EURO Advanced Tutorials on Operational Research
Series Editors: M. Grazia Speranza · José Fernando Oliveira

Pieter Vansteenwegen
Aldy Gunawan

Orienteering Problems

Models and Algorithms for Vehicle Routing Problems with Profits

EURO
THE ASSOCIATION OF EUROPEAN OPERATIONAL RESEARCH SOCIETIES

Springer

EURO Advanced Tutorials on Operational Research

The EURO Advanced Tutorials on Operational Research are a series of short books devoted to an advanced topic—a topic that is not treated in depth in available textbooks. The series covers comprehensively all aspects of Operations Research. The scope of a Tutorial is to provide an understanding of an advanced topic to young researchers, such as Ph.D. students or Post-docs, but also to senior researchers and practitioners. Tutorials may be used as textbooks in graduate courses.

More information about this series at http://www.springer.com/series/13840

Pieter Vansteenwegen · Aldy Gunawan

Orienteering Problems

Models and Algorithms for Vehicle Routing Problems with Profits

Springer

Pieter Vansteenwegen
KU Leuven Mobility Research Centre
University of Leuven
Leuven, Belgium

Aldy Gunawan
School of Information Systems
Singapore Management University
Singapore

ISSN 2364-687X ISSN 2364-6888 (electronic)
EURO Advanced Tutorials on Operational Research
ISBN 978-3-030-29745-9 ISBN 978-3-030-29746-6 (eBook)
https://doi.org/10.1007/978-3-030-29746-6

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

We dedicate this tutorial to our families.
To Sofie, Sien, Adriaan, Lena, and Simon.
To Novi, Natasha, and Nicole.

Preface

When logistic or e-commerce companies distribute all kinds of products to their customers, they want to minimize their distribution costs. This is typically obtained by solving the so-called 'vehicle routing problems', where the objective is to minimize the total cost, distance, or time required to visit all customers.

This tutorial introduces readers to **routing problems with profits**. In these routing problems, each customer is modeled with a certain profit and not all customers need to be visited. Since the orienteering problem (OP) is by far the most-studied problem in this category of routing problems, the OP is the main focus of this book and the other problems are presented as variants of the OP, focusing on the similarities and differences between these variants. The goal of the OP is to determine the best subset of customers to visit, and in which order, so that the total profit is maximized and a given time budget is not exceeded.

This book clearly explains and illustrates, to both academics and practitioners, the available definitions and mathematical models for these routing problems with profits and introduces research opportunities for academic researchers. This book shows the readers that many practical problems can be modeled more appropriately as a routing problem with profits instead of as a regular routing problem. In addition, the book presents different scientific rigorous solution techniques for these OP variants. Solution techniques and mathematical models are accompanied by easy to understand examples and figures.

Plenty of examples facilitate the understanding of these routing problems. Several benchmark instances including their latest best known results are also included, to explain the complexity of solving these problems. Links to online benchmark instances, solvers, and detailed solutions are provided as well. In order to better understand the problems and their relevance, this tutorial also discusses a variety of applications from different fields such as logistics, tourism, and crowdsourcing.

This book discusses the OP, the team OP, the (team) OP with time windows, the profitable tour problem, and the prize-collecting traveling salesperson problem in detail. Other variants (with capacity constraints, time dependency, stochasticity, etc.) are discussed more briefly. We further limit this book to the so-called 'node

routing problems', where customers have a given location, and we do not consider the 'arc routing problems' with profits, where driving through an entire street is considered as a single customer. Obviously, we include references for readers who are interested in arc routing problems with profits, node routing variants or many other solution approaches we could not discuss in this book.

The book mainly aims for graduate students (Master's, MBA, or Ph.D.) in engineering, economics or applied mathematics, and obviously operations research. It is also suitable for advanced undergraduate students with some background in operations research. Moreover, practitioners and planning engineers in logistic companies will be inspired by understanding these alternative routing problems and the presented solution techniques.

Leuven, Belgium — Pieter Vansteenwegen
Singapore — Aldy Gunawan
May 2019

Acknowledgements

First, we want to acknowledge the help of several graduate students and researchers over the past years, extending our knowledge on routing problems with profits: Masoud Chitsaz, Ali Divsalar, Ander Garcia, Cedric Verbeeck, Guansheng Peng, Kun Lu, Truong Trong Nghia, and Audrey Tedja Widjaja. All of them have helped us in numerous ways. We would also like to thank Prof. Hoong Chuin Lau, Prof. Vincent F. Yu, Wouter Souffriau Ph.D., Prof. Reginald Dewil, Prof. Dirk Cattrysse, and Prof. Dirk Van Oudheusden as our collaborators for work related to the Orienteering Problem. We also wish to thank the reviewers for their many helpful suggestions. Special thanks goes to the editors and to the production staff at Springer. They have provided suggestions and comments resulting in this readable tutorial.

Contents

Acronyms

ABC	Artificial bee colony
ACO	Ant colony optimization
ACS	Ant colony system
ALNS	Adaptive large neighborhood search
AOP	Arc OP
CIRP	Cyclic IRP
ClOP	Clustered OP
COPTW	Cooperative OP with time windows
CTOP	Capacitated team orienteering problem
CTOP-IS	CTOP with incomplete service
CTPP	Cycle trip planning problem
DP	Dynamic programming
DSOP	Dynamic stochastic OP
DSSR	Decremental state space relaxation
ELS	Evolutionary local search
EOS	Earth observation satellite
ETG	Electronic tourist guides
GA	Genetic algorithm
GLS	Guided local search
GOP-RDR	Generalized OP with resource dependent rewards
GRASP	Greedy randomized adaptive search procedure
GVNS	Granular variable neighborhood search
IDCH	Iterative destruction/construction heuristic
ILS	Iterated local search
IOP	Inventory OP
IRP	Inventory routing problem
KP	Knapsack problem
LS	Local search
MA	Memetic algorithm
MDPVRP	Multi depot periodic vehicle routing problem

MILP-SAA	Mixed integer linear programming-sample average approximation
MOP-ND	Multi-commodity OP with network design
MuPOPMTW	Multi-period OP with multiple time windows
OARP	Orienteering arc routing problem
OP	Orienteering problem
OPHS	OP with hotel selection
OPmTW	Orienteering problem with multiple time windows
OPSP	OP with Stochastic Profits
OPSTS	OP with stochastic travel and service times
OPSW	OP with stochastic weights
OPSWTW	OP with stochastic weights and time windows
OP-TDR	OP with time dependent rewards
OPTW	Orienteering problem with time windows
OPVP	Orienteering problem with variable profits
PCTSP	Prize-collecting traveling salesperson problem
PET	Personalized electronic tourist guide
POI	Point of interest
PR	Path relinking
PSO	Particle swarm optimization
PSOiA	PSO-inspired algorithm
PTP	Profitable tour problem
RCESPP	Resource constrained elementary shortest path problem
RVNS	Reduced variable neighborhood search
SA	Simulated annealing
SAA	Sample average approximation
SACS	Stochastic ACS
SDCTOP-MDA	SDCTOP with minimum delivery amounts
SetOP	Set orienteering problem
SOPTW	Stochastic OP with time windows
SPM	Set-packing model
SV-CIRP	Single-vehicle CIRP
TDOP	Time dependent OP
TDOPTW	TDOP with time windows
TD-TOPTW	Time dependent TOPTW
TOARP	Team orienteering arc routing problem
TOP	Team orienteering problem
TOPTW	Team orienteering problem with time windows
TRPP	Traveling repairman problem with profits
TSP	Traveling salesperson problem
TSPP	Traveling salesperson problem with profits
TTDP	Tourist Trip Design Problem
VMI	Vendor Managed Inventory
VND	Variable neighborhood descent

VRP	Vehicle routing problem
VRPP	Vehicle routing problems with profits
VRPTW	Vehicle routing problem with time windows

Chapter 1
Introduction

The booming e-commerce and growing reverse logistic flows further increase the need for appropriate vehicle routing optimization, in order to efficiently meet customer demand. Obviously, distribution companies want to minimize their distribution costs. Therefore, they typically have routing software available in order to minimize the total cost, distance, and/or time required to visit all their customers on a daily basis. This routing software is almost always based on the basic vehicle routing problem (VRP) (with different extensions). The objective of the VRP is to minimize the number of vehicles and/or the total distance required to visit a fixed set of customers, starting from a depot. Each customer has a certain demand and the available vehicles have a limited capacity. Another famous routing problem is the traveling salesperson problem (TSP). In the TSP, the objective is to find the shortest (single) route visiting all customers.

This book shows the readers that many of these practical problems in logistics can be **modeled more appropriately as a routing problem with profits** instead of as a regular routing problem. Also tourist trip planning and, for instance, satellite scheduling and crowdsourcing problems are modeled adequately by routing problems with profits. As soon as all customers or points of interest or nodes have a certain profit and not all of them need to be visited, but a selection has to be made, routing problems with profits, in different variations, become relevant.

1.1 Classification

Routing problems with profits can be classified in two ways. One way is based on the number of vehicles or routes: problems with a single vehicle and problems with multiple vehicles. The second way is based on the manner in which the profit and the travel cost, mostly distance or time, are modeled. For the single vehicle routing

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_1

problems three variants are defined according to the second classification. The first one, called the profitable tour problem (PTP), combines both profit and travel cost in the objective function. Therefore, the objective of the PTP is to visit a subset of customers that maximizes the total collected profit minus the total travel cost. Obviously, both parts of the objective can be weighted together in any way. In the second problem, called the prize-collecting traveling salesperson problem (PCTSP), the objective is to minimize the total travel cost to collect at least a certain amount of profit by visiting a subset of the customers. So in this case, the profit is included as a constraint and minimizing the travel cost is the objective. In the third problem, called the orienteering problem (OP), it is the other way around. The objective is to maximize the total collected profit by visiting a subset of customers, while not exceeding a given travel cost, typically a time constraint. For all three problems a subset of customers with profits is selected and the order in which these are visited should be decided as well. The differences between these three problems (and with the regular routing problems) will be discussed later in this book, together with the implications of those differences on the mathematical models, and the solution approaches.

> Since the OP is by far the most studied problem of the routing problems with profits, the OP will be the main focus of this book and the other problems will be presented as variants of the OP, focusing on their similarities and differences.

Obviously these three categories of the second classification could also be considered with multiple vehicles. However, in literature, only the third one, the OP, appears with multiple vehicles. It is called the team orienteering problem (TOP). Due to the numerous practical applications where customers or tourist attractions have limited opening hours, the (team) orienteering problem with time windows (TOPTW) will also be discussed in detail. Also in the state-of-the-art, the TOPTW is the most studied variant of the OP.

At first sight, the OP might not seem so different from regular routing problems such as the vehicle routing problem (VRP) or the traveling salesperson problem (TSP). In all these problems, the route between the customers needs to be determined. However, two crucial differences should be considered, which make solving an OP significantly different and more time-consuming. First of all, a selection of customers has to be made in the OP. Making this selection is actually closely related to the well-known knapsack problem (KP). In the KP, each item has a profit and requires some volume. The goal is to determine the combination of items that maximizes the total profit and that fits in a given volume. In the OP, this corresponds to customers each having a certain profit and requiring some time to be visited. However, this time to visit a customer depends on which other customers are selected and in which order. This makes clear that the OP actually integrates the difficulties of the KP and the VRP, which are both already quite complex problems on their own. Secondly, the objective is different. This is obviously related to the first difference, but it also implies that finding the shortest route in the OP is not an objective in itself. It is a

means that could help to making time in order to potentially visit an extra customer or a customer with a higher profit instead of a current customer. So, different solutions with the same profit but a different travel cost are considered equally good, as long as they both respect the available travel cost.

1.2 History of Orienteering Problem

The name of the orienteering problem originates from a sport game called 'orienteering'. In this game, every checkpoint has a certain score and the goal for each participant is to maximize the total collected score. All participants start from a specified starting point, try to collect as much score as possible by visiting checkpoints at most once and return to the control point within a given time limit. That also explains the name of the multiple vehicle variant, the team orienteering problem, where members of the same team could run around individually and collect profit for the team at different checkpoints.

Currently the most used name for this problem is the orienteering problem, but exactly the same problem was also called the selective traveling salesperson problem, the maximum collection problem, and the bank robber problem. Although the problem was defined in 1984, and researched since then, the OP-related research started booming around the year 2010.

1.3 Relevance of Orienteering Problems

Orienteering problems arise in several applications, many of which are currently modeled as a regular VRP or TSP. In practice, this often means that a selection of the customers to service (that day) was already made. By using the standard orienteering problem (OP) as a model, this selection could be integrated and optimized as well. Obviously, also many specific and diverse applications of the OP have been described in the literature. Three random examples are given here. (1) The home fuel delivery problem focuses on sending a fleet of vehicles (trucks) to a large number of customers. Due to the time budget constraint, only a subset of customers can be visited each day. The selection of the subset depends on the urgency of the customers fuel inventory level. (2) The online matching of demand for transportation and services offered is considered by the capacitated team orienteering problem taking the capacity of each vehicle into account. (3) Vehicle routing, inventory management, and customer selection are integrated in the Single-Vehicle Cyclic Inventory Routing Problem. This problem is also considered as the Inventory OP, the inventory routing variant of the OP. Obviously, also the booming OP research since 2010 is an indication of its relevance.

Of the recent developments in modeling and solving the OP, TOP, OPTW, and TOPTW, most are probably linked to applications in tourism. The central idea is that a number of points of interest (POIs) are available in a given city or region, each with a different personal profit or satisfaction for a tourist. The OP is then used to model the problem of selecting the best set of attractions to visit, and in which order, given the limited available time of the tourist. Only the most basic 'tourist trip design problem' can be modeled with the regular OP, TOP, OPTW, or TOPTW. Many more trip design problems from practice require additional sets of constraints and/or different objective functions in order to model budget limitations, weather-dependent profits, combinations of attractions increasing or decreasing the sum of the profits, congestion, etc.

1.4 Outline

In Chap. 2, a formal definition and mathematical model are presented for each of the three above mentioned vehicle routing problems with profits and one vehicle: the profitable tour problem (PTP), the prize-collecting TSP (PCTSP), and the orienteering problem (OP). Based on easy-to-understand examples we focus on the differences between these problems. Chapter 3 discusses the formal definitions and mathematical models of the TOP, the OPTW, and the TOPTW. In Chap. 4, the state-of-the-art solution approaches for the PTP and the PCTSP are presented and discussed. Chapter 5 extensively discusses a selection of exact, metaheuristic, and hybrid solution approaches for the OP and TOP and focuses on the typical local search moves that allow to improve an existing solution in case of routing problems with profits. In Chap. 6, the same topics are discussed, but for OPs with time windows and with a focus on efficient techniques to deal with time windows in case of routing problems with profits. Chapter 7 discusses in detail a variety of applications that require modelling based on orienteering problems. In Chap. 8, a number of other OP variant, with capacity constraints, time dependency, stochasticity, etc., are presented together with relevant references to other work for each of them.

Arc routing problems with profits (where driving through an entire street is considered as a single customer) will not be considered in this book, which is limited to so-called node routing problems (where customers have a given location).

1.5 Related Literature

Reviews on routing problems with profits can be found in Feillet et al. (2005) and Archetti et al. (2014b).

More details on the sport game of orienteering can be found in Chao et al. (1996) and other names for the orienteering problems can be found in, e.g., Laporte and

Martello (1990), Gendreau et al. (1998), Thomadsen and Stidsen (2003), Kataoka and Morito (1988), and Butt and Cavalier (1994).

Applications of OPs can be found in, e.g., Golden et al. (1987), Archetti et al. (2009), and Vansteenwegen and Mateo (2014).

We refer the reader interested in arc routing problems with profits to Archetti et al. (2014a), Archetti and Speranza (2014), and Verbeeck et al. (2014).

References

Archetti C, Speranza MG (2014) Arc routing problems with profits. In: Corberán Á, Laporte G (eds) Arc routing: problems, methods, and applications. MOS-SIAM series on optimization. SIAM, Philadelphia, pp 281–300

Archetti C, Feillet D, Hertz A, Speranza MG (2009) The capacitated team orienteering and profitable tour problems. J Oper Res Soc 60:831–842

Archetti C, Speranza MG, Corberán Á, Sanchis JM, Plana I (2014a) The team orienteering arc routing problem. Transp Sci 48(3):442–457

Archetti C, Speranza MG, Vigo D (2014b) Vehicle routing problems with profits. In: Toth P, Vigo D (eds) Vehicle routing: problems, methods, and applications. MOS-SIAM series on optimization. SIAM, Philadelphia, pp 273–298

Butt SE, Cavalier T (1994) A heuristic for the multiple tour maximum collection problem. Comput Oper Res 21(1):101–111

Chao IM, Golden BL, Wasil EA (1996) A fast and effective heuristic for the orienteering problem. Eur J Oper Res 88(3):475–489

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transp Sci 39(2):188–205

Gendreau M, Laporte G, Semet F (1998) A tabu search heuristic for the undirected selective travelling salesman problem. Eur J Oper Res 106(2–3):539–545

Golden BL, Levy L, Vohra R (1987) The orienteering problem. Naval Res Logist 34(3):307–318

Kataoka S, Morito S (1988) An algorithm for the single constraint maximum collection problem. J Oper Res Soc Jpn 31(4):515–530

Laporte G, Martello S (1990) The selective travelling salesman problem. Discret Appl Math 26(2–3):193–207

Thomadsen T, Stidsen T (2003) The quadratic selective travelling salesman problem. Informatics and mathematical modelling technical report IMM-Technical report-2003–17, Technical University of Denmark

Vansteenwegen P, Mateo M (2014) An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. Eur J Oper Res 237(3):802–813

Verbeeck C, Vansteenwegen P, Aghezzaf EH (2014) An extension of the arc orienteering problem and its application to cycle trip planning. Transp Res Part E 68:64–78

Chapter 2
Definitions and Mathematical Models of Single Vehicle Routing Problems with Profits

In this chapter, single vehicle routing problems with profits are introduced and defined. Three variants are considered: the profitable tour problem, the prize-collecting traveling salesperson problem, and the orienteering problem. The difference between these variants is the way in which the profit and the travel cost, mostly distance or time, are modeled. Profit and travel cost can be modeled as (part of) the objective or as a constraint. All three problems differ from the well-known traveling salesperson problem, for which the only objective is to find the shortest route to visit all customers in a given set. In vehicle routing problems with profits, some customers will be selected to be visited and others will not, based on the objective considered.

The example network, illustrated in Fig. 2.1, will be used to explain the difference between these variants.

In this network, a depot (with index 1) is connected with four customers or nodes (with indices 2, 3, 4, and 5). Each customer has a certain profit P: 5 for customer 2, 8 for customer 3, 1 for customer 4, and 4 for customer 5. A symmetric travel cost, related to the distance, is indicated for each link. Visiting all the customers requires a cost of at least 17, in the order 1-2-3-4-5-1 (or 1-5-4-3-2-1). The profit of visiting all customers is $5 + 8 + 1 + 4 = 18$.

In this chapter, each single vehicle variant will be defined and an appropriate mathematical model will be presented, together with a discussion on the differences and similarities with the other variants. First, the profitable tour problem is discussed, then the prize-collecting traveling salesperson problem and finally the orienteering problem.

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_2

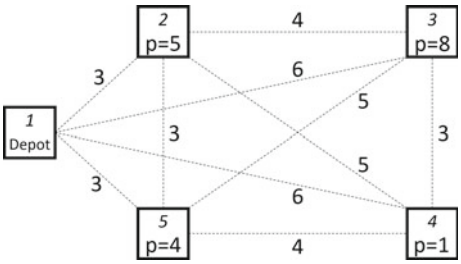

Fig. 2.1 Network with a depot, four customers with profits P and travel costs on all links

2.1 Profitable Tour Problem

In the profitable tour problem (PTP), the profit and the travel cost are both considered in the objective function. Therefore, the objective is to maximize the collected profit minus the travel cost. Obviously, this only makes sense if both are expressed in the same units, otherwise a conversion is required first. Both objectives can also be weighted differently. This also implies that no limits are imposed on the profit or the travel cost. In Fig. 2.2, an example solution is presented where customers 3 and 5 are visited. For this example, the collected profit is 12 and the total travel cost is $6 + 5 + 3 = 14$, which gives a result of -2. Obviously, staying in the depot, with a result of 0 is better than visiting customers 3 and 5.

The best solution for this example network is illustrated in Fig. 2.3, visiting customers 2, 3, and 5 with a profit of $5 + 8 + 4 = 17$, a travel cost of $3 + 4 + 5 + 3 = 15$ and a result of 2. In the optimal solution, the route visiting the selected customers will always correspond to the TSP route between these selected customers.

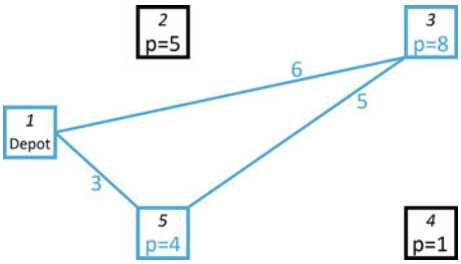

Fig. 2.2 Example solution for the PTP, visiting customers 3 and 5

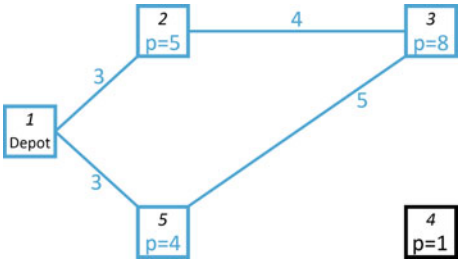

Fig. 2.3 Optimal PTP solution for the example network

The differences between the PTP and the regular TSP are obvious. In the TSP, all customers need to be visited, so no selection is required and no profits are considered. In the PTP, a customer is selected based on the trade-off between its profit and the extra travel cost required to include the customer. The most interesting customers have a high profit and require little extra travel cost.

2.1.1 Formal Definition and Mathematical Model PTP

The PTP is defined as follows. Consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. For the ease of modeling, a profit $P_i = 0$ is associated with nodes 1 and $|N|$. Nodes 1 and $|N|$ can also represent a single depot, when both are on the same location. This is the case in the example network we use throughout this chapter. The non-negative travel cost between nodes i and j is represented as t_{ij}. The goal of the PTP is to determine a route, starting at node 1 and ending at node $|N|$ that visits a subset of N and maximizes the total collected profit minus the total travel cost. It is assumed that travel costs and collected profits can be added and that each node can be visited at most once.

The PTP can be formulated as an integer programming model with the following decision variables: $x_{ij} = 1$ if a visit to node i is followed by a visit to node j, and 0 otherwise. The variables u_i will be used in the subtour elimination constraints and allow to determine the position of the visited nodes in the route.

$$\text{Maximize} \sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} (P_i x_{ij} - t_{ij} x_{ij}) \tag{2.1}$$

The objective function (2.1) is to maximize the total collected profit minus the total travel cost.

$$\sum_{j=2}^{|N|} x_{1j} = \sum_{i=1}^{|N|-1} x_{i|N|} = 1 \tag{2.2}$$

Constraints (2.2) ensure that the route starts from node 1 and ends on $|N|$.

$$\sum_{i=1}^{|N|-1} x_{ik} = \sum_{j=2}^{|N|} x_{kj} \leq 1; \ \forall k = 2, \ldots, (|N| - 1) \tag{2.3}$$

Constraints (2.3) ensure the connectivity of the route and guarantee that each node is visited at most once.

$$2 \leq u_i \leq |N|; \ \forall i = 2, \ldots, |N| \tag{2.4}$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - x_{ij}); \ \forall i, j = 2, \ldots, |N| \tag{2.5}$$

The combination of constraints (2.4) and (2.5) prevents subtours, based on the Miller–Tucker–Zemlin formulation. Constraints (2.2), (2.3), (2.4), and (2.5) are also present in the mathematical formulation of the two other routing problems with profits.

2.1.2 Subtour Elimination

A common issue in many routing problems is that incomplete mathematical formulations might allow subtours to appear in a solution. Without constraints (2.4) and (2.5), the combination of routes 1-5-1 and 2-3-4-2 would be an acceptable solution. This solution is illustrated in Fig. 2.4. This example will illustrate how, for any possible subtour, the combination of constraints (2.4) and (2.5) will be violated. Consider the subtour between nodes 2, 3, and 4. This subtour implies that $x_{23} = x_{34} = x_{42} = 1$. With these values for x_{ij}, constraints (2.5) result in:

$$u_2 - u_3 + 1 \leq 0; \tag{2.6}$$

$$u_3 - u_4 + 1 \leq 0; \tag{2.7}$$

$$u_4 - u_2 + 1 \leq 0. \tag{2.8}$$

Adding up the left and right sides of these constraints leads to:

$$3 \leq 0. \tag{2.9}$$

This is obviously not correct, indicating constraints (2.5) will always be violated with these values for x_{ij}.

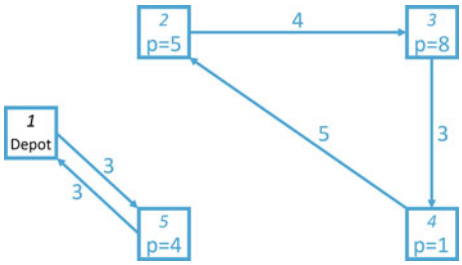

Fig. 2.4 Unacceptable PTP solution with two subtours

An alternative formulation to avoid subtours indicates that for every subset of nodes (not containing the depot), the number of arcs inside the subset (x_{ij}-values equal to 1) should be strictly smaller than the number of nodes in the subset.

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1; \forall S \subset N \setminus \{1\} \tag{2.10}$$

This constraint is violated in the example of Fig. 2.4, where the subset of nodes $S = \{2, 3, 4\}$ contains 3 arcs for 3 nodes.

Another alternative formulation to avoid subtours states that each possible subset should have at least one *outgoing arc*. This is an arc starting from a node inside the subset and ending at a node outside the subset. Although this formulation is closely related to the previous alternative, it cannot be used in case of vehicle routing problems with profits. In vehicle routing problems with profits, some nodes might not be selected and will thus have no outgoing arc. Also subsets of unselected nodes will have no outgoing arc.

2.1.3 Complexity

It is proven that the PTP is NP-hard. Therefore, no polynomial time algorithm has been designed, or is expected to be designed, to solve this problem to optimality. This implies that exact solution algorithms are very time-consuming and for practical applications heuristics will be mostly required. Solution approaches for the PTP are discussed in Chap. 4.

When looking back at the example network, it can be concluded that in this example customer 4 will never be selected. The extra travel cost to include this customer will always be larger than its profit of 1. In general, it will simplify the problem when some customers can be excluded from consideration. However, this should be done with care. It is not because all links towards a certain customer have a cost that is higher than its profit that it should be excluded. Only if the *extra* travel cost of adding this customer to any possible solution is larger than the profit, the customer should be excluded. Suppose the example is changed and customer 4 would get a profit of 3 instead of 1, the optimal solution will include customer 4. A solution with all customers would then have a profit of $5 + 8 + 3 + 4 = 20$, a travel cost of $3 + 4 + 3 + 4 + 3 = 17$ and thus a result of 3. The solution without customer 4 has a result of only 2, see Fig. 2.3.

