traveling salesman problem with deadlines. In *Proceedings of ISCO 2012 - The 2nd International Symposium on Combinatorial Optimization*, 2012d.

D. Weyland, R. Montemanni, and L.M. Gambardella. Convergence results for vehicle routing problems with stochastic demands. In *3rd Annual International Conference on Operations Research and Statistics (ORS 2013)*, 2013a.

D. Weyland, M. Salani, R. Montemanni, and L.M. Gambardella. Vehicle routing for exhausted oil collection. *Journal of Traffic and Logistics Engineering*, 1(1): 5–8, 2013b.

D.J. Wilkinson. Stochastic modelling for quantitative description of heterogeneous biological systems. *Nature Reviews Genetics*, 10(2):122–133, 2009.

T.H.F. Wong and N.L.G. Somes. A stochastic approach to designing wetlands for stormwater pollution control. *Water science and technology*, 32(1):145–152, 1995.

T.T. Wong and M. Wong. Parallel evolutionary algorithms on consumer-level graphics processing unit. *Parallel Evolutionary Computations*, pages 133–155, 2006.

C. Wu, R. Jeraj, G.H. Olivera, and T.R. Mackie. Re-optimization in adaptive radiotherapy. *Physics in medicine and biology*, 47(17):3181, 2002.

W.H. Yang, K. Mathur, and R.H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112, 2000.

F. You and I.E. Grossmann. Mixed-integer nonlinear programming models and algorithms for large-scale supply chain design with stochastic inventory management. *Industrial and Engineering Chemistry Research*, 47(20):7802–7817, 2008.

Y.S. Zheng. On properties of stochastic inventory systems. *Management Science*, 38(1):87–103, 1992.

X.Y. Zhou and D. Li. Continuous-time mean-variance portfolio selection: A stochastic LQ framework. *Applied Mathematics and Optimization*, 42(1):19–33, 2000.

W. Zhu and J. Curry. Parallel ant colony for nonlinear function optimization with graphics hardware acceleration. In *Systems, Man and Cybernetics, SMC 2009*, pages 1803–1808, 2009.