Although the number of possible solutions for a certain problem is not necessarily directly related to the computational effort required to solve the problem, it might help to better understand the complexity of the problem. It also helps to compare the

complexity of the regular TSP with the PTP, or actually any vehicle routing problem with profits.

Consider a network with p ($= |N| - 2$) real customers. In order to simplify matters, we exclude the fixed starting and ending depot from this analysis. The number of possible solutions (possible sequences) for the regular TSP then corresponds to $p!$. In the vehicle routing problems with profits, however, also a subset of customers needs to be selected. The total number of subsets equals:

$$2^p. \tag{2.11}$$

The number of possible subsets with k customers (excluding the depot) out of p possible customers equals:

$$\binom{p}{k} = \frac{p!}{k!(p-k)!}. \tag{2.12}$$

For example, for a problem with 50 customers, $1.26\ 10^{14}$ subsets with 25 customers are possible. In order to determine the optimal route between k customers, $k!$ different routes should be evaluated. For 25 customers, this means $1.55\ 10^{25}$ possible routes.

In order to combine the selection problem and TSP problem, all possible routes should be calculated for each possible selection. This defines the number of possible solutions for a vehicle routing with profits with p customers locations as:

$$\sum_{i=0}^{p} \frac{p!}{(p-i)!}. \tag{2.13}$$

A problem with 50 customers has $8.27\ 10^{64}$ possible solutions. A smaller problem with only 20 customers still has $6.61\ 10^{18}$ possible solutions. If a computer can evaluate one billion solutions per second, it still needs 210 years of calculation to evaluate all the possible solutions.

2.2 Prize-Collecting Traveling Salesperson Problem

In the prize-collecting traveling salesperson problem (PCTSP) the profit is stated as a constraint. The objective is to find a route that minimizes the total travel cost and collects at least a given amount of profit. If the minimal profit is set to 10, a feasible solution, visiting customers 3 and 5, is presented in Fig. 2.5. For this example, the collected profit is 12 (>10) and the total travel cost is $6 + 5 + 3 = 14$.

The best solution for this example network is illustrated in Fig. 2.6, visiting customers 2 and 3 with a profit of $5 + 8 = 13$ and a travel cost of $3 + 4 + 6 = 13$. In the

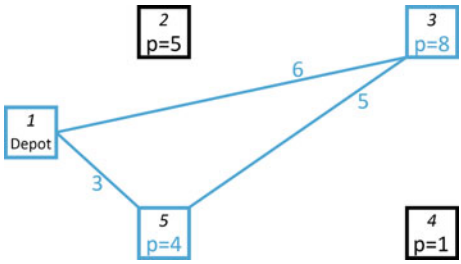

Fig. 2.5 Example solution for the PCTSP, visiting customers 3 and 5

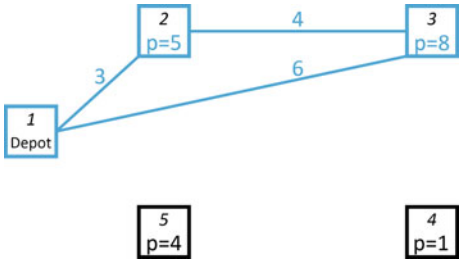

Fig. 2.6 Optimal PCTSP solution for the example network, with a minimal profit constraint of 10

optimal solution, the route visiting the selected customers will always correspond to the TSP route between these selected customers.

The difference between the PCTSP and the TSP is that not all customers need to be visited in the PCTSP and that a minimal profit needs to be collected. The difference between the PCTSP and the PTP is that the profit collection is stated as a constraint in the PCTSP and not as a part of the objective function.

> In many practical applications, visiting a customer also requires time, typically called service time. When the travel cost corresponds to time and the objective is to minimize the total time for collecting a certain profit, these service times should be considered as well. A straightforward way to consider the service times is to include these in the travel cost. The travel cost between two customers will then be composed of the actual travel time between these customers and the service time required by the first customer. For the rest of this chapter and also for the mathematical models, we will assume that service times are included in the travel cost.

2.2.1 Formal Definition and Mathematical Model PCTSP

The PCTSP is defined as follows. Consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. As with the PTP, nodes 1 and $|N|$ can also represent a single depot, when both are on the same location, and a profit

$P_i = 0$ is associated with both nodes. The goal of the PCTSP is to determine a route that visits a subset of N, collecting at least a given profit P_{min} and minimizing the total travel cost. It is assumed that collected profits can be added and that each node can be visited at most once. The non-negative travel cost between nodes i and j is represented as t_{ij}.

The PCTSP can be formulated as an integer programming model with the following decision variables: $x_{ij} = 1$ if a visit to node i is followed by a visit to node j, and 0 otherwise. The variables u_i will be used in the subtour elimination constraints and allow to determine the position of the visited nodes in the route.

Constraints (2.15), (2.16), (2.18), and (2.19) are the same as for the PTP formulation.

$$\text{Minimize} \sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} t_{ij} x_{ij} \tag{2.14}$$

The objective function (2.14) is to minimize the total travel cost.

$$\sum_{j=2}^{|N|} x_{1j} = \sum_{i=1}^{|N|-1} x_{i|N|} = 1 \tag{2.15}$$

Constraints (2.15) ensure that the route starts from node 1 and ends on $|N|$.

$$\sum_{i=1}^{|N|-1} x_{ik} = \sum_{j=2}^{|N|} x_{kj} \leq 1; \ \forall k = 2, \ldots, (|N| - 1) \tag{2.16}$$

Constraints (2.16) ensure the connectivity of the route and guarantee that each node is visited at most once.

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} P_i x_{ij} \geq P_{min} \tag{2.17}$$

Constraint (2.17) sets a minimum value P_{min} to the profit collected.

$$2 \leq u_i \leq |N|; \ \forall i = 2, \ldots, |N| \tag{2.18}$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - x_{ij}); \ \forall i, j = 2, \ldots, |N| \tag{2.19}$$

The combination of constraints (2.18) and (2.19) prevents subtours.

2.2.2 Complexity

It is proven that the PCTSP is NP-hard. Therefore, no polynomial time algorithm has been designed, or is expected to be designed, to solve this problem to optimality. This implies that exact solution algorithms are very time-consuming and for practical applications heuristics will be mostly required. Determining a solution requires the integration of selecting the most appropriate subset of customers, satisfying the profit constraint, and determining the shortest route to visit all customers in that subset, which corresponds to solving a TSP for the subset.

If a relatively low profit needs to be collected, only a few customers will need to be visited and determining the TSP-route will be easy. When a very high profit needs to be collected, only a limited number of subsets will meet this constraint, but solving the TSP in the considered subsets might require some computation time. Intuitively, the most difficult PCTSP to solve are the instances where little more than half of the customers need to be visited to gather the required profit. In that case, many different subsets will need to be considered and solving the TSP for each of the subsets will require some calculation time. Solution approaches for the PCTSP are discussed in Chap. 4.

2.3 Orienteering Problem

In the orienteering problem (OP) the travel cost is limited by a constraint. The objective is to find a route that maximizes the total collected profit while respecting the travel cost constraint. If the travel cost is limited to 15, a feasible solution, visiting customers 3 and 4, is presented in Fig. 2.7. For this example, the collected profit is 9 and the total travel cost is $6 + 3 + 6 = 15$ ($\leq$15).

The best solution for this example network is illustrated in Fig. 2.8, visiting customers 2, 3, and 5 with a profit of $5 + 8 + 4 = 17$ and a travel cost of $3 + 4 + 5 + 3 = 15$ ($\leq$15). Important to notice here is that in the optimal solution, the route visiting the selected customers does not have to correspond to the TSP route between these selected customers, since the travel cost is not a part of the objective function. Nevertheless, reducing the travel cost for the current set of selected customers could allow visiting additional customers and further increase the profit.

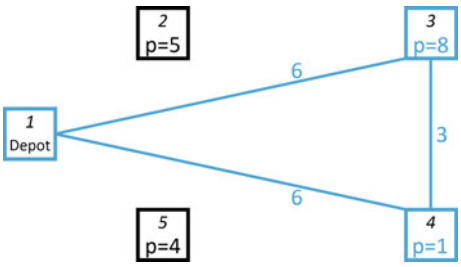

Fig. 2.7 Example solution for the OP, visiting customers 3 and 4

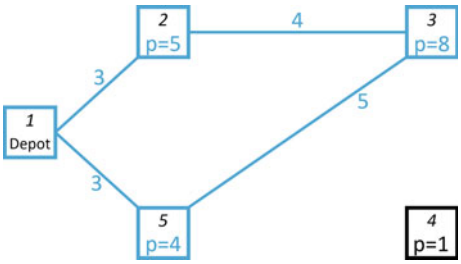

Fig. 2.8 Optimal OP solution for the example network, with a maximum travel cost constraint of 15

The name orienteering problem originates from the sport game of orienteering. In this game, individual competitors start at a specified control point, try to visit as many checkpoints as possible and return to the control point within a given time frame. Each checkpoint has a certain score and the objective is to maximize the total collected score.

The difference between the OP and the TSP is that not all customers need to be visited in the OP and that the travel cost is not a part of the objective function but a maximum travel cost is stated as a constraint. The only objective in the OP is to maximize the collected profit. The difference with the PTP is that travel cost is stated as a constraint in the OP and not as a part of the objective function. Compared to the PCTSP, the profit constraint and the travel cost objective have now changed to a profit objective and a travel cost constraint in the OP. As with the PCTSP, service times at customers can easily be included in the considered travel cost between customers. In the mathematical model below, we assume that service times are included in the travel cost.

> The OP is by far the most studied single vehicle routing problem with profits. Regardless a few exceptions, it is also the only vehicle routing problem with profits where multiple vehicles are considered. That is also why the rest of this book mainly focuses on the OP and why vehicle routing problems with profits are sometimes called 'orienteering problems'.

2.3.1 Formal Definition and Mathematical Model OP

The OP is defined as follows. Consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. As with the PTP and the PCTSP, nodes 1 and $|N|$ can also represent a single depot and a profit $P_i = 0$ is associated with both nodes. The goal of the OP is to determine a route that visits a subset of N, limited by a given travel cost budget T_{max} and maximizing the total collected profit. It is assumed that collected profits can be added and that each node can be visited at most once. The non-negative travel cost between nodes i and j is represented as t_{ij}.

The OP can be formulated as an integer programming model with the following decision variables: $x_{ij} = 1$ if a visit to node i is followed by a visit to node j, and 0 otherwise. The variables u_i will be used in the subtour elimination constraints and allow to determine the position of the visited nodes in the route.

Constraints (2.21), (2.22), (2.24), and (2.25) are the same as for the PTP and PCTSP formulation.

$$\text{Maximize} \sum_{i=2}^{|N|-1} \sum_{j=2}^{|N|} P_i x_{ij} \tag{2.20}$$

The objective function (2.20) is to maximize the total collected profit.

$$\sum_{j=2}^{|N|} x_{1j} = \sum_{i=1}^{|N|-1} x_{i|N|} = 1 \tag{2.21}$$

Constraints (2.21) ensure that the route starts from node 1 and ends on $|N|$.

$$\sum_{i=1}^{|N|-1} x_{ik} = \sum_{j=2}^{|N|} x_{kj} \leq 1; \ \forall k = 2, \ldots, (|N| - 1) \tag{2.22}$$

Constraints (2.22) ensure the connectivity of the route and guarantee that each node is visited at most once.

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} t_{ij} x_{ij} \leq T_{max} \tag{2.23}$$

Constraint (2.23) limits the total travel time within the time budget T_{max}.

$$2 \leq u_i \leq |N|; \ \forall i = 2, \ldots, |N| \tag{2.24}$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - x_{ij}); \ \forall i, j = 2, \ldots, |N| \tag{2.25}$$

The combination of constraints (2.24) and (2.25) prevents subtours.

2.3.2 Complexity

It is proven that the OP is NP-hard. Therefore, no polynomial time algorithm has been designed, or is expected to be designed, to solve this problem to optimality. This implies that exact solution algorithms are very time-consuming and for practical applications heuristics will be mostly required. Solution approaches for the OP are discussed in Chap. 4. Solving the OP actually corresponds to solving two well-

known optimization problems in a integrated way. The first problem is the Knapsack Problem, when selecting the most appropriate subset of customers in order to maximize the collected profit. The second problem is the TSP for the selected subset of customers. Although finding the route with minimal travel cost between the selected customers is not an explicit part of the objective, it might help to save travel cost which can be used to visit an additional customer and collect more profit. It is obvious that customers with a low profit and requiring a lot of *extra* travel cost should be considered as less interesting. *Extra* travel cost in this case corresponds to being located far away from the depot or far away from a route under construction. However, even these less interesting customers might be selected in the end to increase the profit of a solution under consideration.

For solution techniques that apply some kind of enumeration, such as many metaheuristics or branch-and-bound methods, the required calculation time to solve the problem will be proportional to the number of 'non-dominated' solutions. Non-dominated solutions for the OP are defined as solutions that consume (almost) the complete travel cost. If it is possible to add an extra customer to a certain solution, the new solution will always have a higher profit and it will 'dominate' the former solution. The exact number of non-dominated solutions is problem specific and difficult to be determined exactly. Nevertheless, it remains possible to get an idea about the type of problems that will require the longest calculation time. Indeed, it can be assumed that the travel cost constraint is an indication of the number of customers k that can be selected. Most non-dominated solutions will have (almost) k customers. For a given problem, Eq. (2.12) will be maximized if the value of k is half the value of p.

> In other words, if the travel cost allows the selection of half of the customers, the highest number of non-dominated selections will have to be evaluated by the algorithm. Moreover, determining the TSP route between the selected customers becomes more complicated when the number of customers increases. Therefore, the most difficult OP to solve by any kind of enumeration technique, are those problems where the selected number of customers is a little more than half the total number of customers.

2.4 Related Literature

Reviews on routing problems with profits can be found in Feillet et al. (2005) and Archetti et al. (2014). Surveys on the orienteering problem (and many variants) can be found in Vansteenwegen et al. (2011) and Gunawan et al. (2016). Various applications of OPs can be found in, e.g., Golden et al. (1987), Archetti et al. (2009), and Vansteenwegen and Mateo (2014). More information on subtour elimination constraints can be found in Miller et al. (1960) and Achuthan et al. (1996).

References

Achuthan N, Caccetta L, Hill S (1996) A new subtour elimination constraint for the vehicle routing problem. Eur J Oper Res 91(3):573–586

Archetti C, Feillet D, Hertz A, Speranza MG (2009) The capacitated team orienteering and profitable tour problems. J Oper Res Soc 60:831–842

Archetti C, Speranza MG, Vigo D (2014) Vehicle routing problems with profits. In: Toth P, Vigo D (eds) Vehicle routing: problems, methods, and applications. MOS-SIAM series on optimization. SIAM, Philadelphia, pp 273–298

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transp Sci 39(2):188–205

Golden BL, Levy L, Vohra R (1987) The orienteering problem. Naval Res Logist 34(3):307–318

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332

Miller C, Tucker A, Zemlin R (1960) Integer programming formulations and travelling salesman problems. J ACM 7:326–329

Vansteenwegen P, Mateo M (2014) An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. Eur J Oper Res 237(3):802–813

Vansteenwegen P, Souffriau W, Van Oudheusden D (2011) The orienteering problem: a survey. Eur J Oper Res 209(1):1–10

Chapter 3
Definitions and Mathematical Models of OP Variants

We have described the basic orienteering problem (OP) in Chap. 2, as one known variant of the single vehicle routing problems with profits (VRPP). In this chapter, we introduce the best-known variants of the OP. The first one is the team orienteering problem (TOP). In the TOP multiple routes can be composed to visit a subset of customers. In the context of the game of orienteering, the TOP corresponds to several players of the same team, each collecting profits in parallel, during the same time span.

In this chapter, we will use the same example network from the previous chapter, illustrated in Fig. 3.1, to explain the TOP in Sect. 3.1. In this network, four customers or nodes (with indices 2, 3, 4, and 5) are connected to a depot with index 1. When all customers can be visited, the collected profit is 18. However, we have seen in the previous chapter that if the problem is solved as an OP with a maximal travel cost of 15, the total collected profit can be at most 17 (see Fig. 2.8). Obviously, if more routes are allowed, we should be able to obtain a higher profit. This will be further explained in Sect. 3.1).

Another well-known variant of the basic OP is the OP with time windows (OPTW), imposing time windows for each customer. If we consider, for example, an application in tourism, we should account for the fact that most tourist attractions are only open within certain opening hours. Arriving at an attraction after it has closed or long before it will open, should be avoided. Both variants, the TOP and the OPTW come together in the third well-known variant, the team OP with time windows (TOPTW), where time windows are considered together with multiple vehicles (or a multiple day trip).

Figure 3.2 illustrates an example network for the OPTW. Each node is limited by its time windows. For example, the start of the service at node 2 can only be started between time 2 and time 5. The time windows of the depot indicate that the entire travel time is bounded to 15 time units. Due to the time windows considered in this chapter, we will use from here on both 'travel time' and 'travel cost' interchangeably.

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_3

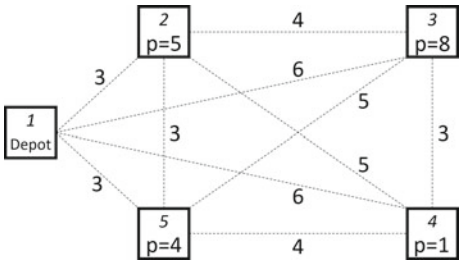

Fig. 3.1 Network with a depot, four customers with profits P and travel costs on all links

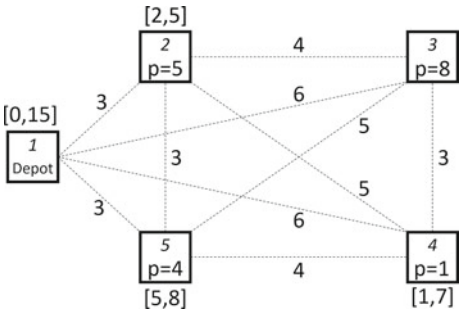

Fig. 3.2 Network with a depot, four customers with profits and time windows and travel times on all links

The TOP, OPTW, and TOPTW will be presented in detail in this chapter, together with an appropriate mathematical model. Other variants and extensions of the OP, modeling more complex and complicated optimization problems from practice, will be explained in Chap. 8.

3.1 Team Orienteering Problem

The extension of the OP to multiple routes is defined as the team orienteering problem (TOP) with the objective of maximizing the collected profit across multiple routes. Figure 3.3 illustrates a possible TOP solution for the example network considered. The collected profit is $9 + 8 = 17$ from the two routes together. Each route is bounded by a time budget of 15 time units. The first route has a travel time of $3 + 3 + 3 = 9$ while the second one has a travel time of $6 + 6 = 12$ time units. The second route only visits node 3.

The optimal TOP solution for this example is shown in Fig. 3.4. In this small example, all nodes can be visited when a time budget of 15 is available for each route. Obviously, with a smaller time budget or a larger instance, not all nodes can be visited. The collected profit is 18 in this case. The first route visits nodes 2 and 5 with a profit of $5 + 4 = 9$ and a travel time of $3 + 3 + 3 = 9$. The second route visits nodes 3 and 4 with a profit of $8 + 1 = 9$ and a travel time of $6 + 6 + 3 = 15$.

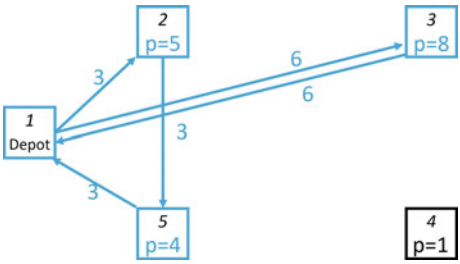

Fig. 3.3 Example solution for the TOP, visiting customers 2, 3, and 5

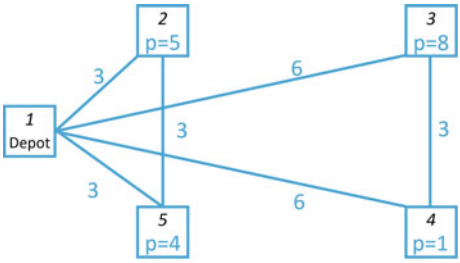

Fig. 3.4 Optimal TOP solution for the example network

3.1.1 Formal Definition and Mathematical Model TOP

As with the OP, consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. As with the OP, nodes 1 and $|N|$ can also represent a single depot and a profit $P_i = 0$ is associated with both nodes. The goal of the TOP is to determine M routes, each limited by T_{max}, that visit a subset of N and that maximize the total collected profit. It is assumed that collected profits can be added and that each node can be visited at most once. The non-negative travel cost or travel time between nodes i and j is represented as t_{ij}. The TOP can be formulated as an integer programming model with these decision variables: $x_{ijm} = 1$, if in route m, a visit to node i is followed by a visit to node j, and 0 otherwise; $y_{im} = 1$, if vertex i is visited in route m, and 0 otherwise. The variables u_{im} will be used to avoid subtours and they are an indication of the relative position of node i in vehicle route m.

$$\text{Maximize} \sum_{m=1}^{M} \sum_{i=2}^{|N|-1} P_i y_{im} \tag{3.1}$$

The objective function (3.1) is to maximize the total collected profit of visited nodes.

$$\sum_{m=1}^{M} \sum_{j=2}^{|N|} x_{1jm} = \sum_{m=1}^{M} \sum_{i=1}^{|N|-1} x_{i|N|m} = M \tag{3.2}$$

Constraints (3.2) ensure that each route starts from node 1 and ends on node $|N|$.

$$\sum_{m=1}^{M} y_{km} \leq 1; \ \forall k = 2, \ldots, (|N| - 1) \tag{3.3}$$

Constraints (3.3) ensure each node is visited at most once.

$$\sum_{i=1}^{|N|-1} x_{ikm} = \sum_{j=2}^{|N|} x_{kjm} = y_{km}; \ \forall k = 2, \ldots, (|N| - 1); \ \forall m = 1, \ldots, M \tag{3.4}$$

Constraints (3.4) ensure the connectivity of the routes.

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} t_{ij} x_{ijm} \leq T_{max}; \ \forall m = 1, \ldots, M \tag{3.5}$$

Constraints (3.5) ensure the limited time budget for each route.

$$2 \leq u_{im} \leq |N|; \ \forall i = 2, \ldots, |N|; \ \forall m = 1, \ldots, M \tag{3.6}$$

$$u_{im} - u_{jm} + 1 \leq (|N| - 1)(1 - x_{ijm}); \ \forall i, j = 2, \ldots, |N|; \ \forall m = 1, \ldots, M \tag{3.7}$$

The combination of constraints (3.6) and (3.7) prevents subtours for each route.

> It should be noted that in this formulation each route is bounded by the same maximal travel time T_{max}. If each route would have a different maximal travel time, T_{max} in the above formulation can be replaced easily by $T_{max,m}$.

Another mathematical formulation for the TOP is explained below. This formulation is mostly used when solving the problem with exact algorithms, such as branch-and-cut and branch-and-price algorithms. This formulation starts from the (typically very large) set $\Omega = \{r_1, r_2, \ldots, r_{|\Omega|}\}$ containing all the possible routes. Each route starts in node 1, ends in node $|N|$ and visits each node at most once in such a way that the total travel time does not exceed the time budget T_{max}. $\hat{p}_k$ represents the profit generated by route $r_k \in \Omega$. Let $a_{ik} = 1$ if route r_k visits node i and $a_{ik} = 0$ otherwise. For this model, the binary decision variables x_k indicate whether route $r_k \in \Omega$ is used or not. TOP is then formulated as follows:

$$\text{Maximize} \sum_{r_k \in \Omega} \hat{p}_k x_k; \tag{3.8}$$

$$\sum_{r_k \in \Omega} a_{ik} x_k \leq 1; \quad \forall i \in N \setminus \{1 \cup | \ N \ |\}; \tag{3.9}$$

$$\sum_{r_k \in \Omega} x_k \leq M; \tag{3.10}$$

$$x_k \in \{0, 1\}; \quad \forall r_k \in \Omega. \tag{3.11}$$

The objective function (3.8) calculates the collected profit from selected routes. Constraints (3.9) ensure that each node is visited at most once. The number of routes is limited to M, as represented in constraint (3.10). The binary constraints for the decision variables x_k are shown in constraints (3.11).

3.1.2 Complexity

Since it is proven that OP is NP-hard, the TOP extension of the OP is obviously also NP-hard. Therefore, no polynomial time algorithm has been designed, or is expected to be designed, to solve the TOP to optimality. By considering multiple routes in the mathematical model, the number of decision variables and constraints increases significantly. Moreover, solving a TOP with M routes is much more complex than solving the OP M times. For instance, each node can only be included in one route at most. So the TOP is not only about selecting the best nodes, but also about deciding which combination of nodes should be visited in each route. This comes down to optimally distributing the selected nodes over the available routes.

Therefore, it obviously gets harder to solve the TOP with exact algorithms. Exact algorithms will only be feasible for problems with a small number of nodes and small number of routes. Heuristic approaches will be mostly required, as discussed in Chap. 5. The computational times for the heuristics will increase since it should be considered how to move nodes from one route to another, group nodes together, and use the time budget in each route as much as possible.

3.2 Orienteering Problem with Time Windows

In the orienteering problem with time windows (OPTW), the visit to each node is constrained by time windows. In practice, this corresponds to customers which can only be visited during specified time intervals or tourist attractions which can only be visited during their opening hours.

The best solution for this network is 9 with the total travel cost of 9 (Fig. 3.5). The visited nodes are nodes 2 and 5. Node 2 is visited at time = 3 which is between its time window [2, 5] and node 5 is visited at time = 6 with its time window [5, 8]. This is different from the optimal OP, as shown in Fig. 2.8. The optimal solution is generated by visiting nodes 2, 3, and 5 with the total collected profit of 17. Node 3 is skipped since it can only be reached at time = 7 which is beyond its time window.

It should be emphasized that in scientific papers, the **time windows** for any routing problem always restrict the **start of the service, not the entire service**. This might be counter-intuitive because opening hours typically limit the entire service (and not only the start of the service). When considering an application in tourism, the opening

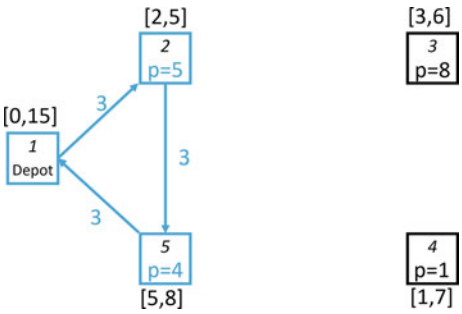

Fig. 3.5 Optimal OPTW solution for the example network

hours of a museum could for instance be from 10 am until 6 pm. This means that a tourist should enter and leave the museum between 10 am and 6 pm. If we assume that a visit to this particular museum takes two hours, the museum will be modeled with a window from 10 am until 4 pm, since the tourist visit should start before 4 pm.

It is also worthwhile to repeat here (see Sect. 2.2) the way that service or visiting times (two hours in the previous example) are modeled in many practical applications. When the total time available considers both traveling between nodes and visiting nodes, these service times should be modeled appropriately. A straightforward way to model the service times is to include these in the travel time. When considering time windows, this inclusion should be performed carefully. To make sure that the start of a visit occurs during the time window, the travel time between two customers will be composed of the service time at the first customer plus the actual travel time between the first customers and the second customer. In this way, the arrival time at the second customer corresponds to the actual arrival time and can be compared with the available time window of the second customer.

A small example should clarify and justify this way of modeling. Suppose we have a museum with a typical visiting time of two hours and opening hours from 10 am until 6 pm. As mentioned above, it will be modeled with a time window (for the start of the visit) from 10 am until 4 pm. A church in the same city can be visited between 9 and 11 am and a typical visit takes one hour. This church will be modeled with a time window from 9 am until 10 am. The actual travel time between the church and the museum is half an hour. For this example, the travel time (or cost) from the church to the museum will be modeled as one hour and a half and the travel time from the museum to the church will be modeled as two hours and a half. When a tourist starts the day with a visit to the church at 9.30 am, (s)he will arrive at the museum and start a visit there at 11 am. Indeed, the tourist will visit the church until 10.30 (outside the time window but inside the opening hours) and then travel half an hour to the museum. Note that with these time windows it is not possible to visit the museum before the church.

3.2.1 Formal Definition and Mathematical Model OPTW

As with the OP and TOP, consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. As with the OP and TOP, nodes 1 and $|N|$ can also represent a single depot and a profit $P_i = 0$ is associated with both nodes. In the OPTW, each node is assigned a time window $[O_i, C_i]$ and a visit to a node can only start during this time window. The goal of the OPTW is to determine a route that visits a subset of N and that maximizes the total collected profit. It is assumed that collected profits can be added and that each node can be visited at most once. The non-negative travel cost or travel time between nodes i and j is represented as t_{ij}. The start and end nodes are assigned the time window $[0, T_{max}]$. In this way, the total time available for traveling and visiting customers is limited to T_{max}. As mentioned in the previous section, service times are considered to be included in the travel times.

The OPTW can now be formulated as an integer programming model with the following decision variables: $x_{ij} = 1$ if a visit to node i is followed by a visit to node $j - 0$ otherwise; $s_i =$ the start of the service at node i and L a large constant.

$$\text{Maximize} \sum_{i=2}^{|N|-1} \sum_{j=2}^{|N|} P_i x_{ij} \tag{3.12}$$

The objective function (3.12) is to maximize the total collected profit from visited nodes.

$$\sum_{j=2}^{|N|} x_{1j} = \sum_{i=1}^{|N|-1} x_{i|N|} = 1 \tag{3.13}$$

Constraints (3.13) ensure that the route starts from node 1 and ends at $|N|$.

$$\sum_{i=1}^{|N|-1} x_{ik} = \sum_{j=2}^{|N|} x_{kj} \leq 1; \ \forall k = 2, \ldots, (|N| - 1) \tag{3.14}$$

Constraints (3.14) ensure the connectivity of the route and guarantee that each node is only visited at most once.

$$O_i \leq s_i; \ \forall i = 1, \ldots, |N| \tag{3.15}$$

$$s_i \leq C_i; \ \forall i = 1, \ldots, |N| \tag{3.16}$$

Both constraints (3.15) and (3.16) restrict the start of the service to the time window.

$$s_i + t_{ij} - s_j \leq L(1 - x_{ij}); \ \forall i, j = 1, \ldots, |N| \tag{3.17}$$

Constraints (3.17) ensure the timeline of the route. Indeed, if the vehicle is not traveling between i and j, the right-hand side of the constraint is equal to the large constant L (which could be T_{max} for instance) and no limits are imposed on s_i and s_j. But if the vehicle is traveling from i to j, the right-hand side becomes zero and the start of the service at node j should be not earlier than the start of the service at node i plus the travel time between i and j (which includes the service time at i). It should also be noted that 'less than or equal to' is used in this constraint and not just 'equal to'. This is required since the service at j does not have to start immediately upon arrival at j. The vehicle might wait before starting the service, for instance when the time window is not open yet.

Since the starting times of the services are now part of the mathematical formulation, it is no longer necessary to include subtour elimination constraints. Thanks to these starting times and the ordering by constraints (3.17), subtours are avoided.

One might also expect a constraint such as this one to limit the total travel time to the time budget T_{max}:

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} t_{ij} x_{ij} \leq T_{max}. \tag{3.18}$$

However, the problem with this constraint would be that it only limits the actual travel (and service) time, but it does not consider the fact that a vehicle can arrive at a customer before the time window opens and then waits there until it opens. This waiting time is not included in constraint (3.18) and it should be since T_{max} typically limits the total time available for traveling, servicing and waiting. By imposing the time window $[0,T_{max}]$ to the start ($i = 1$) and end node ($i = |N|$) this is accurately and elegantly modeled. A constraint similar to constraint (3.18) could be formulated to limit the battery capacity for electric vehicles or fuel for combustion vehicles.

3.2.2 Complexity

It is proven that the OP is NP-hard, as explained in Sect. 2.3.2. Therefore, the OPTW extension of the OP is also NP-hard. So, no polynomial time algorithm has been designed, or is expected to be designed, to solve this problem to optimality. By including the time windows in the mathematical model, the number of constraints significantly increases. Moreover, the mathematical model now keeps track of the starting times of the services. Therefore, the OPTW is harder to solve by exact algorithms than the OP. Exact algorithms are only feasible for problems with a small number of nodes (or wide time windows).

Heuristic approaches will be mostly required, as discussed in Chap. 6. The computational times for heuristics may also increase due to additional time windows constraints that need to be taken into consideration, e.g., inserting nodes needs to consider the time budget as well as the time window of all other visited nodes.

In many practical cases, the customers or tourist attractions actually have multiple time windows. They are open, for instance, between 9 am and 12 am and between 2 pm and 6 pm. This problem is called the **OP with multiple time windows (OPmTW)** where each node can have multiple time windows. To model these problems, each node or customer can be duplicated. So the single node with multiple time windows is replaced by one node open between 9 am and 12 am and another node on the same location is open between 2 pm and 6 pm, both with the same profit. Obviously, a constraint should then be included to make sure that at most one of those two (or more) duplicate nodes can be visited. These additional (capacity) constraints are further discussed in Sect. 8.1.

3.3 Team Orienteering Problem with Time Windows

The third well-known OP variant that will be discussed here combines the TOP with the OPTW. Therefore, this problem is called the team OP with time windows (TOPTW). Next to the time windows, M routes are considered. Obviously, the objective is still to maximize the total collected profit across these multiple routes.

If we assume two routes ($M = 2$), Fig. 3.6 illustrates a solution for the TOPTW. It seems that now all customers can be visited, but when we take a closer look, the second route visits node 4 at time 9, which actually violates the time window of node 4. Therefore, the optimal solution of the TOP is no longer optimal for the TOPTW due to time window constraints.

The optimal TOPTW solution is shown in Fig. 3.7. The first route visits nodes 2 and 5, returning to the depot at time 9, and the second route visits only node 3, returning to the depot at time 12. Node 4 cannot be included in these routes due to the time window constraints. The total collected profit is equal to $9 + 8 = 17$.

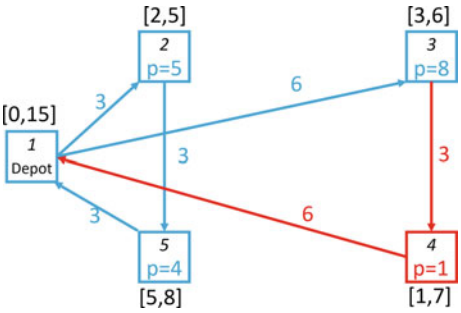

Fig. 3.6 Example of an infeasible solution for the TOPTW

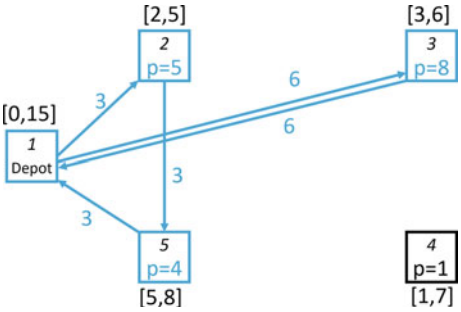

Fig. 3.7 Optimal TOPTW solution for the example network

3.3.1 Formal Definition and Mathematical Model TOPTW

Again, consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. As with the OP and TOP, nodes 1 and $|N|$ can also represent a single depot and a profit $P_i = 0$ is associated with both nodes. In the TOPTW, each node is assigned a time window $[O_i, C_i]$ and a visit to a node can only start during this time window. The goal of the TOPTW is to determine M routes that visit a subset of N and that maximize the total collected profit. It is assumed that collected profits can be added and that each node can be visited at most once. The non-negative travel cost or travel time between nodes i and j is represented as t_{ij}. The start and end nodes are assigned the time window $[0, T_{max}]$. In this way, in each route the total time available for traveling and visiting customers is limited to T_{max}. As mentioned above, service times are considered to be included in the travel times.

The TOPTW can now be formulated as an integer programming model with the following decision variables: $x_{ijm} = 1$, if in route m, a visit to node i is followed by a visit to node j, and 0 otherwise; $y_{im} = 1$, if vertex i is visited in route m, and 0 otherwise; s_{im} = the start of the service at node i in route m; L is large constant.

$$\text{Maximize} \sum_{m=1}^{M} \sum_{i=2}^{|N|-1} P_i y_{im} \tag{3.19}$$

The objective function (3.19) is to maximize the total collected profit of visited nodes from all routes.

$$\sum_{m=1}^{M} \sum_{j=2}^{|N|} x_{1jm} = \sum_{m=1}^{M} \sum_{i=1}^{|N|-1} x_{i|N|m} = M \tag{3.20}$$

Constraints (3.20) ensure that each route starts from node 1 and ends in node $|N|$.

$$\sum_{m=1}^{M} y_{km} \leq 1; \ \forall k = 2, \ldots, (|N| - 1) \tag{3.21}$$

Constraints (3.21) ensure each node can only be visited at most once.

$$\sum_{i=1}^{|N|-1} x_{ikm} = \sum_{j=2}^{|N|} x_{kjm} = y_{km};\ \forall k = 2, \ldots, (|N|-1);\ \forall m = 1, \ldots, M \quad (3.22)$$

$$s_{im} + t_{ij} - s_{jm} \leq L(1 - x_{ijm});\ \forall i, j = 1, \ldots, |N|;\ \forall m = 1, \ldots, M \quad (3.23)$$

Constraints (3.22) and (3.23) ensure the connectivity and timeline of each route.

$$O_i \leq s_{im};\ \forall i = 1, \ldots, |N|;\ \forall m = 1, \ldots, M \quad (3.24)$$

$$s_{im} \leq C_{im};\ \forall i = 1, \ldots, |N|;\ \forall m = 1, \ldots, M \quad (3.25)$$

Both constraints (3.24) and (3.25) restrict the start of the service to the time window for each route. Again, no subtour elimination constraints are required and the total time available is limited by the time windows of the start and end node.

3.3.2 Complexity

Since the OP, TOP, and OPTW are all NP-hard, the TOPTW is also NP-hard. Including multiple routes as well as the time windows constraints obviously makes this problem even harder to solve by exact algorithms, as illustrated in Figs. 3.6 and 3.7. Again, heuristic approaches will be mostly required, as discussed in Chap. 6.

3.4 Related Literature

A comprehensive summary of TOP, OPTW, and TOPTW, including algorithms, benchmark instances and new best known solutions can be found in Vansteenwegen et al. (2011a) and Gunawan et al. (2016). The introduction of the TOP can be found in Chao et al. (1996b). Various applications of TOP are listed in works of Butt and Cavalier (1994) and Tang and Miller-Hooks (2005).

A very fast algorithm for the OPTW and the TOPTW and an overview of benchmark instances can be found in Vansteenwegen et al. (2009). Other algorithms are available in Lin and Yu (2012) and Hu and Lim (2014). The two most recent algorithms, Iterated Local Search and a hybridization of Simulated Annealing and Iterated Local Search, called SAILS, are introduced in Gunawan et al. (2017). Recently, ADOPT is introduced in Gunawan et al. (2018), combining two processes of determining parameter configurations and designing the local procedure to enhance the performance of ILS and SAILS. The details of these state-of-the-art algorithms and benchmark instances will be described in Chaps. 5 and 6.

References

Butt SE, Cavalier T (1994) A heuristic for the multiple tour maximum collection problem. Comput Oper Res 21(1):101–111

Chao IM, Golden BL, Wasil EA (1996b) The team orienteering problem. Eur J Oper Res 88(3):464–474

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332

Gunawan A, Lau HC, Vansteenwegen P, Kun L (2017) Well-tuned algorithms for the team orienteering problem with time windows. J Oper Res Soc 68(8):861–876

Gunawan A, Lau HC, Kun L (2018) ADOPT: combining parameter tuning and adaptive operator ordering for solving a class of orienteering problems. Comput Ind Eng 121:82–96

Hu Q, Lim A (2014) An iterative three-component heuristic for the team orienteering problem with time windows. Eur J Oper Res 232(2):276–286

Lin SW, Yu VF (2012) A simulated annealing heuristic for the team orienteering problem with time windows. Eur J Oper Res 217(1):94–107

Tang H, Miller-Hooks E (2005) A tabu search heuristic for the team orienteering problem. Comput Oper Res 32(6):1379–1407

Vansteenwegen P, Souffriau W, Vanden Berghe G, Van Oudheusden D (2009) Iterated local search for the team orienteering problem with time windows. Comput Oper Res 36(12):3281–3290

Vansteenwegen P, Souffriau W, Van Oudheusden D (2011a) The orienteering problem: a survey. Eur J Oper Res 209(1):1–10

Chapter 4
State-of-the-Art Solution Techniques for PTP and PCTSP

In Chaps. 2 and 3, different orienteering problems (or routing problems with profits) were introduced. The single vehicle problems were discussed in Chap. 2: the profitable tour problem (PTP), the prize-collecting traveling salesperson problem (PCTSP), and the orienteering problem (OP). The multi vehicle problems were discussed in Chap. 3: the team orienteering problem (TOP) and the team orienteering problem with time windows (TOPTW). For discussing the state-of-the-art solution techniques for these different orienteering problems in Chaps. 4, 5, and 6, the problems will be classified differently, based on the similarities between the solution techniques. Therefore, the PTP and PCTSP are discussed in this chapter, the OP and TOP in the next chapter and the problems with time windows, OPTW and TOPTW, in Chap. 6. Moreover, it should be noted that only a few solution techniques have been developed for the PTP and the PCTSP, while many more solution techniques have been developed for the OP, TOP, OPTW, and TOPTW.

4.1 Benchmark Instances

For the **PTP**, no benchmark instances are available, as far as we know. This is obviously related to the fact that only very few solution approaches have been developed for tackling the PTP. However, a number of benchmark instances are available for the PTP with consistency constraints and multiple vehicles. These could also be used as benchmark instances for the basic PTP and they will be described here as such.

Thirteen small-scale instances for the PTP (with consistency constraints and multiple vehicles) and 36 medium- and large-scale instances were created. The medium and large-scale instances are based on the traditional VRP instances of Christofides and Eilon containing 50–199 nodes. For each node i, a profit was generated based on this formula:

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_4

$$P_i = b_i l \frac{\sum_{j \in N} t_{ij}}{|N|}. \tag{4.1}$$

Here, l is a random number uniformly distributed in the interval [0, 1] and b_i is a number linking the acquired profit to the transportation cost in such a way that the total profit equals 3–5 times the transportation cost. To this end, the node with the highest demand is assigned the value 5, i.e., $b_i = 5$, and the node with the lowest demand is assigned the value 3, i.e., $b_i = 3$. All other nodes are assigned a corresponding value b_i according to the ranking of their demand. Adopting the same process, the small instances contain 10–12 nodes. Finally, the small-scale instances set was enriched with some larger instances containing up to 18 nodes.

For the **PCTSP**, a first set of benchmark instances is randomly generated. This means that the instances themselves are not unique or made available as such, but instead it is specified how the instances can be generated. These instances have between 20 and 500 nodes ($|N| = \{20, 40, 60, 80, 100, 200, 300, 400, 500\}$) and the travel costs t_{ij} are derived from a uniform distribution in [0, 1000]. The profits P_i are also randomly derived from a uniform distribution in [0, 100]. Then, typically, different sets of instances are generated by defining the minimal profit that needs to be collected as a percentage of the sum of the total available profit. That percentage, called α, is 20, 50, or 80%. Often, the performance of the solution techniques will be different for different values of α. A higher value of α implies that more nodes will have to be visited. In general, when α is 80% solution techniques starting from visiting (almost) all nodes will be more efficient, while instances with an α of only 20% will be solved more efficiently by techniques that start from an empty solution and include (the most promising) nodes one by one.

Another set of PCTSP instances is based on 5 VRP and 46 TSP instances. All instances of the TSPLIB with 70 to 532 nodes, for which the node coordinates are available, are used. For the VRP instances, the demand is used as profit for the PCTSP. For the TSP instances, three different sets of instances are generated where the profit for each node i is generated differently:

- $P_i = 1$;
- $P_i = 1 + (7141i + 73) mod 100$;
- $P_i = 1 + \lfloor 99 t_{1,i} / \theta \rfloor$, where $\theta = max_{j \in N} \{t_{1,j}\}$.

The idea behind these three sets of TSP-based instances is that the first set contains in general easy instances because all profits are equal. The second set contains instances with pseudorandom profits and the third set are harder instances where larger profits are associated with nodes further from the depot. As in the first set of benchmark instances, the minimal profit that needs to be collected is defined as a percentage of the sum of the total available profit. In this case, an α of 25, 50, or 75% is used.

4.2 Exact Approaches

Until now, no exact approaches have been designed for the **PTP**. However, a number of approximation algorithms have been proposed for both the undirected (or symmetric) PTP and the directed (or asymmetric) PTP. These approximation algorithms are used to solve the PTP and they guarantee that the obtained solution has a certain quality relative to the optimal solution. More details on these approximation algorithms can be found in the papers mentioned in Sect. 4.4.

The earliest research on the **PCTSP** focused on bounding procedures, based on different relaxations of the constraints. We will discuss a branch-and-cut algorithm exploiting valid inequalities derived from studies on the knapsack and traveling salesperson polytopes and the orienteering problem. Two types of inequalities are based on knapsack constraints. The first is a *cover* inequality based on the constraint for collecting the minimal required profit and the second is a *cost-cover* inequality, using an upper bound on the optimal cost value to define a cost-based knapsack constraint. Moreover, the *cycle-cover* and the *conditional* inequalities use a knapsack constraint to strengthen the subtour elimination constraints. Some valid inequalities are also adapted from the TSP. In this case, the *comb inequalities* are considered. Finally, *logical* inequalities are introduced, stating that, when an edge is part of the solution, also its corresponding node is visited.

The branch-and-cut algorithm starts with generating an initial feasible solution and thus an upper bound. The initial solution is found through a heuristic algorithm (called GENIUS) and an improvement heuristic (called EXTENSION and COLLAPSE). The heuristic has a route construction phase based on insertions and a post-optimization phase. The improvement heuristic is an iterative execution of an EXTENSION phase (adding nodes) and a COLLAPSE phase (reducing the travel cost). The iterative part of the branch-and-cut algorithm always starts with selecting the next branching node to be explored. Nodes are considered in non-decreasing order of their associated LP solution value (thus the lower bound). The LP itself is solved with a commercial software such as CPLEX or Gurobi. The quality of the upper bound is regularly improved in order to speed up the algorithm. During the separation phase violated inequalities are identified. They consider both the subtour elimination constraints as well as all the above mentioned sets of inequalities. Different separation sequences are tested. The algorithm uses a widely known branching rule, always branching on the most fractional variable.

This branch-and-cut algorithm was tested successfully on the above mentioned instances with between 70 and 532 nodes. It turns out that subtour elimination and logical inequalities alone are not sufficient to tackle hard problems. The inclusion of cover inequalities appears to be sufficient for problems where α is 25%. The comb inequalities are required when α is 75%. When α is 50%, both the cover and comb inequalities are required.

4.3 Heuristics

For a detailed description of a complete algorithm, we refer to the papers mentioned in Sect. 4.4. Here, we discuss the main features and try to indicate some principles and insights for designing a successful (meta)heuristic when dealing with the PTP or the PCTSP.

Although the **PTP** is a single-objective problem, two conflicting components need to be considered. The first component maximizes the total collected profit, whereas the second component minimizes the total traveling cost. When maximizing the profit, the subset of customers that will be visited needs to be extended. Minimizing the distance considers optimizing the visiting sequence. The Adaptive Tabu Search metaheuristic we will discuss here takes advantage of different search landscapes and seeks to explore the solution space on the basis of both components.

This metaheuristic is a multi-start local search approach. It uses a greedy randomized constructive heuristic to generate a number of diversified initial solutions. During the improvement of the initial solutions, multiple neighborhood moves are applied to explore the solution space.

> As is typically the case for vehicle routing problems with profits, two different sets of moves are utilized: the ones that increase the total collected profit and the ones that decrease the total traveling cost. Since these moves are relevant for almost any algorithm addressing the PTP (or PCTSP or OP), we discuss these in detail here.

Local search moves increasing the profit. The moves increasing the profit, namely 1–1 REPLACE, 2–1 REPLACE and 1–2 REPLACE, change the decisions made regarding the subset of visited customers. The 1–1 REPLACE seeks to swap a visited node with a non-visited node (with a higher profit). All possible insertion positions for the non-visited node are examined prior to and after the removal of the visited node. If a feasible insertion position is found, without removing a visited node, then a single node insertion is performed in the least-cost feasible position. This is illustrated in Fig. 4.1: node 4 can be inserted between node 5 and the depot.

If removing a visited node is necessary to insert an unvisited node and obtain a feasible solution, the visited node is replaced by the unvisited node. This is illustrated in Fig. 4.2: node 4 is now removed and node 3 is inserted, increasing the collected profit with 7.

Similarly, the 2–1 REPLACE attempts to remove two visited nodes, regardless their position, and insert a non-visited one. All possible insertion positions are examined after the removal of both visited nodes. The 1–2 REPLACE attempts to remove one visited node and to insert two non-visited nodes. As before, all feasible insertion positions are examined for the pair of non-visited nodes.

Local search moves decreasing the travel cost. For decreasing the traveling cost in a solution, the well-known 2- OPT move is considered, illustrated in Fig. 4.3. By

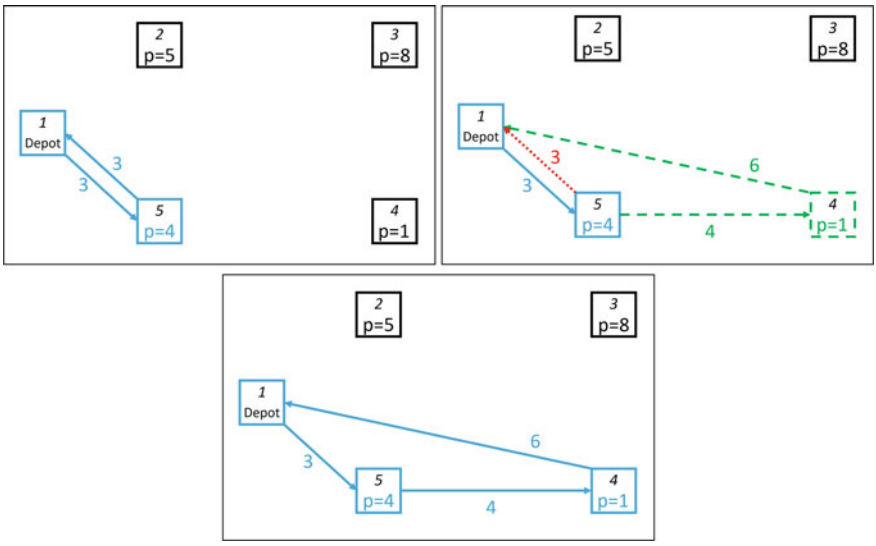

Fig. 4.1 Example of the single node insertion

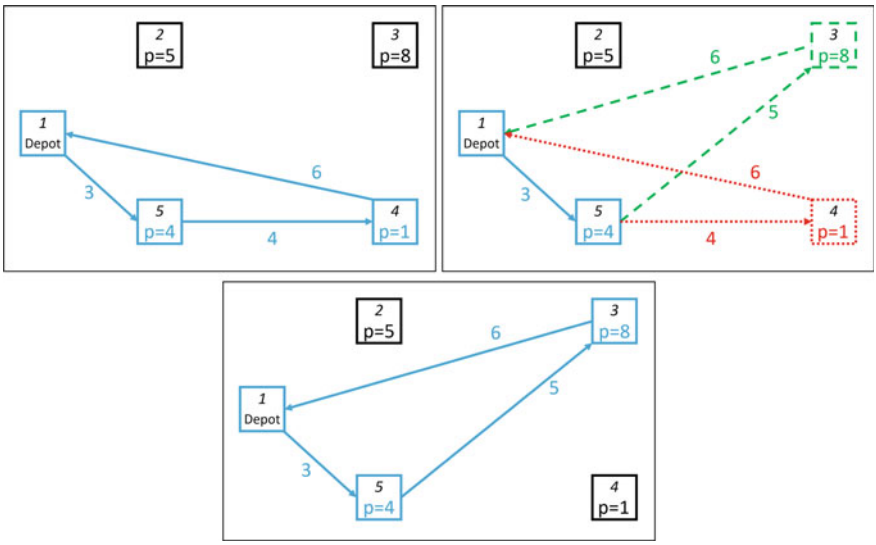

Fig. 4.2 Example of 1–1 REPLACE

changing the order in which the nodes are visited, 5 travel cost units can be saved. In the first situation, the travel cost is $6 + 4 + 5 + 6 = 21$, in the last situation, after the 2-Opt move, the travel cost is $3 + 4 + 3 + 6 = 16$. It should be noted that the direction of the arc between nodes 4 and 5 is also changed due to the 2- OPT move.

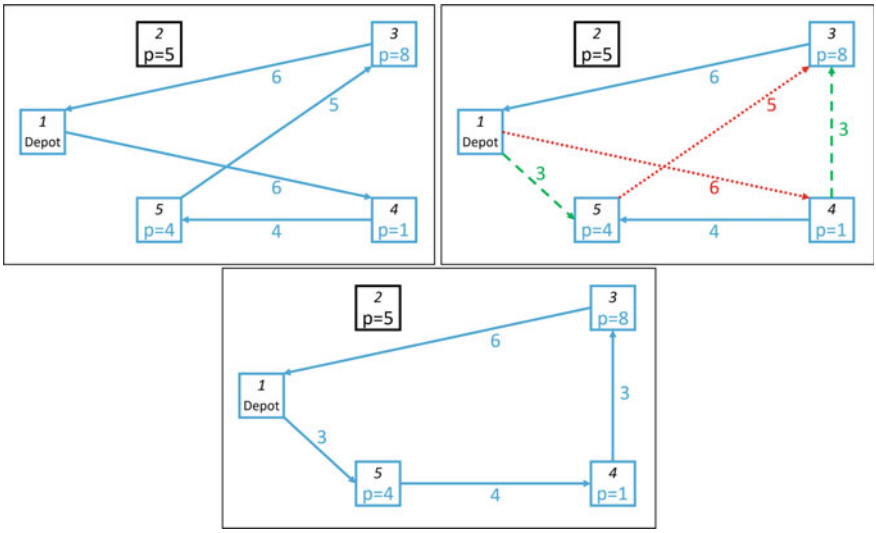

Fig. 4.3 Example of 2- OPT

Other moves considered are 1–1 EXCHANGE, swapping the position of two nodes, and 1–0 RELOCATE, changing the position of a single node in the solution. Since the algorithm addresses a PTP with multiple vehicles, a CHANGEVEHICLE move is also considered. The CHANGEVEHICLE move seeks to alter the assignment of a node from one vehicle to another. The advantage of the aforementioned moves is their simplicity and their ease of implementation. When considering insertions, it is clear that nodes with a high profit and a low additional travel cost are more promising and should be considered first.

A key element of the algorithm is the use of short- and long-term memory structures. The short-term memory prevents cycling and revisiting the same solutions, while the long-term memory records the high quality solution characteristics obtained during the search.

Only a few heuristic approaches for solving the **PCTSP** have been developed. We will discuss a Lagrangian heuristic here. This heuristic starts by calculating a lower bound by relaxing in a Lagrangian fashion the compact formulation of the PCTSP. Therefore, the constraint on the minimal profit that needs to be collected is added to the objective function with a Lagrange multiplier. The resulting problem can then be reduced to the asymmetric TSP. This asymmetric TSP is solved for different values of the Lagrange multiplier in order to obtain the best lower bound. When this lower bound is not a feasible solution, the algorithm tries to obtain a feasible solution, by visiting additional nodes. Obviously, these nodes are selected based on the ratio of profit over additional cost. As with the other routing problems with profits, a higher profit and a lower cost make a node more interesting. Once a feasible solution is obtained, the heuristic tries to further improve this solution

by successively extending and reducing the solution (called Extension and Collapse and also mentioned above). Extensions and reductions are both based on profit and cost and the solution always remains feasible. In the best version of the Lagrangian heuristic, extending and reducing of the solution is integrated in calculating the lower bound.

Noticeably, solution approaches for the PTP and the PCTSP have a number of similarities. Both are looking for inserting additional nodes with high profits and low additional travel cost. Both also try to minimize the travel cost to visit the set of selected nodes. Therefore, very similar or even the same local search moves can be considered in the solution approaches. However, an important difference between both problems, resulting also in different solution approaches, is that for the PTP any set of nodes contains a feasible solution, while for the PCTSP only sets with sufficient profit are acceptable. Moreover, once the minimal profit is reached for the PCTSP, no additional nodes should be considered for insertion, while for the PTP, visiting an additional node should always be considered as long as the additional travel cost is smaller than its profit.

4.4 Related Literature

Reviews on solution techniques for the PTP and the PCTSP can be found in Feillet et al. (2005), Aksen and Aras (2006), and Archetti et al. (2014).

The recent paper by Stavropoulou et al. (2019) discusses in detail the Adaptive Tabu Search metaheuristic for tackling the PTP (with multiple vehicles and consistency constraints). To the best of our knowledge, this is the only metaheuristic available for (a variant of) the PTP. This paper also introduces benchmark instances for the PTP with consistency constraints and multiple vehicles.

The PCTSP was first mentioned in Balas (1989). It is also discussed in DellAmico et al. (1995) and Balas (2002). The branch-and-cut algorithm for the PCTSP discussed above is proposed in Bérubé et al. (2009). A polyhedral study of the PCTSP (with penalties), can be found in Balas (1995). The Lagrangian heuristic to solve the PCTSP is proposed in DellAmico et al. (1998).

The way the profits are generated for the set of PCTSP instances based on 46 TSP instances is discussed in detail in Fischetti et al. (1998).

An approximation algorithm for the undirected PTP can be found in Goemans and Bertsimas (1990). An approximation algorithm for the directed version of the PTP is available in Nguyen (2010) and Nguyen and Nguyen (2010). Approximating algorithms for the PCTSP are discussed in Goemans (2009).

References

Aksen D, Aras N (2006) Customer selection and profit maximization in vehicle routing problems. In: Haasis H, Kopfer H, Schönberger J (eds) Research proceedings 2005. Springer, Berlin, pp 37–42

Archetti C, Speranza MG, Vigo D (2014) Vehicle routing problems with profits. In: Toth P, Vigo D (eds) Vehicle routing: problems, methods, and applications. MOS-SIAM series on optimization. SIAM, Philadelphia, pp 273–298

Balas E (1989) The prize collecting traveling salesman problem. Networks 19:621–636

Balas E (1995) The prize collecting traveling salesman problem. II: polyhedral results. Networks 25:199–216

Balas E (2002) The prize collecting traveling salesman problem and its applications. In: Gutin G, Punnen A (eds) Traveling salesman problem and its variations. Kluwer Academic Publishers, Dordrecht, pp 663–695

Bérubé JF, Gendreau M, Potvin JY (2009) A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem. Networks 54:56–67

DellAmico M, Maffioli F, Vbrand P (1995) On prize-collecting tours and the asymmetric travelling salesman problem. Int Trans Oper Res 2:297–308

DellAmico M, Maffioli F, Sciomachen A (1998) A Lagrangian heuristic for the prize collecting travelling salesman problem. Ann Oper Res 81:289–306

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transp Sci 39(2):188–205

Fischetti M, Salazar-González JJ, Toth P (1998) Solving the orienteering problem through branch-and-cut. INFORMS J Comput 10:133–148

Goemans M (2009) Combining approximation algorithms for the prize-collecting TSP. In: Proceedings of CORR

Goemans M, Bertsimas D (1990) On the parsimonious property of connectivity problems. In: SODA 1990 proceedings of the first annual ACM-SIAM symposium on discrete algorithms, pp 388–396

Nguyen V (2010) A primal-dual approximation algorithm for the asymmetric prize-collecting TSP. In: Wu W, Daescu O (eds) Combinatorial optimization and applications. Lecture notes in computer science. Springer, Berlin, pp 260–269

Nguyen V, Nguyen T (2010) Approximating the asymmetric profitable tour. Electron Not Discret Math 36:907–914

Stavropoulou F, Repoussis P, Tarantilis C (2019) The vehicle routing problem with profits and consistency constraints. Eur J Oper Res 274:340–356

Chapter 5
State-of-the-Art Solution Techniques for OP and TOP

Definitions and mathematical models of the OP and the TOP were introduced in Chaps. 2 and 3. In this chapter, we will discuss the benchmark instances and state-of-the-art solution techniques for both OP and TOP. Some illustrations of benchmark instances and solutions are included in order to increase the understanding in the difficulty of this problem and to provide additional insights. The solution techniques are classified into two different categories: exact approaches and (meta)heuristic techniques.

5.1 Benchmark Instances

A number of benchmark OP and TOP instances are available from many different research papers. All benchmark instances are available in a digital format on this webpage: http://www.mech.kuleuven.be/en/cib/op.

5.1.1 OP Benchmark Instances

The first three sets of OP instances, namely Tsiligirides-set1-32, Tsiligirides-set2-21, and Tsiligirides-set3-33, consist of 32, 21, and 33 nodes, respectively. Given the Cartesian coordinates, the Euclidean distances between two nodes i and j can be determined. Obviously, also a profit is associated to each node. Each set contains a number of instances on the same network, with different values of T_{max}: the first set has values $\{5, 10, 15, \ldots, 85\}$, the second set has values $\{15, 20, 25, \ldots, 45\}$, and the last set has values $\{15, 20, 25, \ldots, 110\}$. Figures 5.1, 5.2 and 5.3 show these benchmark networks with 32, 21, and 33 nodes. For each node, its index is presented

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_5

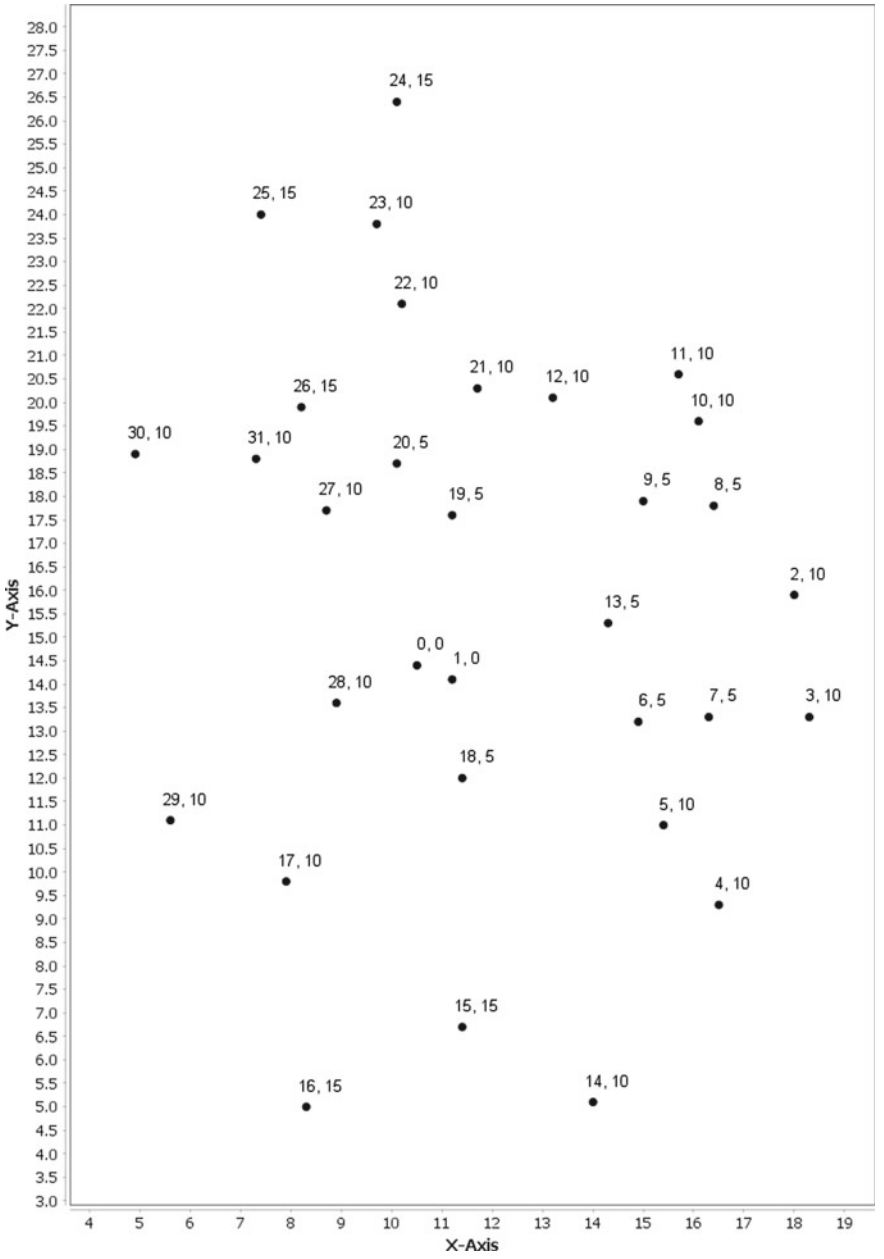

Fig. 5.1 Benchmark instance with 32 nodes (Tsiligirides-set1-32)

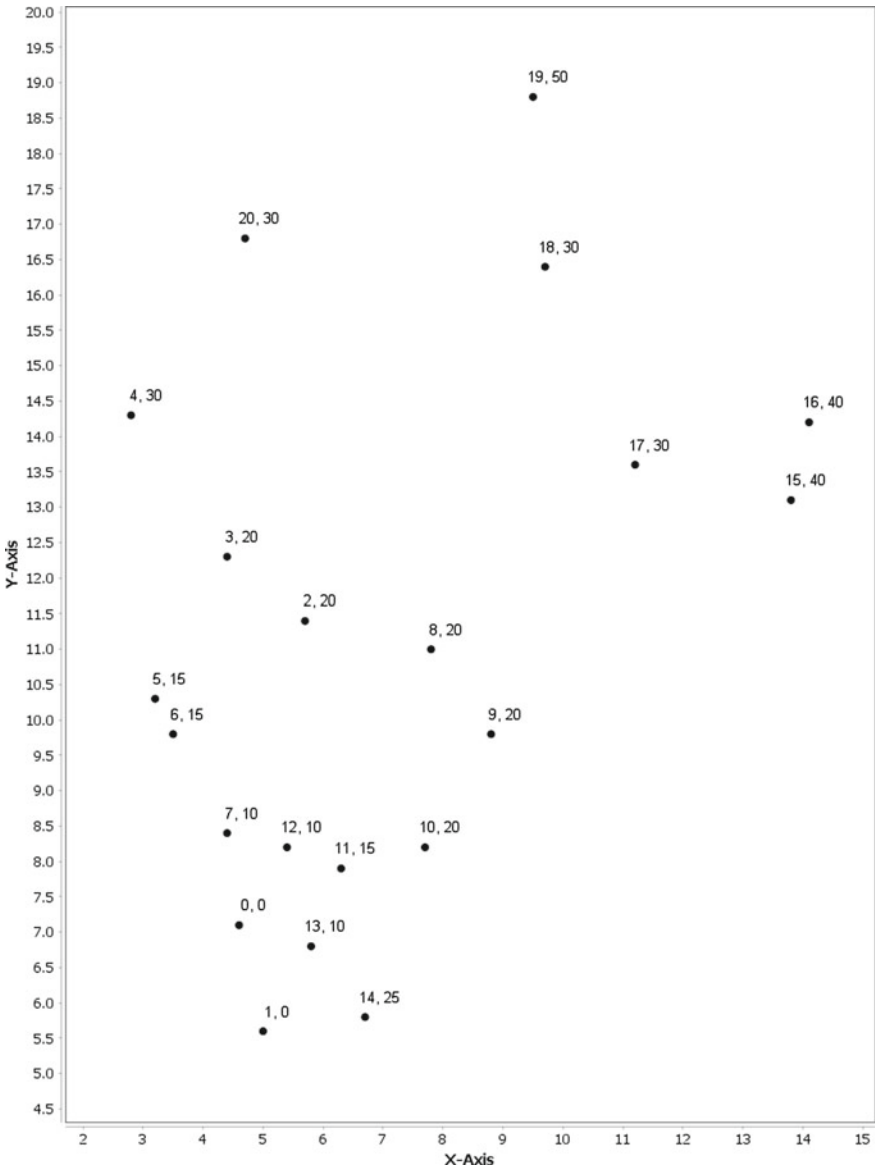

Fig. 5.2 Benchmark instance with 21 nodes (Tsiligirides-set2-21)

together with its profit. The node with index 0 (and profit 0) is the start node, the node with index 1 (and profit 0) is the end node.

Two other sets of instances for the **OP** are available as well. The first set, Chao-set1-66, consists of 26 instances in which the locations of the 66 nodes in the network form a square. The start and end nodes are close to each other and located at the center

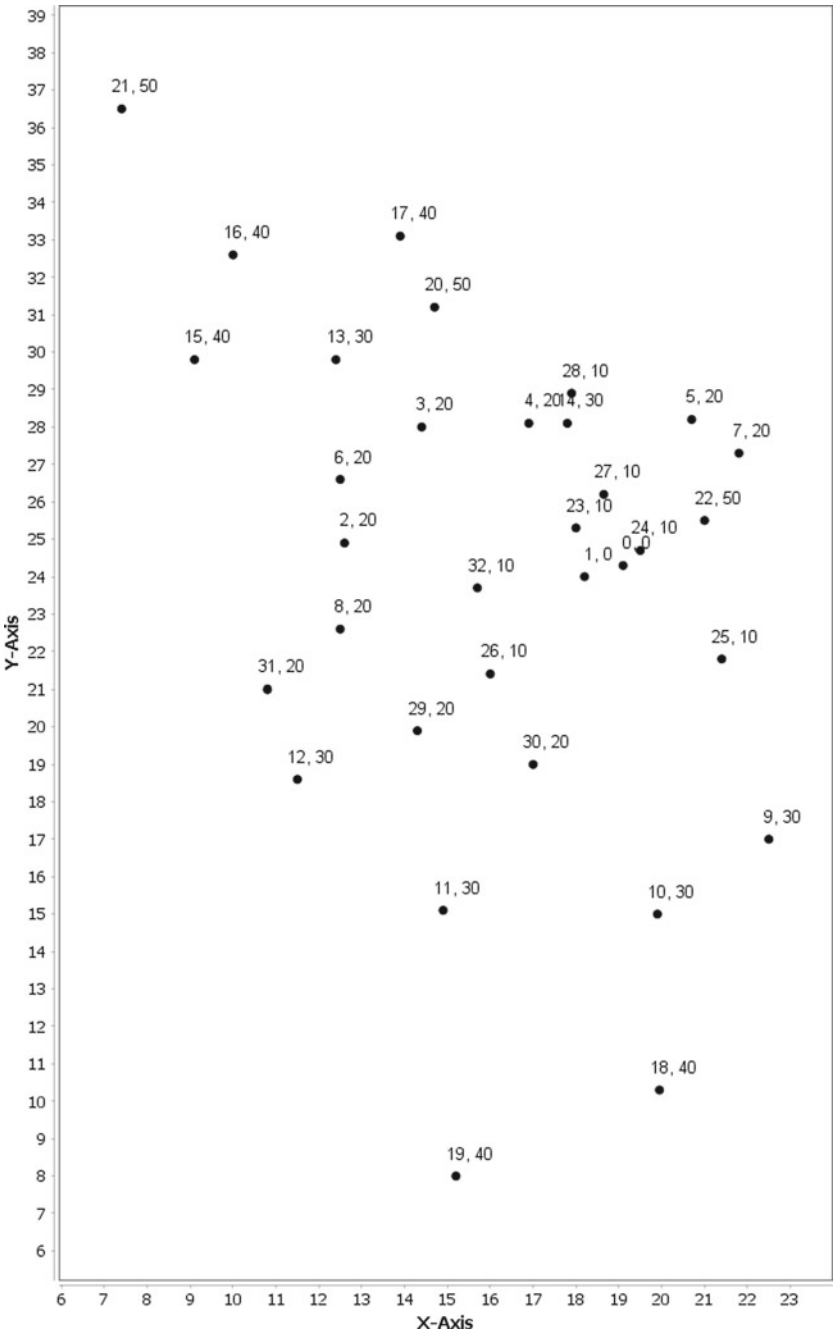

Fig. 5.3 Benchmark instance with 33 nodes (Tsiligirides-set3-33)

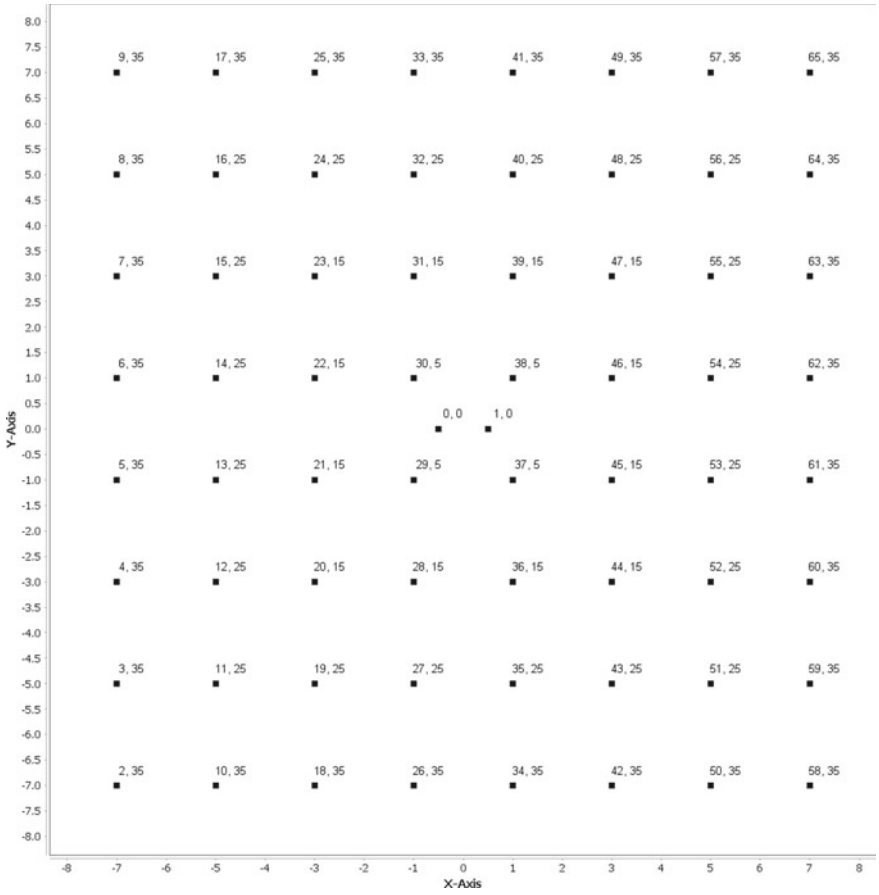

Fig. 5.4 Benchmark instance with 66 nodes

of four concentric squares. The distance between two adjacent nodes is either 2 or $\sqrt{2}$. The scores are set in such a way that nodes that comprise the innermost, second, third, and outermost concentric squares have scores of 5, 15, 25, and 35, respectively. The values of T_{max} are taken from $\{5, 10, 15, \ldots, 130\}$. Figure 5.4 illustrates the network with 66 nodes. The node with index 0 is the start node and the node with index 1 is the end node.

The second set, Chao-set2-64, consists of 14 instances in which the locations of 64 nodes take on a diamond shape. Unlike the first set, the start and end nodes are located far apart at the top and bottom of the diamond. The scores are multiples of 6 and range from 6 to 42. The values of T_{max} are taken from $\{15, 20, 25, \ldots, 80\}$. Figure 5.5 illustrates the network with 64 nodes. The node with index 0 is the start node and the node with index 63 is the end node.

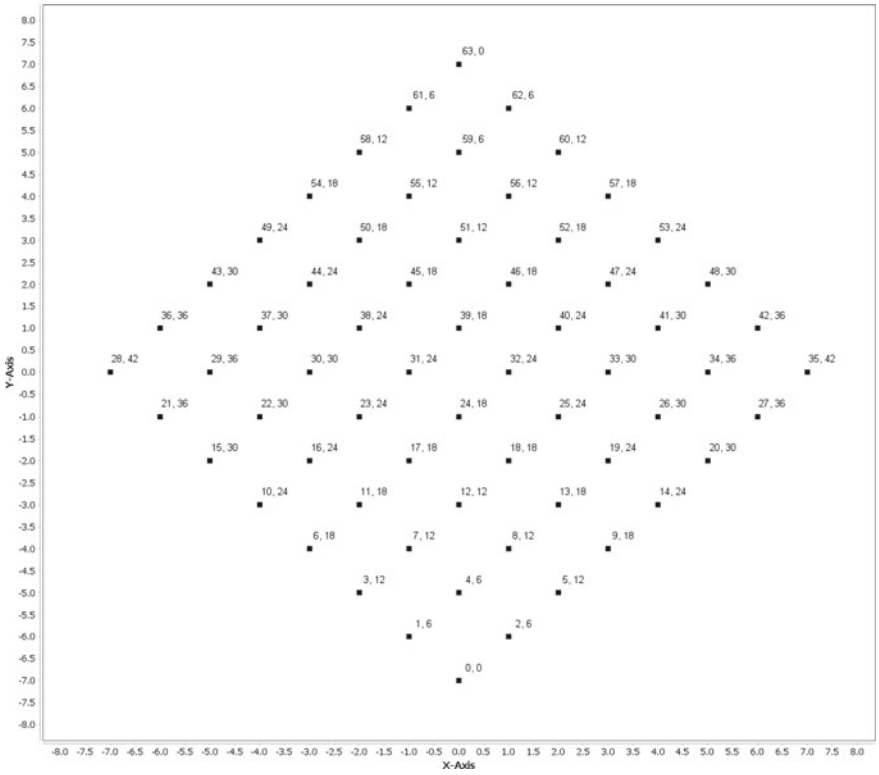

Fig. 5.5 Benchmark instance with 64 nodes

Four more sets of instances are available. The first set of instances, Fischetti-set1, is based on twelve VRP instances and the first three OP instances (Tsiligirides) mentioned above. For the OP instances, the travel times of the above-mentioned instances are multiplied by 100 and rounded to the nearest integer. For the VRP-based instances, the customer demand is interpreted as the node profit.

The second set of OP instances, Fischetti-set2, is based on 44 TSP instances with three different ways of generating the profit of each node i:

- $P_i = 1$;
- $P_i = 1 + (7141i + 73) mod 100$;
- $P_i = 1 + \lfloor 99t_{1,i}/\theta \rfloor$, where $\theta = max_{j \in N} \{t_{1,j}\}$.

These instances are generated in the same way as the PCTSP instances discussed in Chap. 4. The first category contains somewhat easier instances because all profits are equal. So the objective is to visit as many nodes as possible. The second category contains instances with pseudorandom profits and the third category are harder instances where larger profits are associated with nodes further from the depot.

One example of Fischetti-set2 is shown in Fig. 5.6. This figure shows a benchmark instance, kroA100, with the profit of each node equal to 1.

The last two Fischetti sets of benchmark instances, Fischetti-set3 and Fischetti-set4, are randomly generated in different ways. Fischetti-set3 consists of 4×11 instances with four different values of T_{max} and eleven different values of N. Both profits and X-Y coordinates are generated as uniformly random integers in range [1, 100]. The travel times are computed through the shortest-path. Fischetti-set4 consists of 5×15 instances with five different values of T_{max} and 15 different values of N. These instances are similar to Fischetti-set2 but nodes are generated as random points in the [0, 100] square according to a uniform distribution.

> For all Fischetti instances, the maximum total travel time is defined as $\lceil \alpha \cdot v(TSP) \rceil$, where $v(TSP)$ represents the length of the shortest Hamiltonian route visiting all nodes and α is a parameter taking different values, e.g., 0.25, 0.50, etc.

Recently, a new set of instances with more than 400 nodes is generated, Kobeaga-set7397. 41 TSPLib instances are selected for this new set with between 417 and 7397 nodes and 4 different generations of the profits, resulting in 164 instances. Again, the maximum total travel time is defined as $\lceil \alpha \cdot v(TSP) \rceil$, where $v(TSP)$ represents the length of the shortest Hamiltonian route visiting all nodes and α is a parameter. In this case, each instance has a particular value of α. These larger instances needed to be generated since most 'smaller' instances are solved to optimality, as we will discuss in Sect. 5.2.1. These 164 instances are available here: https://www.github.com/bcamath-ds/OPLib. Figure 5.7 illustrates one instance with 783 nodes (rat783).

5.1.2 TOP Benchmark Instances

For the TOP, benchmark instances are generated by using the OP instances: Tsiligirides-set1-32, Tsiligirides-set2-21, Tsiligirides-set3-33, Chao-set1-66, Chao-set2-64. Each instance consists of two, three, or four routes. The T_{max} value for each route in these instances is obtained by dividing the original T_{max} value by the number of routes m.

Later, a set of 333 larger instances, Dang-set, is generated from Fischetti-set1 (VRP instances) and Fischetti-set2 with three different ways of generating the profit, as explained earlier. Again, the time budget per route T_{max} in these instances is obtained by dividing the original T_{max} value by the number of routes m. These instances can be accessed in the following link https://www.hds.utc.fr/~moukrim/dokuwiki/en/top.

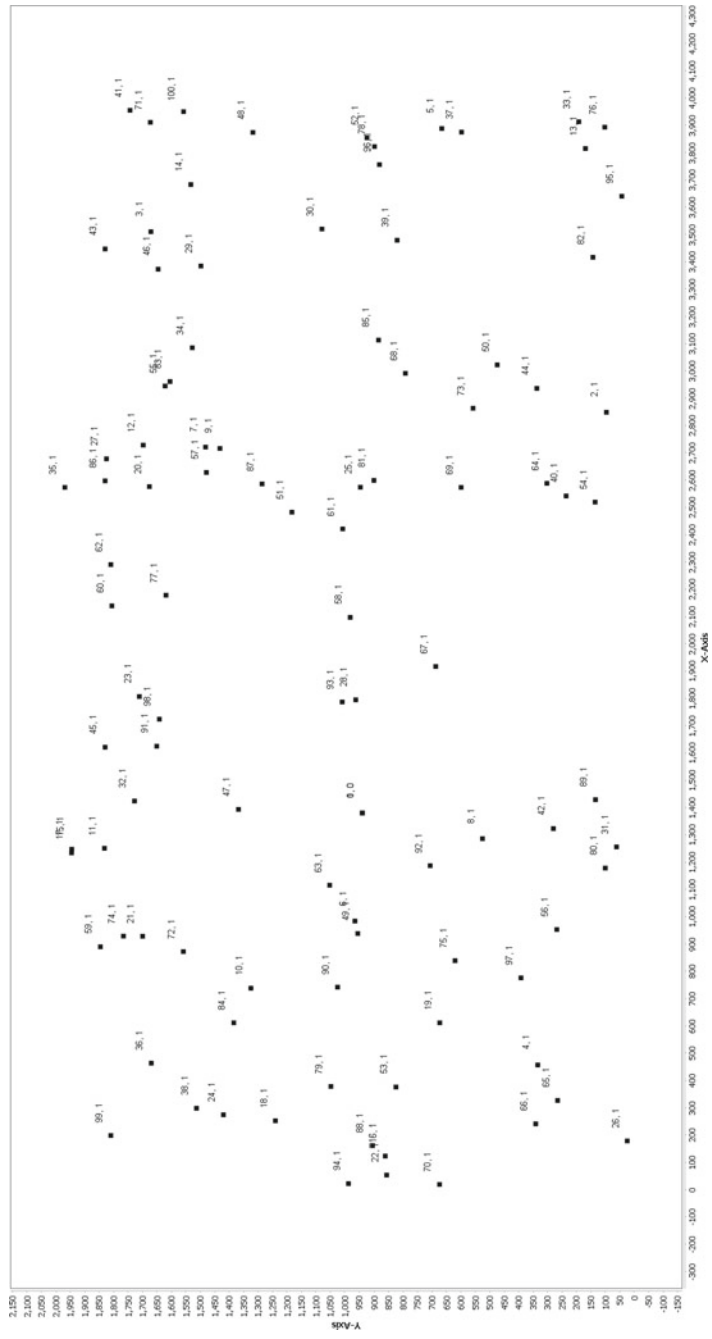

Fig. 5.6 Fischetti-set2 benchmark instance with 100 nodes (kroA100)

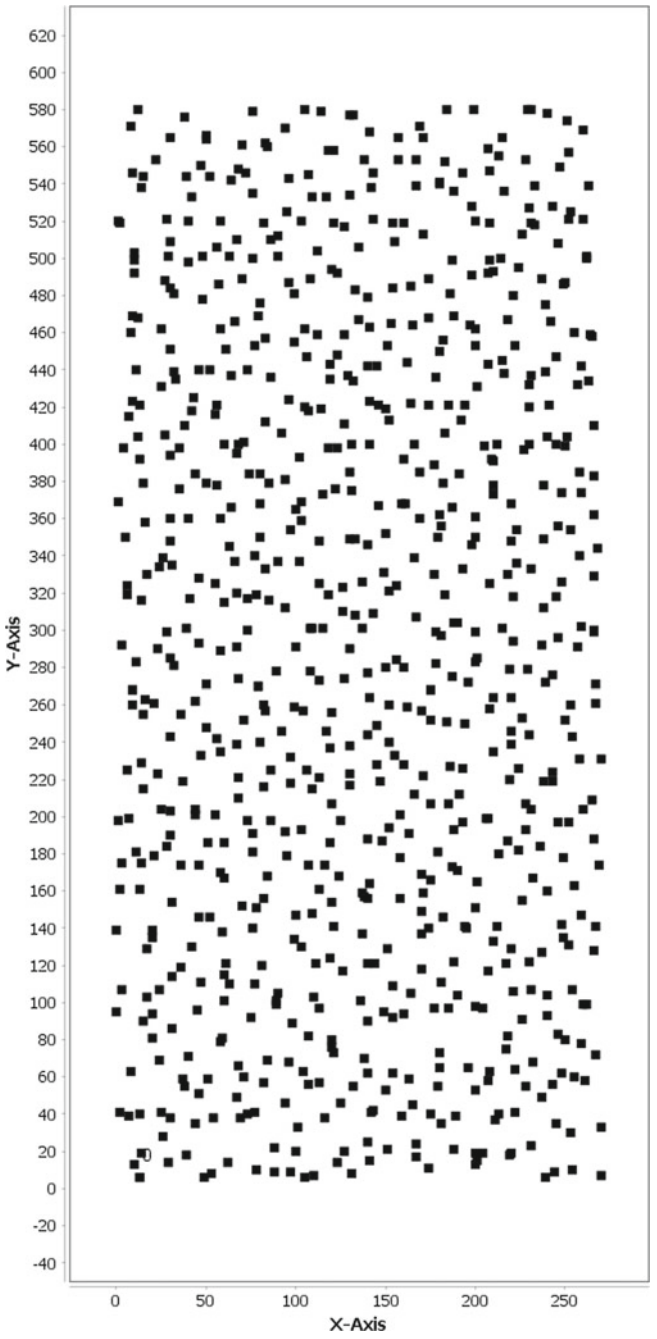

Fig. 5.7 Large benchmark instance with 783 nodes (rat783)

One example of the TOP instances is illustrated above, in Fig. 5.5. This figure shows the instance from Chao-set2-64. This instance will be solved and illustrated in Sect. 5.3 with different values for the number of routes m.

5.2 Exact Approaches

An obvious approach to obtain the optimal solution for an OP or a TOP is to implement the mathematical model presented in Sects. 2.3.1 (OP) or 3.1.1 (TOP) and solve it with a commercial solver. This might work for smaller OP instances with up to around 30 nodes (Tsiligirides-set1-32, set2-21, and set 3-33) and small values of T_{max}, but for larger instances (or higher values of T_{max} or multiple routes) this leads to memory issues and/or extremely long computation times, even with the current computation power available. This confirms the complexity of the problem explained in Chaps. 2 and 3. Nevertheless, several much more efficient exact algorithms have been proposed to solve the OP and the TOP.

5.2.1 Exact Approaches for the OP

The details on the exact OP algorithms can be found in the papers mentioned in Sect. 5.4. Here, we will briefly discuss a branch-and-cut algorithm for the OP. The algorithm is based on five classes of additional inequalities. These inequalities strengthen the LP-relaxation of the OP model and therefore make it easier/faster to find optimal solutions. The first type of inequality is a *logical* constraint, stating when an edge is part of the solution also its corresponding node is visited. The second class inequalities are based on the 2-matching constraints originally formulated for the TSP, adding up the degree of constraints for all nodes in a subset of N and bound constraints. The third one is a *cycle-cover* inequality, using a knapsack constraint to improve the LP relaxation of the OP model. The fourth class are *path* inequalities that exploit both the cycle and knapsack relaxation of the OP problem. The last one considers *conditional cuts*, typically used in a cutting-plane context.

The branch-and-cut algorithm starts from an initial feasible solution which is generated by a heuristic algorithm. The heuristic algorithm works in two stages. In the first stage, a feasible cycle with a large number of edges is detected. Edges are evaluated based on the probability of having edges in an optimal solution. In the second stage, refining procedures, local search improvement moves, are applied to derive a good feasible cycle. At each iteration, 2-opt edge exchanges are performed and followed by the node insertion. These local search moves are discussed in detail in the next section.

The solution obtained by this heuristic algorithm is used as the root node of the branch-and-cut decision tree. It is a feasible solution to the OP and thus also a lower bound. At each node of the branch-and-cut decision tree, the optimal primal and dual solutions of the current LP-relaxations are determined. When the algorithm enters the separation phase, constraint violations are identified. First, the search focuses on the constraint pool, followed by the generalized subtour elimination constraints (GSECs) and the five above-mentioned classes of additional inequalities.

This branch-and-cut algorithm was implemented and tested successfully on the Fischetti instances with a satisfactory performance. In most cases, both upper and lower bound values at the root node are very tight. Actually, for most benchmark instances with less than 400 nodes, the optimal solution is now available. Most recent exact approaches need on average between 3 and 30 min (depending on the type of generation of the profits) to obtain these optimal solutions. The state-of-the-art metaheuristic approaches, discussed in the next section, only need a few seconds to obtain solutions within 1% of the optimal solution.

Figure 5.8 illustrates an example of an instance that is solved by an exact algorithm (with high computational time). The optimal profit is 56. This instance has 100 nodes with a profit of 1 for each node. For the larger instances of Kobeaga-set7397, with more than 400 nodes, only 34 of the 164 instances have been solved to optimality so far. One of these larger instances solved to optimality has 783 nodes and the optimal solution, with $\alpha = 0.5$, is shown in Fig. 5.9. The available travel time allows visiting around half of the nodes. In this case, the profits of the nodes are generally larger further away from the depot. This explains why the effort is done, in this solution, to collect most of the profit far away from the depot.

5.2.2 Exact Approaches for the TOP

For the TOP, only a few exact algorithms have been proposed. These algorithms are mainly branch-and-cut and branch-and-price algorithms. We will again discuss a branch-and-cut algorithm. The algorithm is based on a linear formulation with a polynomial number of binary variables. Due to a large number of subtour elimination constraints in the original TOP formulation, they are removed from the formulation and only added when needed. Moreover, they are replaced by the generalized subtour elimination constraints (GSEC). These generalized constraints ensure that each node served by a particular route is connected to the start node. The algorithm also considers a set of dominance properties and valid inequalities which covers symmetry breaking, boundaries on profits/numbers of customers based on dynamic programming, as well as clique cuts based on the graphs of incompatibilities.

Another exact algorithm is based on the branch-and-price approach. The TOP is first reformulated as a set-packing model (SPM), which is shown in constraints (3.8)–(3.11) in Chap. 3. SPM arises in partitioning applications, where nodes need to

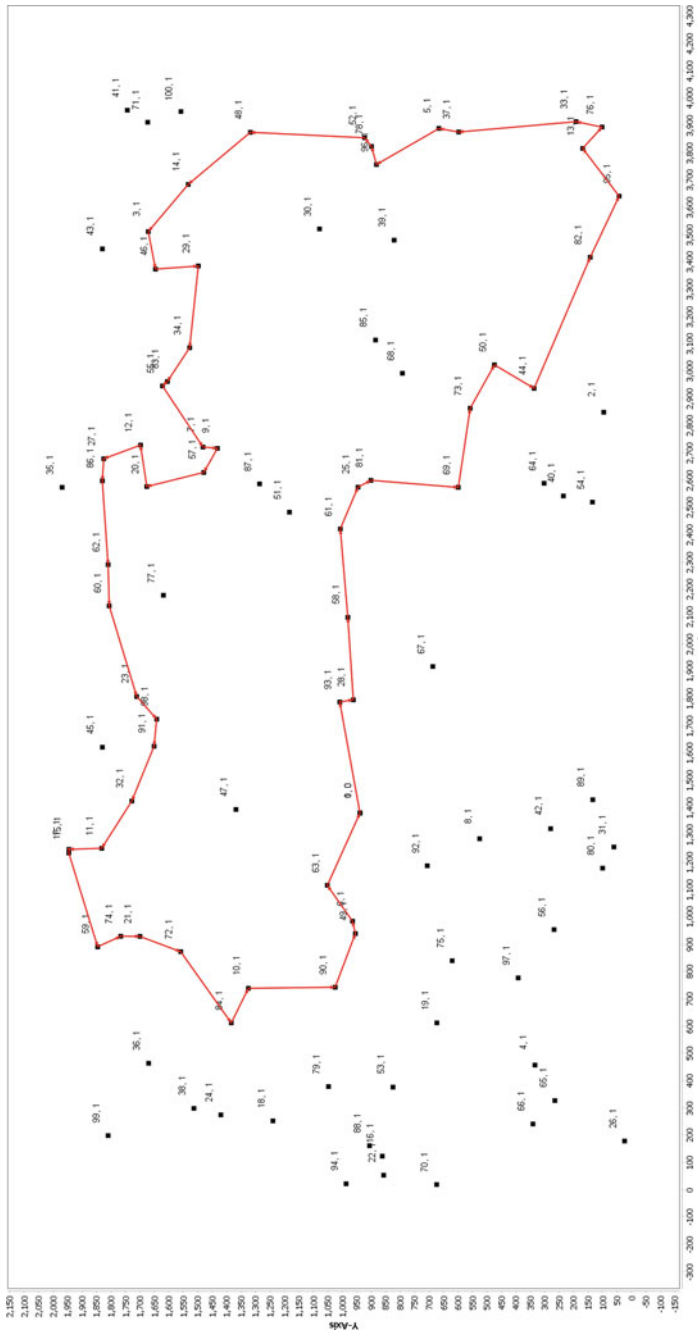

Fig. 5.8 Solution of kroA100 with 100 nodes

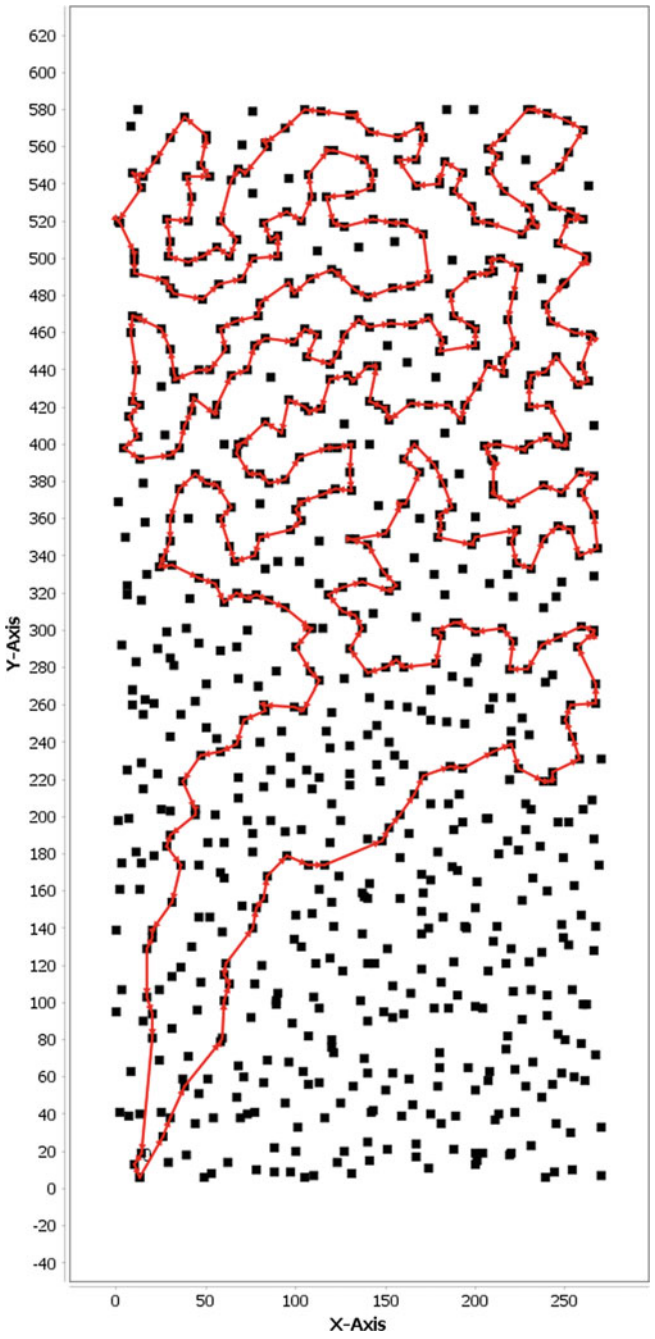

Fig. 5.9 Solution of rat783 with 783 nodes and $\alpha = 0.5$

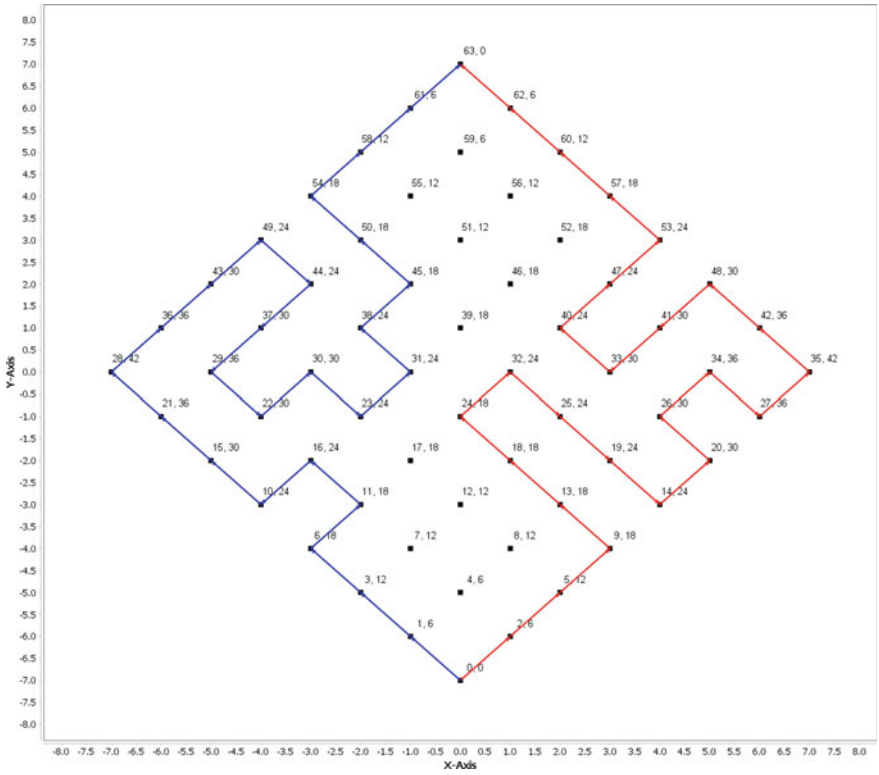

Fig. 5.10 Solution of p.6.2.m with $m = 2$

be partitioned. The key feature of packing problems is that no nodes are permitted to be covered by more than one set, e.g., a route. The upper bound value is calculated by a column generation technique and combined with a branch-and-bound scheme to find an optimal integer solution to SPM. The pricing sub-problem is solved by a bounded bidirectional dynamic programming algorithm with decremental state space relaxation featuring a two-phase dominance rule relaxation. This algorithm is able to solve 301 instances of 387 instances to optimality.

We select the instance shown in Fig. 5.5 to illustrate the optimal solutions for different values of m. Fig. 5.10 uses $m = 2$ and T_{max} is set to 37.5. The total profit collected in these two routes is 1188 and the travel cost is 36.77 in each route. The result for $m = 3$ and $T_{max} = 25$ is shown in Fig. 5.11, with a total profit of 1080, collected in three very different routes. The results of $m = 4$ and $T_{max} = 18.8$, with a total profit of 912, is shown in Fig. 5.12.

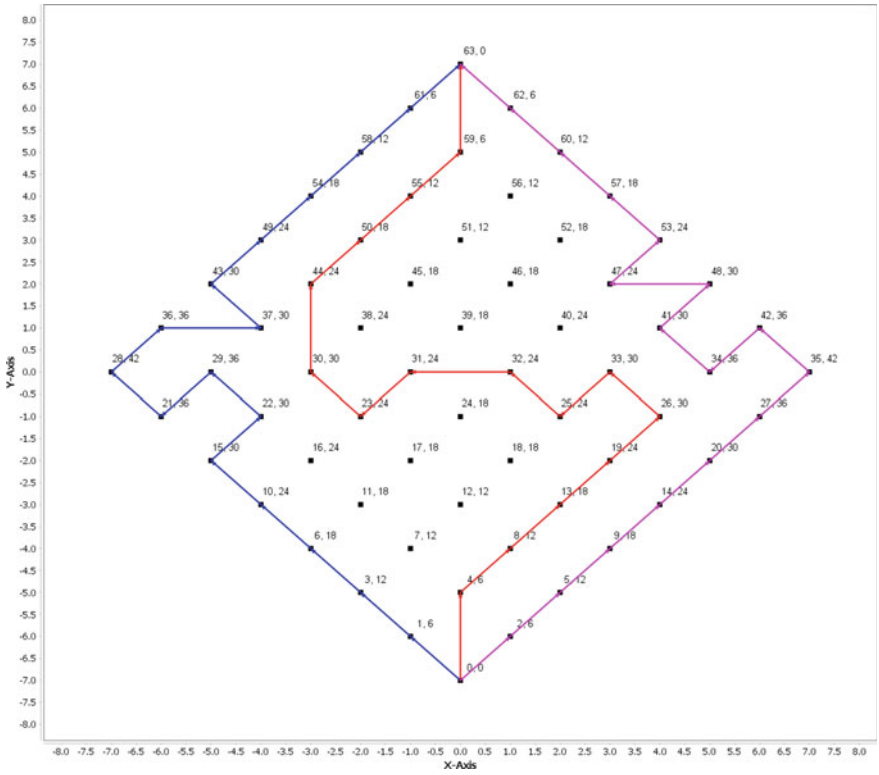

Fig. 5.11 Solution of p.6.3.m with $m = 3$

Based on the performance of above-mentioned exact algorithms, the number of optimal solutions that can be obtained by the branch-and-cut and the branch-and-price approaches are 278 and 301 (out of 387 instances), respectively. This clearly illustrates that for the TOP, in contrast to for the OP, these benchmark instances are still challenging. For the instances that can be solved (within a computation time limit of 2 h), the algorithms need on average around 230 s per instance, although these times obviously fluctuate over different sets of instances.

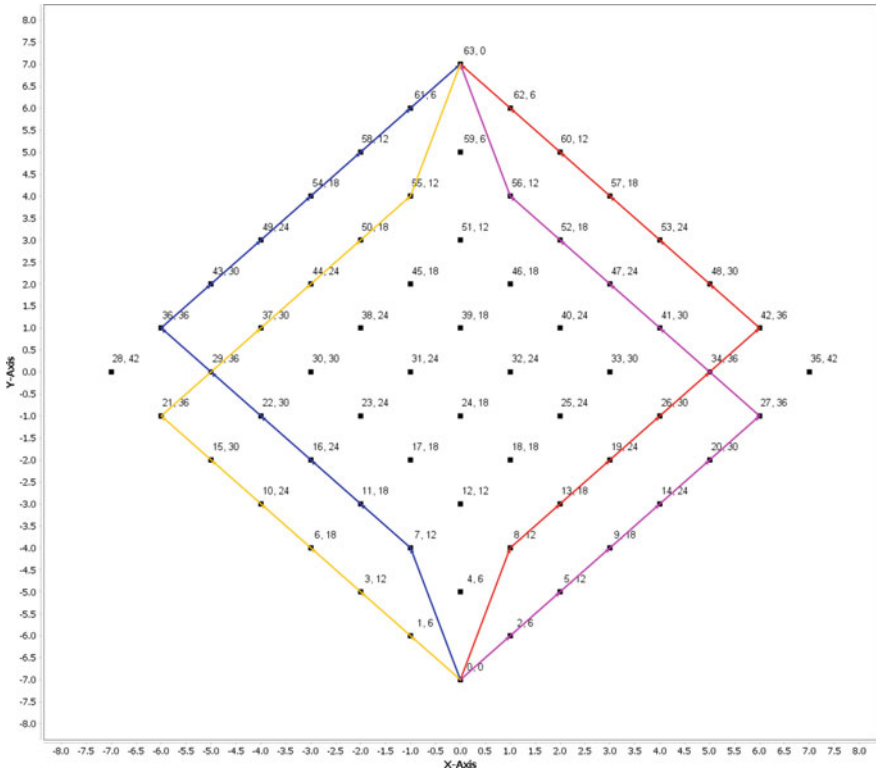

Fig. 5.12 Solution of p.6.4.m with $m = 4$

5.3 Metaheuristics

Several (meta)heuristics have been proposed to solve both the OP and the TOP. For detailed explanations and discussions on the performance, we refer to the papers mentioned in Sect. 5.4. Here, we pick some heuristics and discuss their main features, with the objective of providing insight in the existing solution techniques in order to facilitate the future development of successful (meta)heuristics. Therefore, we start with explaining the most popular local search moves implemented to create or improve high-quality OP and TOP solutions.

5.3.1 Local Search Moves for the OP and the TOP

Obviously, two types of local search moves should be considered: moves that decrease the travel cost required for visiting a subset of nodes and moves that increase the profit by improving the selection of nodes. We will use Fig. 5.13 to illustrate these local search moves.

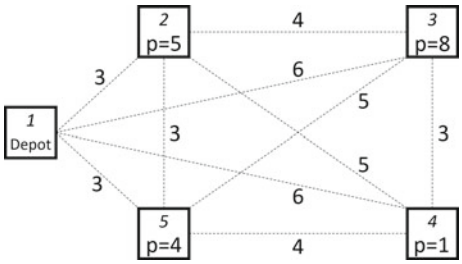

Fig. 5.13 Possible nodes that can be visited and a depot

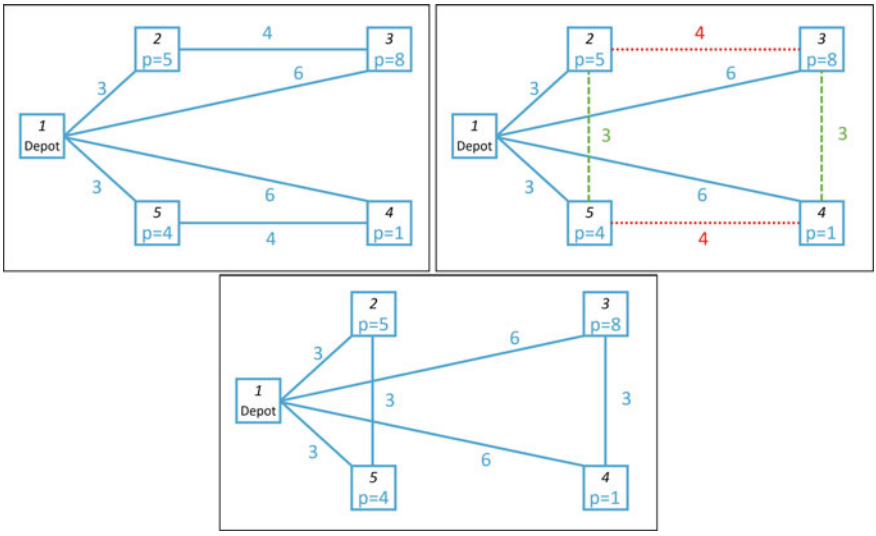

Fig. 5.14 Example of SWAP: node 3 and node 5 swap routes

Local search moves decreasing the travel cost. Typical moves decreasing the travel cost are SWAP, MOVE, RELOCATE, and 2- OPT. The SWAP move exchanges two nodes between two routes. This can only be applied to TOP, unless we want to swap two nodes within a particular route. The move is carried out if total travel cost can be saved. This is illustrated in Fig. 5.14: nodes 3 and 5 are swapped. The total travel cost is reduced from (13 + 13 = 26) to (9 + 14 = 23) travel cost units. Actually, for the TOP, this move could even be considered if the total travel cost would increase, for instance if the travel cost in one route is sufficiently decreased to allow an additional insertion of a node later.

MOVE is only used for the TOP. The idea is to group together the available travel time. It is better to have a lot of time available in one route and no time left in other routes. By doing so, it might become possible to add (an) additional node(s) in the first route. MOVE is done by moving one node from one route to another route. Figure 5.15 presents an example of this move. Node 5 is moved from the first route to the second one, resulting in a reduction of $(21) - (20) = 1$ unit of travel cost. More importantly, however, is that the available travel cost in the first route has

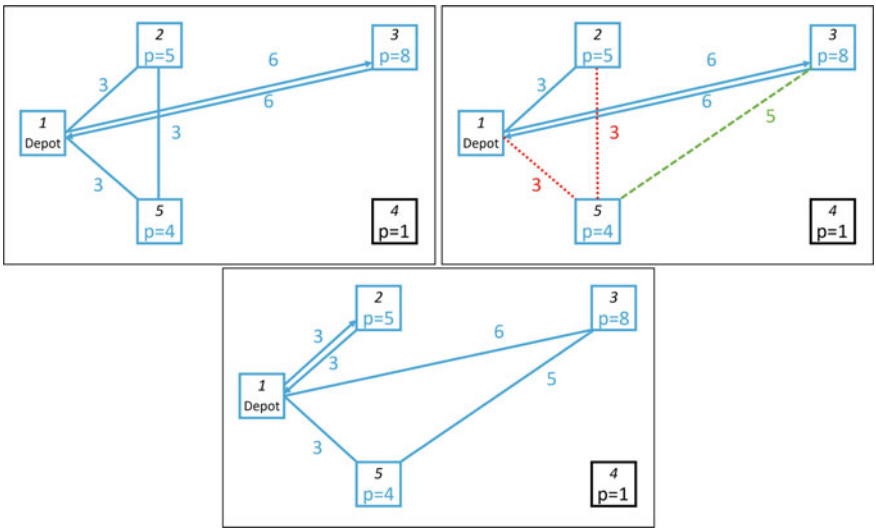

Fig. 5.15 Example of MOVE: node 5 moves from one route to another

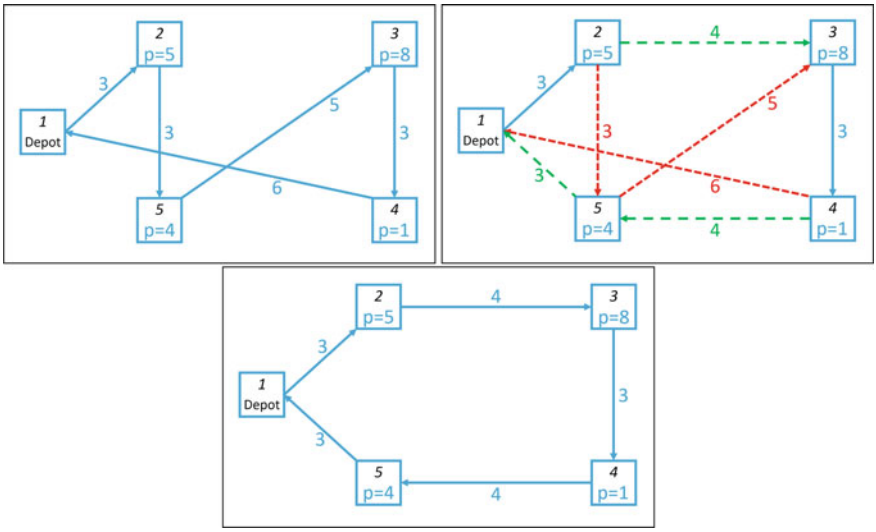

Fig. 5.16 Example of RELOCATE: node 5 moves to a better position in the sequence

increased with 3 units, possibly allowing to include another node in this route later in the solution process.

RELOCATE changes the position of a single node in a route, in order to save travel time. Figure 5.16 presents an example of this move. Node 5 is moved to a better position saving 3 units of travel cost.

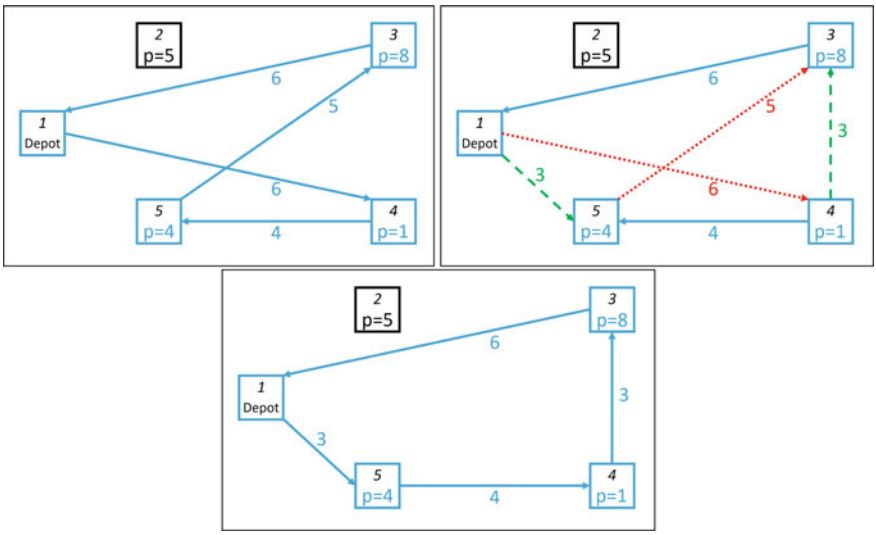

Fig. 5.17 Example of 2- OPT

The last move, 2- OPT, replaces two arcs in the solution by two other, and together shorter, arcs, maintaining a feasible solution. Thereby, the order changes in which the nodes are visited and the travel time is reduced. Figure 5.17 illustrates the 2- OPT move with a reduction of 5 time units. It should be noted that the direction of the arc between nodes 4 and 5 is reversed.

Local search moves increasing the profit. Both INSERT and REPLACE are able to increase the profit by changing the decisions made regarding the subset of visited nodes. INSERT attempts to insert new nodes in the routes in the position where the node consumes the least additional travel time. This move is illustrated in Fig. 5.18. Node 2 is inserted between nodes 1 and 3 with the additional travel cost of $4 + 3 - 6 = 1$ travel cost unit.

REPLACE move seeks to replace a visited node by a non-visited node with a higher profit. It considers one by one the non-visited nodes. First the least time-consuming position for each non-visited node is determined. Then, all non-visited nodes are ranked according to their appropriateness, i.e., a trade-off between its profit and the required travel (and service) time. If the available time budget is insufficient, one (or more) visited node(s), with a lower profit, can be excluded. A replacement will be carried out if the non-visited node with a higher profit can be inserted in the route. Figure 5.19 illustrates one possible REPLACE. We replace node 5 by node 2. The total travel cost is reduced from 14 to 13, but, more importantly, the collected profit is increased by one.

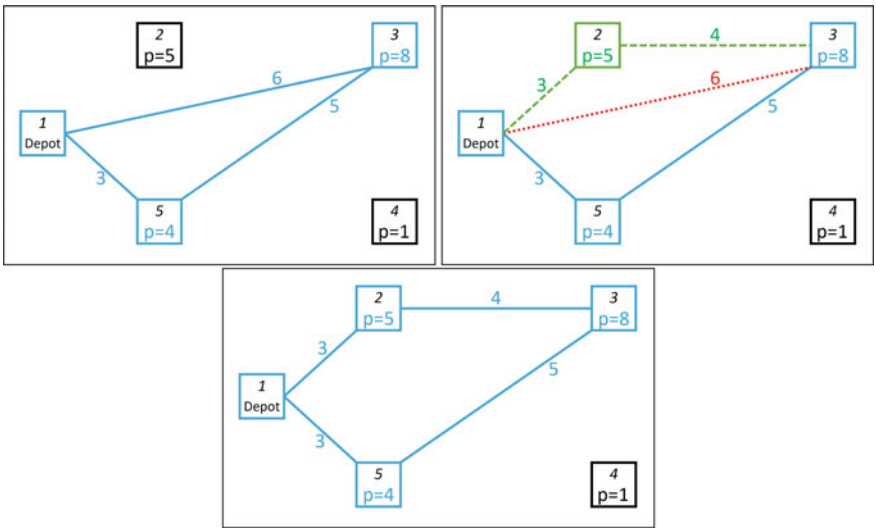

Fig. 5.18 Example of INSERT

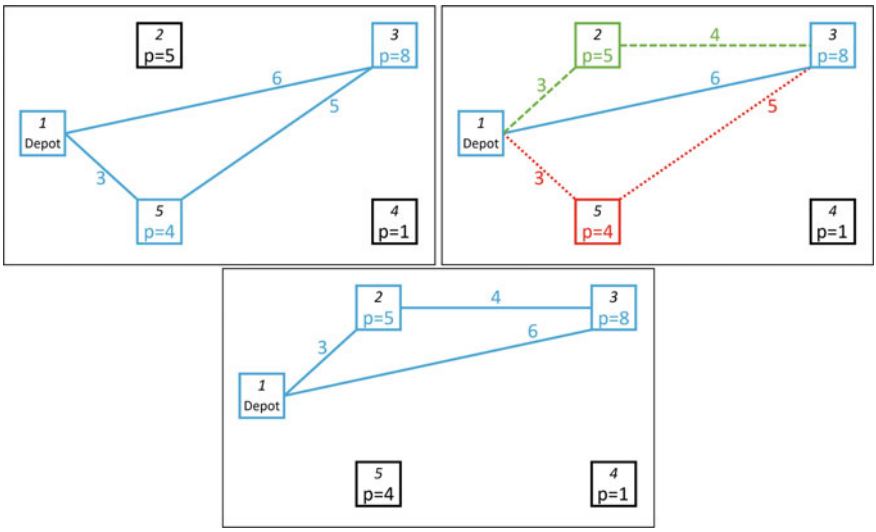

Fig. 5.19 Example of REPLACE

It should be noted that these moves are all called *local* search moves since their impact on the travel cost or on the collected profit can be evaluated *locally*. In the travel cost decreasing moves, only the length of the green arcs need to be compared with the length of the red arcs, the other arcs remain the same. In the small example above, this might seem irrelevant, but also in a solution with 100 or more nodes, these moves can be evaluated by comparing only a few numbers. This limits the required computation time and this is crucial since these moves are considered millions of times when solving the OP or TOP with a (meta)heuristic.

5.3.2 Metaheuristics for the OP

The metaheuristics for the OP can be classified into population-based or single-solution approaches. Population-based heuristics for the OP are mainly based on a Particle Swarm Optimization (PSO) algorithm.

Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm inspired by the intelligent collective behavior of some animals such as flocks of birds or schools of fish. They conform a cooperative way to find food and each member in the swarm, called a particle, flies through the search space with a velocity that is dynamically adjusted according to its own and its companion's historical behavior. The evolution and learning are two basic characteristics of PSO algorithms. The evolution indicates that the solutions are generated and move toward better search spaces during the search process. The 'leader' particles are selected in this evolution process. A particle update its position at each iteration. The velocity update equation consists of three parts, which are previous velocity, cognitive part, and social part. The cognitive part refers to a learning process from its own searching experience and the social part refers to a learning capability from the best of its companions. Like any other algorithm, a PSO algorithms needs to find a balance between exploration (diversification) and exploitation (intensification) during the search for the best solution.

A first algorithm modifies PSO by improving both the exploration and exploitation mechanisms in order to solve the well-known drawbacks of PSO, such as premature convergence and lack of intensification around the local best solutions. These main modifications are done in two ways. First, the exploitation capability of the pioneering-particles is improved by implementing Reduced Variable Neighborhood Search (RVNS). Second, the velocity of the pioneering-particle is updated randomly in order to improve the exploration mechanism and to prevent premature conver-

gence. By doing so, the pioneering-particle continues its search journey from a different position in the following iterations.

One of most recent single-solution heuristics for solving the OP is based on an adaptive large neighborhood search (ALNS) algorithm. Clustering is used as a machine learning method to group together nearby nodes. A popular algorithm in the machine learning community, named the *Dbscan* algorithm, is implemented. This algorithm does not need the number of clusters as a parameter. The clustering is started by defining the radius and the number of core nodes $\hat{N}$. Nodes which have at least $\hat{N}$ neighbors (including themselves) are called core nodes. A node is considered as a neighbor if its distance is not larger than the radius of the core node. Nodes which are neither core nor reachable are considered as outliers. Nodes that are clustered together have a high probability of being included in the route if one of them is included. The clustering approach is also used in the local search moves.

A simple randomized constructive procedure, inserting nodes one by one as long as no constraint is violated, is used to construct an initial solution. Both destroy and repair methods are used to improve the quality of the solutions. RANDOM REMOVE removes a number of selected nodes at random. RANDOM SEQUENCE REMOVE removes a set of nodes that appear consecutively in the route. RANDOM CLUSTER REMOVE removes a set of nodes belonging to a same cluster. Four repair methods are described next. GREEDY REPAIR considers all nodes which are not part of the route. The insertion is done by considering the lowest ratio between the increase in travel time and the increase in collected profit. This is repeated until no more feasible insertion. RANDOM REPAIR is used by taking a fraction of nodes randomly which have not been in the route. The selected nodes will be inserted. PRIZE REPAIR is similar to RANDOM REPAIR. Instead of selecting nodes randomly, we insert a certain fraction of nodes with the highest profit. CLUSTER REPAIR inserts nodes which are close to each other with a small additional travel cost.

Moreover, it should be noted in general that all the above-mentioned local search moves can be implemented in a **first-improvement** or a **best-improvement** fashion. First-improvement means that when these moves are considered for a certain solution, as soon as an improvement is found it will be actually implemented in the solution and then the search for further improvements starts again. On the contrary, best-improvement means that first all possible improvements of a certain kind and for a certain solution are evaluated and then only the best one is actually implemented. Then, the search for further improvements starts again. Typically, first-improvement results in much more, but smaller improvements and best-improvement results in fewer, but higher quality improvements. Which of both is the best for the OP or TOP is impossible to predict and depends on the stopping criterion used, the time required to evaluate and actually implement a modification, etc.

5.3.3 Metaheuristics for the TOP

Various types of solution algorithms are developed for solving the TOP, such as a memetic algorithm (MA), simulated annealing (SA), particle swarm optimization (PSO) and a genetic algorithm (GA). A memetic algorithm is a combination of an evolutionary algorithm and local search (LS) techniques and it has been used for solving the VRP. In the context of the TOP, a genetic algorithm (GA) is selected as one of evolutionary algorithms. Each solution in the GA is encoded by a simple indirect encoding, namely a giant tour. The giant tour is encoded as a sequence, e.g., a permutation of all nodes. A certain number of routes (m) can then be extracted from the giant tour based on the order of the visited nodes in the sequence and the constraints. Three different moves: SHIFT, SWAP, and DESTRUCT-REPAIR are used as mutation operators. SHIFT corresponds to the earlier explained RELOCATE, but now implemented on the giant tour. For each node it is checked if a better position in the giant tour can be found. In SWAP, illustrated in Fig. 5.14, an exchange of all couples of two different nodes in the giant tour is evaluated. DESTRUCT-REPAIR is done by removing a small part of the giant tour sequence with a view to rebuilding an improved solution.

A small number of solutions in the initial population is built by an Iterative Destruction/Construction Heuristic (IDCH) and the rest is generated randomly. New chromosomes are evaluated using the Optimal Splitting procedure. The population is a set of chromosomes for which two criteria are evaluated: the profit and the total travel cost. At the end of the search, the chromosome with the highest profit is treated as the best solution.

Various population-based algorithms based on PSO have been proposed to solve the TOP. We will discuss the latest PSO, namely PSO-inspired algorithm (PSOiA). As explained in Sect. 5.3.2, PSO is a swarm intelligence algorithm with the basic idea of simulating the collective behavior of swarms in nature. Each particle in the swarm has a small probability to be relocated to a completely new position and it also has a probability to be further improved through a local search process. The concept of the giant tour and some of the above-mentioned local search moves are implemented.

Also, a combination of Simulated Annealing and a multi-start hill climbing strategy is able to solve the TOP. The multi-start hill climbing strategy is implemented in order to escape from local optima. Simulated Annealing begins with a randomly generated initial solution. At each iteration, it selects a new solution from the neighborhood of the current solution. A worse solution can be accepted with a small probability. Local search moves, such as SWAP and INSERT, are implemented in order to further improve the quality of the solution.

The last algorithm, Guided Local Search (GLS), combines different local search moves to solve both the OP and the TOP. First, all nodes that cannot be reached anyway are removed from the instance. Then, each route is assigned to one of the nodes which are furthest from the start and end nodes. Next, all remaining nodes are inserted into the routes using cheapest insertion with respect to the available

travel cost. If m routes reach T_{max} and there are remaining non-visited nodes, more routes are generated and the best m routes are selected. In order to further improve the quality of the solutions, local search moves, such as SWAP, MOVE, 2- OPT, Insert, and Replace, are implemented. The GLS framework penalizes, based on a utility function, unwanted solution features during each local search iteration. The penalty augments the objective function during every iteration. In this way the solution procedure may escape from local optima and allow the search to continue.

5.4 Related Literature

A classification of exact algorithms to solve the **OP** is summarized in Feillet et al. (2005). In Fischetti et al. (1998) a branch-and-cut for solving the OP is proposed. A comprehensive review on exact algorithms for the OP can also be found in Vansteenwegen et al. (2011a). However, due to the long computation times of exact algorithms and in order to tackle larger instances, various (meta)heuristics have been proposed. The population-based heuristic for handling the OP, namely PSO, is discussed in detail in Sevkli and Sevilgen (2010a) and Sevkli and Sevilgen (2010b). Other heuristics, such as Memetic-GRASP (MemGRASP), are proposed in Marinakis et al. (2015). For single-solution heuristics, an adaptive large neighborhood search (ALNS) algorithm by incorporating the *Dbscan* clustering algorithm is introduced in Santini (2019). A method which is based on the Greedy Randomized Adaptive Search Procedure (GRASP) and the Path Relinking (PR) methodologies is proposed by Campos et al. (2014).

The recent survey paper by Gunawan et al. (2016) discusses in detail the latest proposed metaheuristics for the OP and many optimal solutions can be found in Kobeaga et al. (2018).

Several exact algorithms have been proposed to solve the **TOP**. A column generation algorithm is introduced in Butt and Ryan (1999). A branch-and-cut algorithm has been proposed in Dang et al. (2013a). Branch-and-price algorithms are proposed in Boussier et al. (2007) and Keshtkaran et al. (2015). The recent paper by Bianchessi et al. (2018) discusses a branch-and-cut algorithm for solving the TOP. A branch-cut and price algorithm is proposed by Poggi et al. (2010).

Metaheuristic approaches for the TOP can be found in Bouly et al. (2010) and Dang et al. (2013b), introducing population-based algorithms based on a memetic algorithm and PSO, respectively. A single-solution algorithm based on simulated annealing was introduced by Lin (2013). The survey paper by Gunawan et al. (2016) discusses in detail different metaheuristics for the TOP.

A number of **benchmark OP instances** are available and can be classified into three groups (Vansteenwegen et al. 2011b). The details can be found in Tsiligirides (1984), Chao et al. (1996a), and Fischetti et al. (1998). In total, there are 385 instances where the number of nodes varies between 21 and 500. They can be accessed via http://www.mech.kuleuven.be/en/cib/op. Several new benchmark instances with

more than 400 nodes and the latest results on both optimal and metaheuristic solutions can be found in Kobeaga et al. (2018).

For the **TOP**, the details on benchmark instances can be found in Chao et al. (1996b) and Dang et al. (2013b). For the latest results we refer the reader to Ke et al. (2016).

References

Bianchessi N, Mansini R, Speranza MG (2018) A branch-and-cut algorithm for the team orienteering problem. Int Trans Oper Res 25:627–635

Bouly H, Dang DC, Moukrim A (2010) A memetic algorithm for the team orienteering problem. 4OR 8(1):49–70

Boussier S, Feillet D, Gendreau M (2007) An exact algorithm for the team orienteering problem. 4OR 5(3):211–230

Butt SE, Ryan DM (1999) An optimal solution procedure for the multiple tour maximum collection problem using column generation. Comput Oper Res 26(4):427–441

Campos V, Martí R, Sánchez-Oro J, Duarte A (2014) GRASP with path relinking for the orienteering problem. J Oper Res Soc 65(12):1800–1813

Chao IM, Golden BL, Wasil EA (1996a) A fast and effective heuristic for the orienteering problem. Eur J Oper Res 88(3):475–489

Chao IM, Golden BL, Wasil EA (1996b) The team orienteering problem. Eur J Oper Res 88(3):464–474

Dang DC, El-Hajj R, Moukrim A (2013a) A branch-and-cut algorithm for solving the team orienteering problem. In: Gomes C, Sellmann M (eds) Integration of AI and OR techniques in constraint programming for combinatorial optimization problems, vol 7874. Lecture Notes in Computer Science. Springer, pp 332–339

Dang DC, Guibadj RN, Moukrim A (2013b) An effective PSO-inspired algorithm for the team orienteering problem. Eur J Oper Res 229(2):332–344

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transp Sci 39(2):188–205

Fischetti M, Salazar-González JJ, Toth P (1998) Solving the orienteering problem through branch-and-cut. INFORMS J Comput 10:133–148

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332

Ke L, Zhai L, Li J, Chan FTS (2016) Pareto mimic algorithm: an approach to the team orienteering problem. Omega 61:155–166

Keshtkaran M, Ziarati K, Bettinelli A, Vigo D (2015) Enhanced exact solution methods for the team orienteering problem. Int J Prod Res 1–11

Kobeaga G, Merino M, Lozano JA (2018) An efficient evolutionary algorithm for the orienteering problem. Comput Oper Res 90:42–59

Lin SW (2013) Solving the team orienteering problem using effective multi-start simulated annealing. Appl Soft Comput 13(2):1064–1073

Marinakis Y, Politis M, Marinaki M, Matsatsinis N (2015) A memetic-GRASP algorithm for the solution of the orienteering problem. In: Le Thi HA, Dinh TP, Nguyen NT (eds) Modelling, computation and optimization in information systems and management sciences, Advances in intelligent systems and computing, vol 360. Springer, pp 105–116

Poggi M, Viana H, Uchoa E (2010) The team orienteering problem: formulations and branch-cut and price. In: Erlebach T, Lübbecke M (eds) 10th workshop on algorithmic approaches for transportation modelling, optimization, and systems (ATMOS'10), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, OpenAccess Series in Informatics (OASIcs), vol 14, pp 142–155

Santini A (2019) An adaptive large neighbourhood search algorithm for the orienteering problem. Expert Syst Appl 123(1):154–167

Sevkli Z, Sevilgen FE (2010a) Discrete particle swarm optimization for the orienteering problem. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2010). Barcelona, Spain, pp 3234–3241, 18–23 July 2010

Sevkli Z, Sevilgen FE (2010b) StPSO: Strengthened particle swarm optimization. Turk J Electr Eng Comput Sci 18(6):1095–1114

Tsiligirides T (1984) Heuristic methods applied to orienteering. J Oper Res Soc 35(9):797–809

Vansteenwegen P, Souffriau W, Van Oudheusden D (2011a) The orienteering problem: a survey. Eur J Oper Res 209(1):1–10

Vansteenwegen P, Souffriau W, Vanden Berghe G, Van Oudheusden D (2011b) The city trip planner: an expert system for tourists. Expert Syst Appl 38(6):6540–6546

Chapter 6
State-of-the-Art Solution Techniques for OPTW and TOPTW

Chapter 3 describes the definitions and mathematical models of the OPTW and the TOPTW in detail. In this chapter, the benchmark instances and the state-of-the-art solution techniques of both problems will be discussed. The solution techniques are classified into two categories: exact approaches and metaheuristic techniques, which is similar to the classification of solution techniques for the OP and the TOP. In this chapter we discuss the most important characteristics of these solution approaches and some crucial insights in their effectiveness. For more details, readers are referred to the original papers mentioned in the last section.

6.1 Benchmark Instances

A number of benchmark OPTW and TOPTW instances are available from many different research papers. Most benchmark instances are available in a digital format on this web page: http://www.mech.kuleuven.be/en/cib/op.

The first two sets of benchmark instances for the team orienteering problem with time windows, Righini-Sol-100-50 and Righini-Sol-100-100, are generated from the well-known Solomon benchmark instances of the vehicle routing problem with time windows (VRPTW). Each set is composed of 29 instances, one with the first 50 nodes and the other with the first 100 nodes of Solomon's instances. Each set contains three different categories of instances depending on the location of the nodes: random, clustered, and random-clustered categories. Clustered means that the locations of the nodes are determined in such a way that the nodes are grouped in different clusters. Inside each of these three categories, the nodes are the same and have the same profits, but the time windows are generated in different ways.

Another set of 27 instances with 100 nodes based on Solomon's instances, Montemanni-Sol-200-100, was developed in the same way, but with a larger time budget and larger time windows for the nodes.

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_6

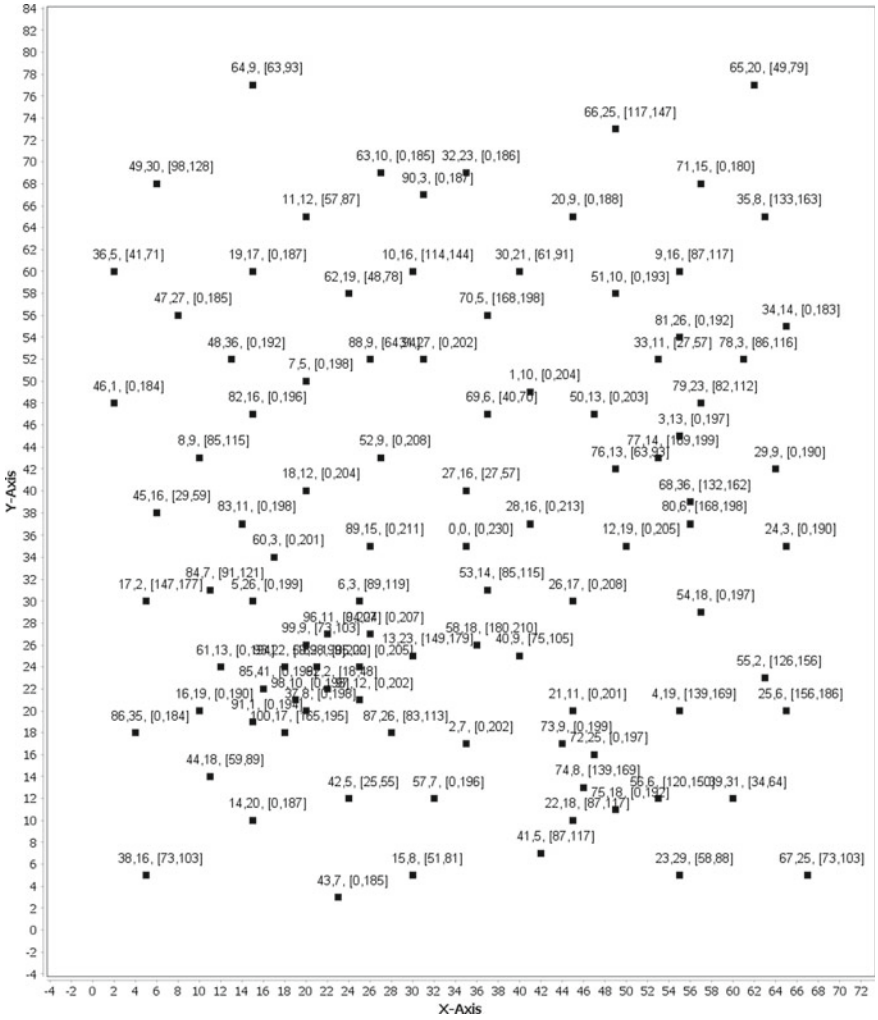

Fig. 6.1 Benchmark instance with randomly located nodes (r107)

The number of routes (m) is not included in these benchmark instances. This value typically varies between 1 and 4. The time budget per route, indicated by T_{max}, equals the closing time of the start node (with index 0). Figures 6.1, 6.2, and 6.3 illustrate three instances from the above-mentioned categories. For each node, we mention the index, the profit, and the time window. The (central) node with index 0 (and profit 0) is the start (and end) node.

Another two sets of 10 instances each are derived from Cordeau's dataset with 20 instances for the Multi Depot Periodic Vehicle Routing Problem (MDPVRP). These sets, Righini-Cor-1-10 and Righini-Cor-11-20, consider all customers that are active

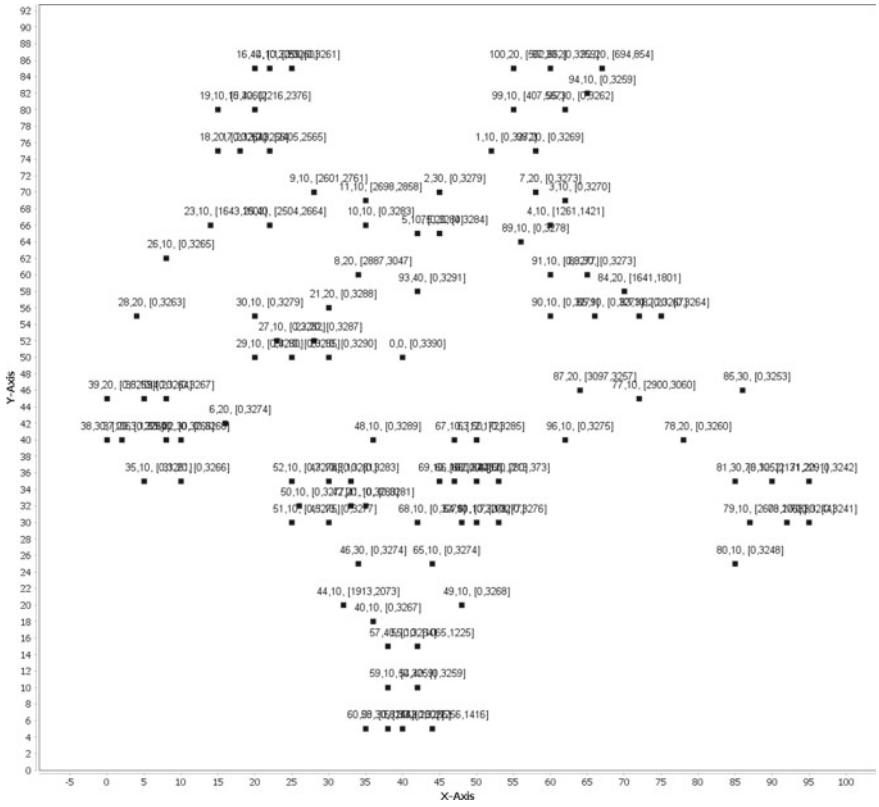

Fig. 6.2 Benchmark instance with clusters of nodes (c204)

on the same day. The demand of a particular node is considered as the profit for that node. The number of nodes in these instances varies from 48 to 288 nodes and they typically have larger time windows. Therefore, these instances are harder to solve. T_{max} is set to 1000 min (closing time of the start node) and the average duration of the time windows is equal to 135 min for the first set of 10 instances and equal to 269 min for the second set of 10 instances. All instances are solved for the number of routes from 1 (OPTW) to 4 (TOPTW). Figure 6.4 illustrates one example of these instances. Clearly, the number of nodes (144) is much higher than in the instances shown above.

Two sets with more difficult instances, Vansteenwegen-Sol-100-100 and Vansteenwegen-Cor-1-10, have also been proposed based on the original sets from Solomon and Cordeau. In these TOPTW instances, the number of routes was set equal to the number of vehicles in the original instances. The advantage of these sets is that the optimal solution is known, since it is possible to visit all nodes and therefore collect all the available profits. For most of the other benchmark sets the optimal solution is still not available. Obviously, the fact that all nodes can be visited in these

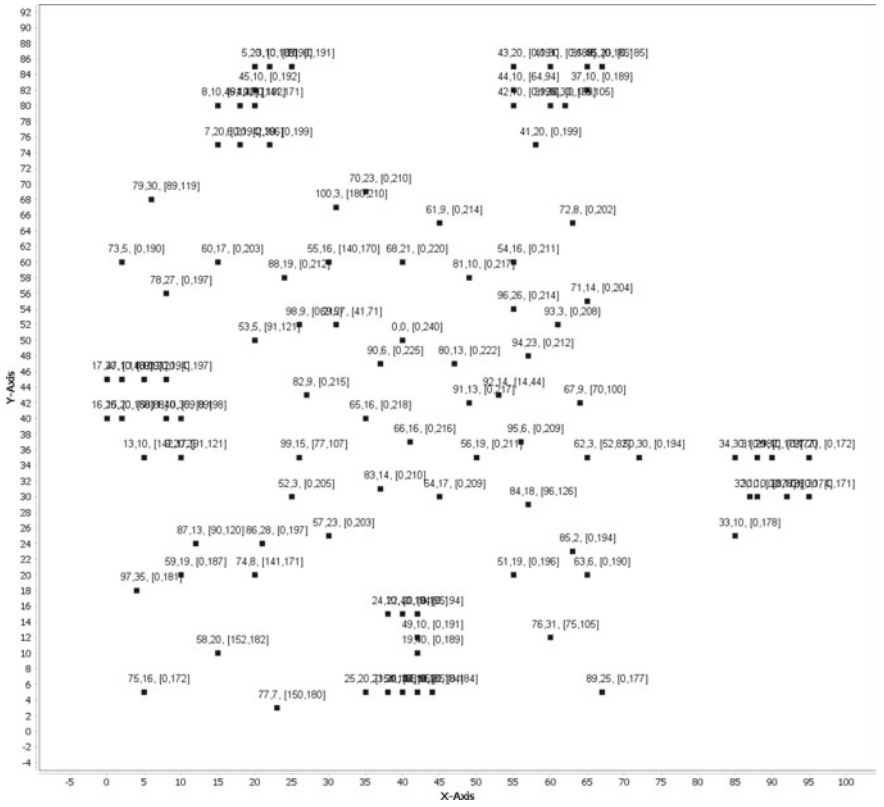

Fig. 6.3 Random Clustered benchmark instance (rc104)

instances, should not be taken into account when designing a solution approach for the general TOPTW. In these sets, the number of nodes varies between 48 and 288 and the number of routes varies between 2 and 20.

> In all the above mentioned OPTW and TOPTW instances, the Euclidean distance is used and rounded down to the first decimal for the Solomon instances and to the second decimal for the Cordeau instances.

The latest set of instances reflects the real-life Tourist Trip Design Problem (TTDP), namely t^* instances, wherein: (a) Point of Interest (POIs) have wider, overlapping, and multiple time windows; (b) POIs are densely located at certain areas; (c) the profit value is associated with the visiting time (or service time) at a POI; and (d) the daily time budget is typically in the order of a few hours per day. 100 new instances are created with the following characteristics: (a) the number of routes is from 1 to 3; (b) the number of nodes is from 100 to 200 nodes, where 80% of them are located around 1–10 zones; (c) the visiting time at any node is from 1 to 120 min

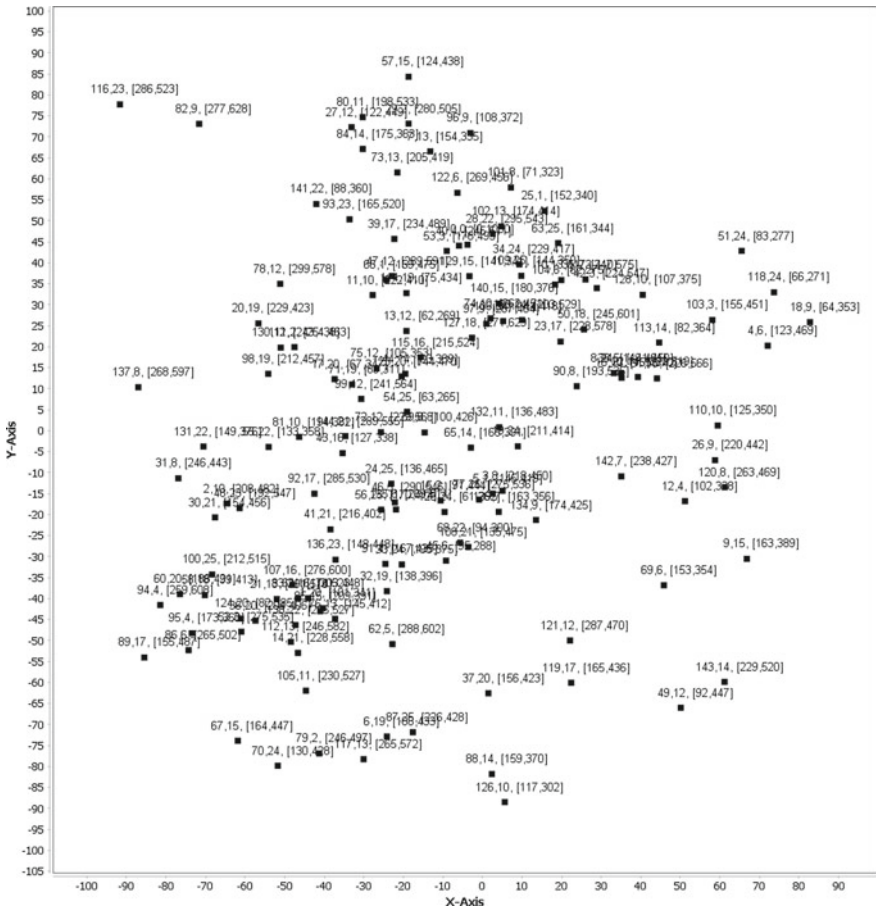

Fig. 6.4 A Righini-Cor-11-20 benchmark instance with 144 nodes (pr18)

and proportional to the profit; (d) 50% of the nodes are open on a 24 h basis, while the remaining nodes are closed either for one or two days. During their opening days, these attractions are open between 08.30 am and 5.30 pm; and (e) the daily time budget is set to either 5 or 10 h.

6.2 Exact Techniques for the OPTW

In this section, we briefly discuss the latest exact algorithm for solving the OPTW. Only the main features will be discussed. For more details on this and other algorithms, we refer the reader to the papers mentioned in Sect. 6.4.

This algorithm is based on bi-directional and bounded dynamic programming (DP) with decremental state space relaxation (DSSR). The OPTW is reformulated as the resource constrained elementary shortest path problem (RCESPP) that can be effectively solved by DP with DSSR. The basic idea of DP is that it repeatedly extends each state to generate new states. In the context of the OPTW, states refer to tuples of the form (S, τ, P, i) where S is a binary vector representing the subset of visited nodes, τ is the overall time elapsed, P is the total collected profit and i is the last reached node. The idea of bi-directional DP focuses on both forward from node 1 and backward from node N. The search is bounded by a certain state to ensure that the state belongs to an optimal path. A critical resource is identified and no path (or route) is allowed to exceed a consumption of the critical resource equal to half of the quantity of the available resource, e.g., time budget. In order to compute shortest paths with resource constraints, extension rules, dominance tests, and matching procedures of forward and backward paths are used.

The concept of DSSR is implemented to reduce the number of states to be explored. In the OPTW, state space relaxation replaces the binary vector S with the number of visited nodes. At each iteration of the DSSR algorithm, the relaxed original problem is solved in order to provide an upper bound to the optimal solution. The proposed algorithm is tested on benchmark instances, showing that it significantly reduces the number of iterations and the computational time required to converge to an optimal solution. It is able to solve 27 of the 29 Righini-Sol-100-100 instances and 8 of the 10 Righini-Cor-1-10 instances to optimality.

> Due to the complexity of the TOPTW, no exact approaches have been proposed for solving the TOPTW. On the other hand, a number of (meta)heuristic techniques have been proposed for both the OPTW and the TOPTW. More explanations will be described in the next section.

6.3 Metaheuristics

A number of metaheuristics has been proposed to solve the OPTW and the TOPTW, e.g., iterated local search (ILS), simulated annealing (SA), ant colony optimization (ACO), and an artificial bee colony algorithm (ABC). Most recent and successful algorithms are based on ILS and a hybridization of SA and ILS that are now described in more detail. Figure 6.5 illustrates the network that will be used in this section to clarify some solution steps.

The ILS algorithm starts by generating an initial feasible solution using a greedy construction heuristic. It starts with an empty route only including the start node or depot, node 1 in our example. Then, a set with all feasible candidate nodes that can be visited is generated.

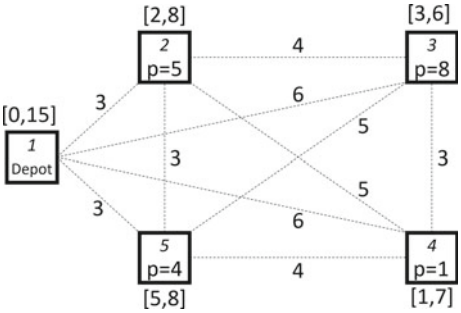

Fig. 6.5 Four possible nodes to be visited and a depot, all with time windows

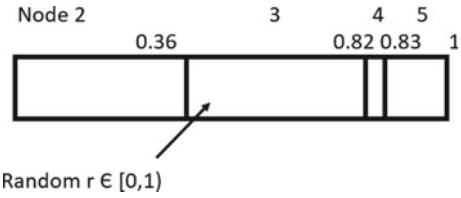

Fig. 6.6 Illustration of the Roulette-Wheel selection

Non-visited nodes to be inserted are selected by using the roulette-wheel selection with respect to the attractiveness of the insertion. The attractiveness is calculated by dividing the square of the profit by the difference between the total time spent before and after the insertion of a particular node. Due to the time windows, the time consumption of an insertion becomes less relevant than the score when deciding which visit is the most attractive to insert next. Therefore, the square of the score is applied in the calculation of the attractiveness.

The higher the value is, the more attractive the node is and the higher the chance it will be visited. For our example, the ratios for nodes 2, 3, 4, and 5 are $\frac{5^2}{3+3} = 4.17$, $\frac{8^2}{6+6} = 5.33$, $\frac{1^2}{6+6} = 0.08$, and $\frac{4^2}{5+3} = 2$, respectively. Take note that node 5 can only be visited at time $= 5$, therefore, the difference between the total time spent before and after is not $3 + 3$, but $5 + 3$, where 3 represents the journey back to the depot.

Then a fitness function is used to assign a fitness level to possible nodes. This fitness level is used to associate with each individual node, a probability of being selected. Therefore, the fitness values for nodes 2, 3, 4, and 5 are $\frac{4.2}{4.2+5.3+0.1+2} = 0.36$, $\frac{5.3}{4.2+5.3+0.1+2} = 0.46$, $\frac{0.1}{4.2+5.3+0.1+2} = 0.01$, and $\frac{2}{4.2+5.3+0.1+2} = 0.17$, respectively. The cumulative probability distribution (CDF) over the list of possible nodes is generated, as shown in Fig. 6.6. A uniform random number from the range [0, 1) is generated. The node with a larger width has a higher chance of being the next one to be chosen and visited. Node 4, with the lowest fitness value, has the lowest probability to be selected. The selected node is then inserted. The entire algorithm is repeated until no more nodes can be inserted.

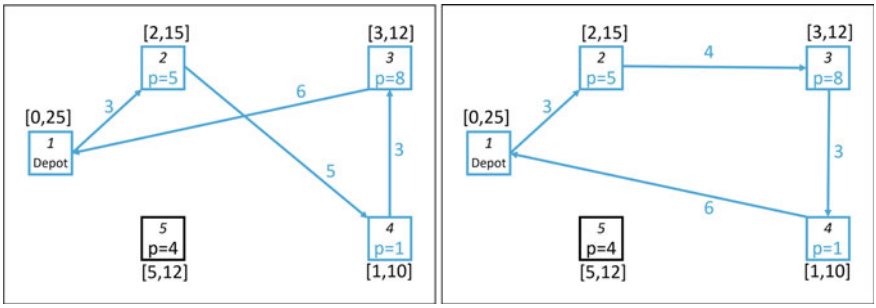

Fig. 6.7 Example of SWAP1

The initial solution obtained is further improved by ILS. We discuss three components of ILS: PERTURBATION, LOCALSEARCH, and ACCEPTANCECRITERION. PERTURBATION is applied in order to escape from local optima. At the end of each iteration, one of two strategies is applied according to the number of iterations already performed. Either the order of the routes in the solution is changed by swapping two adjacent routes or a certain number of nodes is removed from each route.

In LOCALSEARCH, six different local search moves, which are grouped into two groups, are run consecutively. The first group, SWAP1, SWAP2, 2- OPT, and MOVE, only restructures the current solution by decreasing the travel cost while the second one, (INSERT and REPLACE), focuses on increasing the profit.

Local search moves decreasing the travel cost. SWAP1 swaps two visited nodes within one selected route. Figure 6.7 illustrates SWAP1. The initial route (nodes 1-2-**4**-**3**-1) has the travel cost of $3 + 5 + 3 + 6 = 17$, while after applying SWAP1, the travel cost becomes $3 + 4 + 3 + 6 = 16$ with the route of nodes 1-2-**3**-**4**-1.

SWAP2 extends the idea to two different routes (see Fig. 6.8). Two routes (nodes 1-2-**3**-1 and 1-**5**-4-1) are selected with the total cost of $(3 + 4 + 6) + (3+ \mathit{2} + 4 + 6) = 28$. As can be observed, the second route requires a waiting time of 2 when visiting node 5. After swapping two nodes (highlighted in bold above), the total cost is decreased to $(3 + 3 + 3) + (6 + 3 + 6) = 24$. But what is more important in this case, is that in the new route 1-2-5-1 much more time is available to include an additional node.

2- OPT is applied by enumerating all possible combinations of selecting two different nodes from the route with the highest total travel cost. The order of these nodes (and nodes in between) is reversed if this causes no time window violation and the travel cost is decreased. One possible 2- OPT is illustrated in Fig. 6.9. The first route with the order of nodes 1-**5**-**3**-**2**-1 has the travel cost of $(3+ \mathit{2} + 5 + 4 + 3) = 17$. As it can be observed, this route requires a waiting time of 2 when visiting node 5. If nodes 5 and 2 are selected for the 2- OPT, the order of visited nodes become 1-**2**-**3**-**5**-1 with the total travel cost of $(3 + 4 + 5 + 3) = 15$. By reversing the order in which nodes 2, 3, and 5 are visited, the travel cost is reduced by 2.

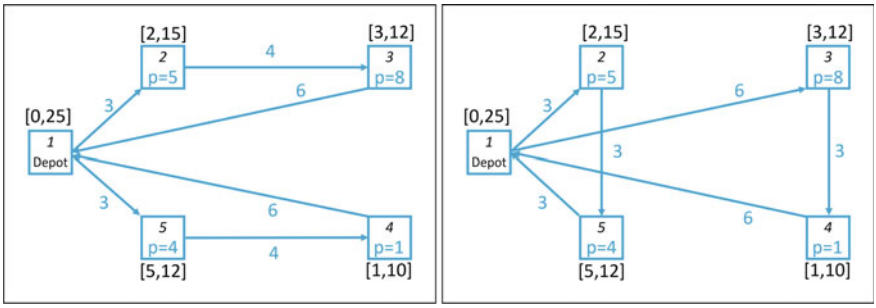

Fig. 6.8 Example of SWAP2

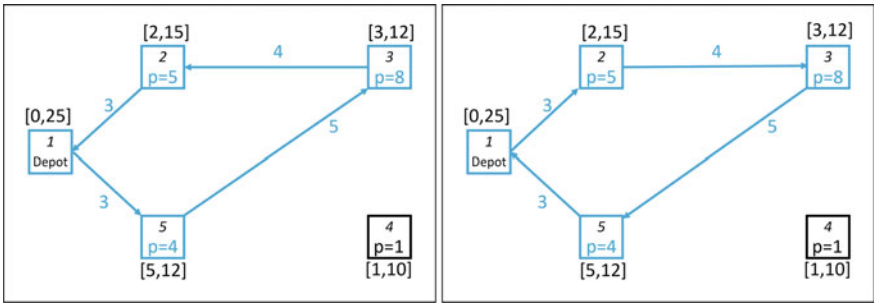

Fig. 6.9 Example of 2- OPT

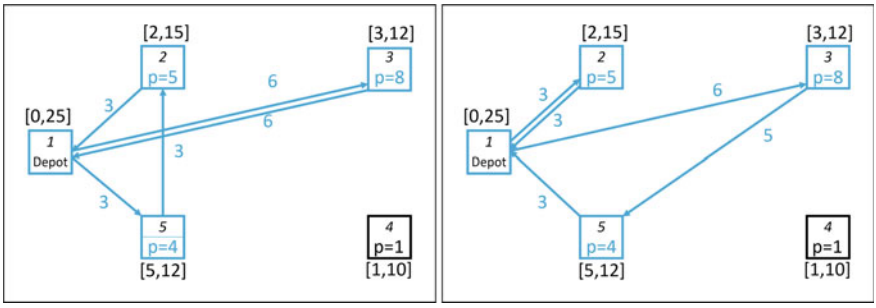

Fig. 6.10 Example of MOVE

MOVE is performed by reallocating a node from one route to another. We explain this in Fig. 6.10. Node 5 from the first route (1-**5**-2-1) is selected to be moved to another route. We consider whether node 5 is reallocated before or after node 3. The selection is done by using the Roulette-Wheel selection, as explained earlier. It turns out that node 5 is moved after servicing node 3 in the second route. The total costs before and after this move are $(3+2+3+3)+(6+6)=23$ and $(3+3)+(6+5+3)=20$, respectively.

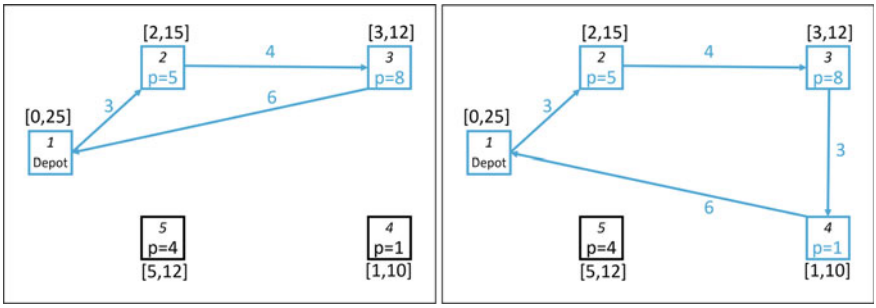

Fig. 6.11 Example of INSERT

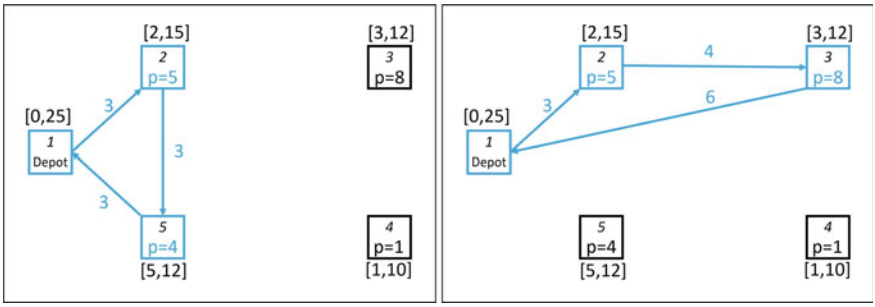

Fig. 6.12 Example of REPLACE

Local search moves increasing the profit. The moves increasing the profit, namely INSERT and REPLACE, change the subset of visited nodes. INSERT is performed by inserting one non-visited node to a particular route while REPLACE replaces one visited node with an unvisited node. Both will be explained using Figs. 6.11 and 6.12.

Figure 6.11 illustrates the insertion of node 4. A set of possible nodes that can be inserted including their positions is generated. For this example, possible insertions for node 4 are 1-4-2-3-1, 1-2-4-3-1 or 1-2-3-4-1. The selection between these possible insertions is done by using the Roulette-Wheel selection. By servicing node 4, the total profit increases from 13 to 14 with an additional travel cost of 3.

The last local search move, REPLACE, replaces one visited node with an unvisited node. The move is started by selecting the route with the lowest travel cost. By doing so, the amount of unused total travel cost is increased. A non-visited node is selected with the highest profit and it is evaluated by checking if it can replace one node from the selected route. Figure 6.12 illustrates an example of this move. The current route visits nodes 2 and 5 and the selected non-visited node is node 3. Node 3 can replace either nodes 2 or 5. In this case, we choose node 5 since its profit is lower. By replacing node 5 with node 3, the total profit increases from 9 to 13 with an additional travel cost of 4.

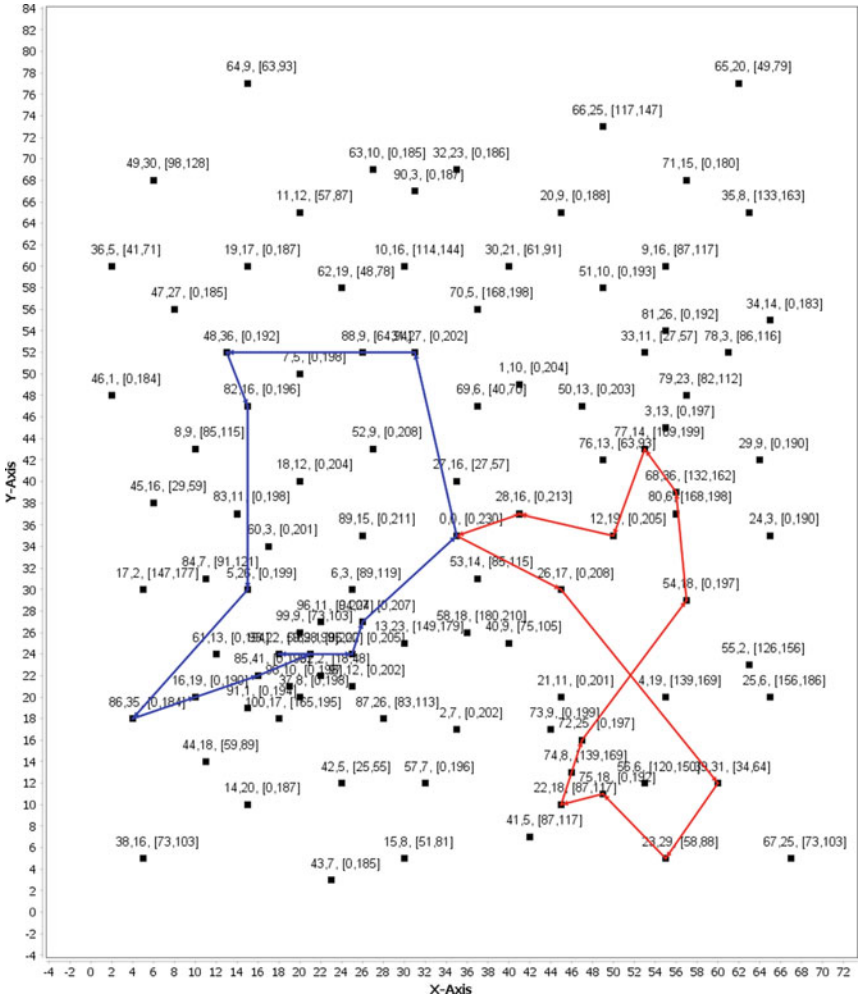

Fig. 6.13 Best known solution for instance r107 with $m = 2$

It should be noted that, due to the time windows, it becomes much more difficult to *locally* evaluate the impact of these *local* search moves. Inserting a node, removing a node, changing the order in which nodes are visited, they all change the moment the nodes are visited and therefore they might all lead to time window violations for all nodes visited after the first change in the route. This significantly increases the computation time of evaluating local search moves. That is why the concept of MAXSHIFT was introduced in this algorithm. MAXSHIFT keeps track of how much a certain visit can be shifted in time, without violating any time window in the route. It allows to significantly reduce the computation time of the different local search moves.

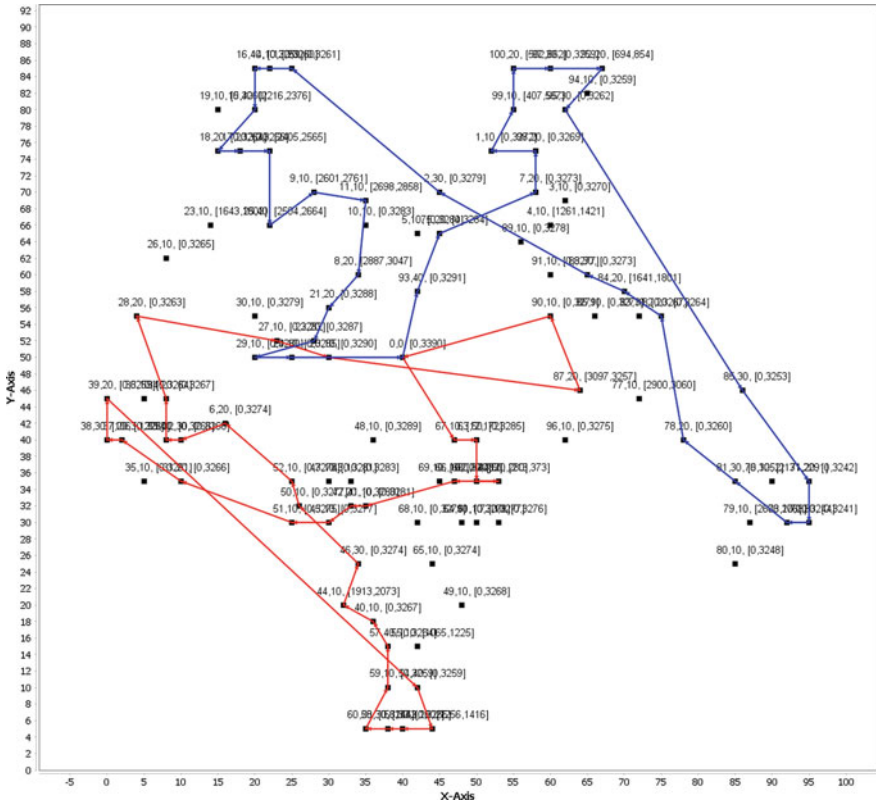

Fig. 6.14 Best known solution for instance c204 with $m = 2$

The last component of this ILS, AcceptanceCriterion, determines whether the search always continues from the current solution or from the best solution found. The strategy in this ILS is to diversify the search and continue from the current solution, until a certain number of iterations without improvement, then the algorithm will be restarted from the best found solution.

Another metaheuristic is a hybrid algorithm that combines simulated annealing (SA) and ILS. The basic concept is the same of the above-mentioned ILS. However, by including the idea of SA and accepting a worse solution with a probability that changes over time, higher quality solutions can be achieved. Both proposed algorithms are tested on benchmark TOPTW instances. Both are able to improve 50 best known solution values on the available benchmark instances.

Four recent best known solutions are illustrated in Figs. 6.13, 6.14, 6.15, and 6.16, taken from different categories of benchmark instances. Those instances are r107 ($m = 2$), c204 ($m = 2$), rc104 ($m = 3$), and pr18 ($m = 3$). The total profits are 538, 1490, 835, and 1282, respectively. Due to the time windows, these TOPTW solutions look less well-organized than the OP and TOP solutions in the previous chapter.

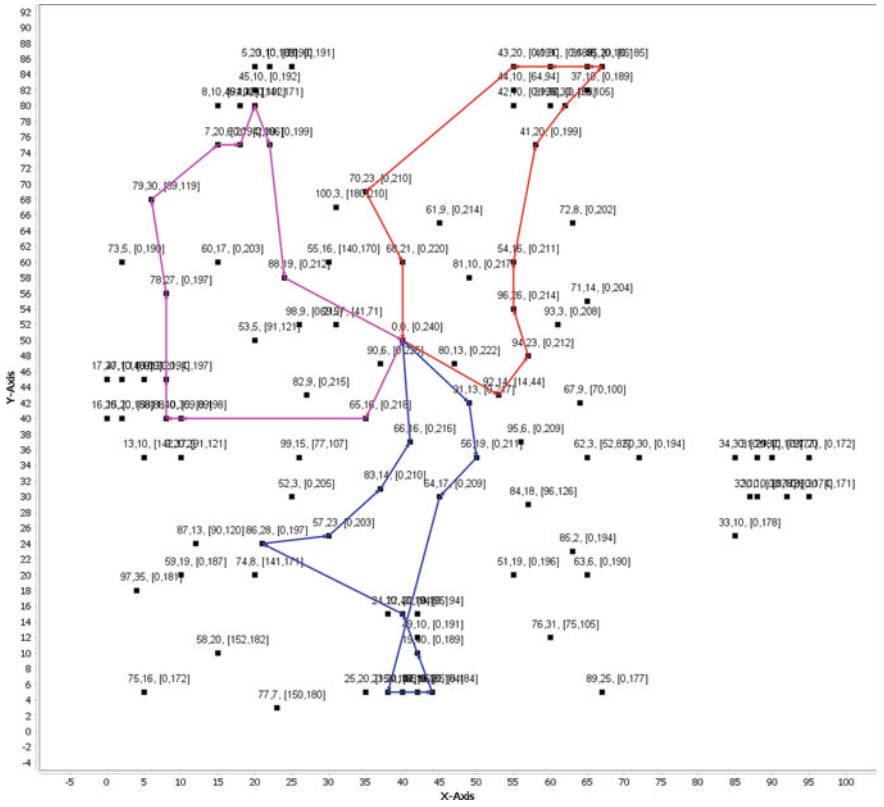

Fig. 6.15 Best known solution for instance rc104 with $m = 3$

6.4 Related Literature

The latest survey by Gunawan et al. (2016) provides the most recent algorithms for solving the OPTW and the TOPTW. Recent benchmark instances are also summarized. The earlier survey by Vansteenwegen et al. (2011a) mentions the first work proposed to solve the OPTW (Kantor and Rosenwein 1992).

Most benchmark OPTW and TOPTW instances are available at https://www.mech.kuleuven.be/en/cib/op. They can be categorized into three groups (Gunawan et al. 2016). The first two groups consists of 105 instances with the number of nodes varying from 48 to 288. The third group (Vansteenwegen et al. 2009), which contains between 48 and 288 nodes, is considered more difficult. In Gavalas et al. (2013), a fourth set of instances based on the TTDP is proposed.

An exact algorithm based on bi-directional dynamic programming is proposed in Righini and Salani (2009) to solve OPTW instances to optimality. The idea of DSSR for reducing the relaxation of the state space iteratively (Righini and Salani 2008) is implemented.

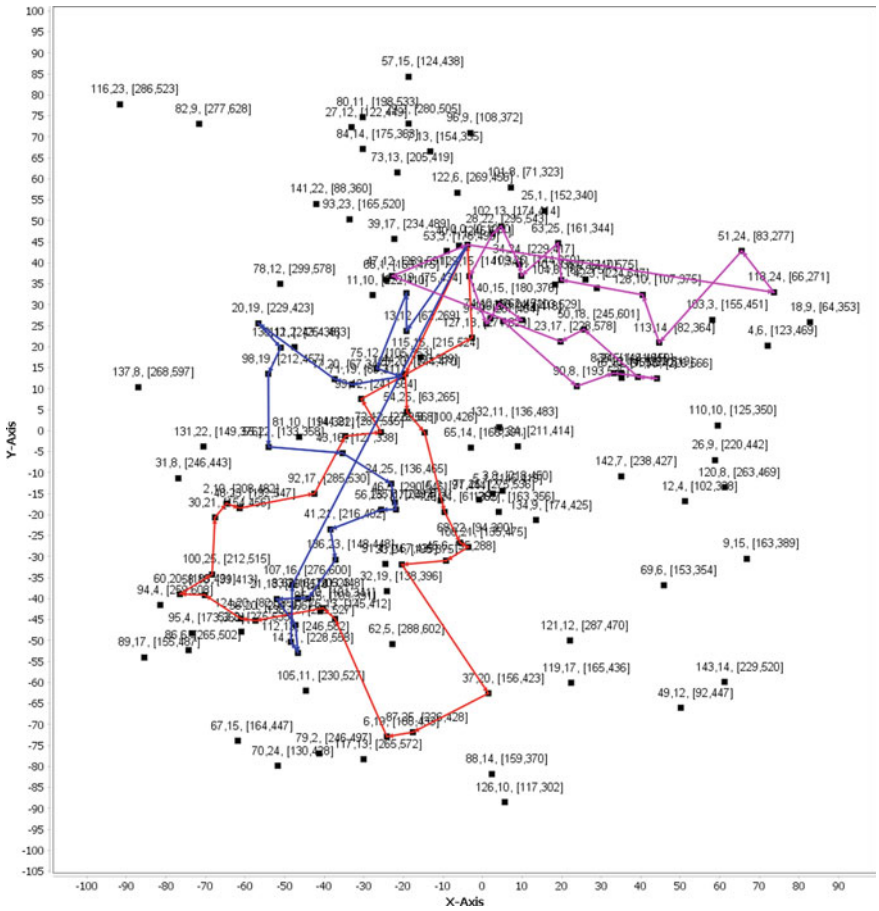

Fig. 6.16 Best known solution for instance pr18 with $m = 3$

In Vansteenwegen et al. (2009), a very fast ILS algorithm to solve the TOPTW is proposed. A population based algorithm, the ant colony system (ACS), is introduced by Montemanni and Gambardella (2009). Many other metaheuristics are available as well: a hybridization of a greedy randomized adaptive search procedure (GRASP) and an evolutionary local search (ELS) algorithm (Labadie et al. 2011), Simulated Annealing (SA) (Lin and Yu 2012), a hybrid algorithm based on GRASP and ILS (Souffriau et al. 2013), an LP-based granular variable neighborhood search (GVNS) (Labadie et al. 2012), and an artificial bee colony (ABC) approach (Cura 2014). Moreover, an iterative framework (I3CH), which is based on two components, a local search (LS) procedure and SA, is proposed by Hu and Lim (2014). Finally, a well-tuned ILS and a hybridization of SA and ILS are proposed by Gunawan et al. (2017).

References

Cura T (2014) An artificial bee colony algorithm approach for the team orienteering problem with time windows. Comput Ind Eng 74:270–290

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G, Tasoulas Y (2013) Cluster-based heuristics for the team orienteering problem with time windows. In: Bonifaci V, Demetrescu C, Marchetti-Spaccamela A (eds) Experimental algorihtms, vol 7933. Lecture Notes in Computer Science. Springer, pp 390–401

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332

Gunawan A, Lau HC, Vansteenwegen P, Kun L (2017) Well-tuned algorithms for the team orienteering problem with time windows. J Oper Res Soc 68(8):861–876

Hu Q, Lim A (2014) An iterative three-component heuristic for the team orienteering problem with time windows. Eur J Oper Res 232(2):276–286

Kantor MG, Rosenwein MB (1992) The orienteering problem with time windows. J Oper Res Soc 43(6):629–635

Labadie N, Mansini R, Melechovskỳ J, Calvo RW (2011) Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. J Heuristics 17(6):729–753

Labadie N, Mansini R, Melechovskỳ J, Calvo RW (2012) The team orienteering problem with time windows: an lp-based granular variable neighborhood search. Eur J Oper Res 220(1):15–27

Lin SW, Yu VF (2012) A simulated annealing heuristic for the team orienteering problem with time windows. Eur J Oper Res 217(1):94–107

Montemanni R, Gambardella LM (2009) Ant colony system for team orienteering problem with time windows. Found Comput Decis Sci 34(4):287–306

Righini G, Salani M (2008) New dynamic programing algorithms for the resource constrained elementary shortest path problem. Networks 51(3):155–170

Righini G, Salani M (2009) Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Comput Oper Res 36(4):1191–1203

Souffriau W, Vansteenwegen P, Vanden Berghe G, Van Oudheusden D (2013) The multiconstraint team orienteering problem with multiple time windows. Transp Sci 47(1):53–63

Vansteenwegen P, Souffriau W, Vanden Berghe G, Van Oudheusden D (2009) Iterated local search for the team orienteering problem with time windows. Comput Oper Res 36(12):3281–3290

Vansteenwegen P, Souffriau W, Van Oudheusden D (2011a) The orienteering problem: a survey. Eur J Oper Res 209(1):1–10

Chapter 7
Applications of the OP

In past years, we observe from literature that the VRP and OP including their variants have been used to model many different planning problems from practice, such as the routing of technicians, athlete recruitment, or military applications. Recently, other practical applications, such as the tourist trip design problem (TTDP), the mobile crowdsourcing problem, the smuggler search problem, the wildfire routing problem, and the integration of vehicle routing, inventory management, and customer selection problems, have been studied and use the OP as a basic model. In this chapter, various practical applications will be discussed in more detail. We will describe how different aspects from practice can actually be modeled as OP variants. More details about the formulations of the most used OP variants and algorithms to tackle these variants, will be explained in Chap. 8.

7.1 Logistics

Several applications in logistics with various characteristics, such as selecting a set of customers to serve and different profits associated with different customers, can be modeled as a VRP with profits. Consider, for example, the reverse logistics problem of a firm that needs to collect used products from its dealers and customers to reuse or recycle (parts of) these products. Daily, decisions like which customers and dealers will be visited and in which order, need to be made. In supply chain management, this reverse supply chain recently became a very attractive topic, given the rising awareness on environmental sustainability and climate change. This motivates researchers and practitioners to develop new models and algorithms in order to further reduce resource consumption. In this section, we focus on practical problems in the logistic domain that can be modeled by the OP and its variants. The below-mentioned logistics models are summarized in Table 7.1 including some references.

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_7

In the context of the VRP with profits, it is not compulsory to visit all customers, which is actually the case for most classical vehicle routing problems. Given a fleet of homogeneous vehicles, the main objective is to determine a route for each vehicle that maximizes the total profit while complying to a time budget constraint that each vehicle has. Each customer can only be visited once by one vehicle. All vehicles start and end at the same node, which is the depot. If in this case **the limited capacity of each vehicle** is considered explicitly, capacity constraints per route need to be considered. This problem is known as the **capacitated team orienteering problem (CTOP)**. Capacity constraints are discussed further in Sect. 8.1.

If the vehicle capacity is considered, it can be beneficial to **only partially serve a customer**. Obviously, this assumes that the partial profit is proportional to the demand served. This is not always possible in practice, but in some cases customers will agree with this principle. Consider the following example to illustrate why partially serving a customer can be beneficial. Instead of serving a first customer completely and collecting its entire profit, the total collected profit for our vehicle could be higher if we use our limited capacity to serve this first customer only half (and collect half of its profit), if that allows us to serve an additional customer (with higher profit per volume) as well. In this problem, each customer is served only once, either completely or partially. This problem is called the **CTOP with incomplete service (CTOP-IS)**.

Another option that is sometimes considered in practice, is that **deliveries to customers can be split over multiple vehicles**. In this case, customers are visited more than once. Again, not all customers will accept this, but some might (maybe a price reduction can be negotiated). So, it is not necessary for customers to be completely served by one vehicle. This problem is known as the **split delivery CTOP (SDCTOP)**. It is proven that allowing split deliveries can sometimes reduce the number of vehicles and the total transportation cost.

In the context of another particular logistic problem, namely the truck-load transportation problem, customers place orders consisting of **requests for a transportation service from an origin to a destination**. It is required that a full truck goes from the corresponding origin to the corresponding destination. These services can be represented as arcs on a graph that have to be traversed in order to satisfy the corresponding customer requests. Another example is that transportation services respond to specific transport questions posted on the web, in the context of an electronic auction. The carrier will make a bid on these questions. In this case, a transportation service first goes to the origin node with an empty truck, fills the truck with a load there, then traverses the arc, and finally unloads the truck completely at the destination. The transportation company may postpone or may not serve some customers, but each customer is associated with a profit. Both above scenarios can be modeled as the **orienteering arc routing problem (OARP)** with the objective of finding a route, starting and ending at the depot, carrying out a selection of transportation questions, while considering the limited time budget. This OARP is extended to the **team orienteering arc routing problem (TOARP)** if the number of vehicles is more than one.

In the traditional supply chain context, supplier and customers independently make their decisions on replenishing their inventory. Both suppliers and customers try to minimize their own costs, which are the transportation cost for suppliers (wanting to minimize the number of deliveries and thus maximize the quantity delivered) and the inventory handling cost for customers (wanting to minimize the inventory and thus increasing the number of deliveries). The Vendor Managed Inventory (VMI) was then introduced later to synchronize the different activities. The supplier takes full responsibility for managing the inventory of customers. The decisions of satisfying customers' demands are decided at the supplier level. The inventory routing problem (IRP), which is a VMI problem, **considers both managing the inventories at the customers and distributing the products** from a central depot to the customers. The supplier manages the routing of vehicles distributing the products and the timely replenishment of inventories at the customers as well. Cyclic IRP (CIRP) is a variant of IRP where customer demand rates are stable with an infinite planning horizon. The main objective is to find a cyclic distribution plan for a set of customers with the objective of minimizing long-term transportation and inventory costs. The vehicles are allowed to perform multiple trips on a single day.

In single-vehicle CIRP (SV-CIRP), which is a special case of CIRP, visiting all customers is not compulsory, but a reward occurs for each visited customer. The objective is to simultaneously minimize transportation and inventory costs and maximize the collected rewards from visited customers. From the OP perspective, the SV-CIRP can be considered as the inventory routing variant of the OP, namely the **inventory OP (IOP)**. Both have the same objective which is to maximize the total profit, collected from visited customers, without violating a time constraint. Each customer can only be visited at most once. The only difference is that the SV-CIRP has no upper bound for the cycle time (which is actually not very realistic). In OP, we can limit the cycle time to a given upper bound value, e.g., a number of hours or a working day. The IOP is discussed in more detail in Sect. 8.5.

In some cases, the customers should not be considered independently, but they belong to a number of chains and a profit is associated with each chain, and not with each individual customer. Then, **the (chain) profit can only be collected when all customers of a chain are visited**. This problem can be modeled as the **clustered OP (ClOP)**. Also the opposite can appear in practice. In this case, **the profit of a chain is collected as soon as (at least) one customer from the chain is visited**. In this case the carrier has contracts with the chains and instead of having to serve all customers belonging to a chain, the carrier may choose to serve only one customer from the chain with the entire quantity demanded by all customers of that chain. The distribution among all customers in the chain will then be organized internally by the chain customers, not by the carrier. This could reduce the cost (and price) for the carrier service. This particular problem is modeled by **the set orienteering problem (SetOP)**.

A last application arises in **humanitarian logistics**. For example, a helicopter looks for the survivors of a disaster, while distributing food and first aid kits whenever survivors are located. The profit is in this case related to the duration of stay at target locations (nodes), e.g., finding more survivors or distributing more food. This problem is known as the **orienteering problem with variable profits (OPVP)**.

Table 7.1 Logistic variants

Practical problem	Related OP	References
Capacitated vehicles	CTOP	Archetti et al. (2009, 2013, 2014a)
Truck-load transportation problem	TOARP	Archetti et al. (2014b, 2015, 2016)
Integrating vehicle routing and inventory management	IOP	Vansteenwegen and Mateo (2014)
Clustered nodes (all)	ClOP	Angelelli et al. (2014)
Clustered nodes (one)	SetOP	Archetti et al. (2018)
Variable profit	OPVP	Erdoğan and Laporte (2013)

7.2 Tourism

Many tourists want to visit as much tourist attractions or points of interest (POIs) as possible during a city trip to cities such as Paris, Barcelona, or London. However, it is impossible to visit all POIs, having a limited amount of time and budget available. Therefore, most tourists gather information from different resources about all POIs. The selection of the POIs to visit is then based on personal preferences. However, the selection of POIs, of course, also depends on several factors, such as the available time, the distance between POIs, and the opening hours of the POIs. The planning process will be further complicated by budget constraints, weather conditions, public transport availability, planning other activities such as lunch breaks and shopping, and so on. When carried out properly, this is a complex and time-consuming process. Therefore, some sort of decision support would be very welcome and this can be developed based on the tourist trip design problem (TTDP). In the TTDP, deciding which POIs to be visited and determining the best sequence for each trip day are the main challenges.

Personalized electronic tourist guides (PETs) have been developed in order to assist the trip planning process and to generate tourist routes based on a personal profile and preferences. Some examples of PETs are the *City Trip Planner* and the *mtrip* travel guide (https://www.mtrip.com/). All these PETs require several input data, such as a set of candidate POIs, the travel time, or distance among POIs by considering different modes of transport, the number of routes to be generated, depending on the period of stay of the user, the visiting duration at a particular POI, the time that a user can spend on visiting POIs each day and the 'profit' of each POI which depends on the user's preferences and interests in specific types of POIs.

The general concept of a PET is illustrated in Fig. 7.1. Each PET should integrate (at least) three functionalities:

- The recommendation functionality (denoted as 'Interest estimation' in Fig. 7.1), which generates a list of POIs for each individual profile, including POI information such as a profit and the typical duration of the visit.
- The route generation functionality (denoted as 'tourist trip design algorithm' in Fig. 7.1), which employs an algorithm to generate personalized routes using the

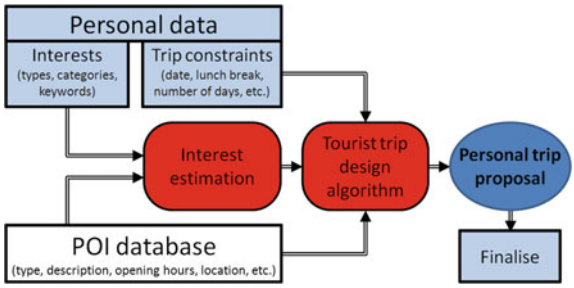

Fig. 7.1 General concept of a PET, taken from Vansteenwegen et al. (2011)

list of POIs generated in the previous functionality. It also considers tourist-related data, such as how many visiting days and the time budget for each day, and POI information such as the location and opening hours.
- The customization functionality (denoted as 'Finalise' in Fig. 7.1), which allows users to add, remove, reorder suggested POIs, and/or ask for a new proposal.

A single tour of the TTDP can be modeled as a traveling salesperson problem with profits (TSPP). In the TSPP, it is not required to visit all cities. The problem is then modeled as a problem with two conflicting aspects to consider: maximizing the collected profit and minimizing the travel cost. Instead of using a real bi-objective model, the problem can be formulated as one of these single objective TSPPs, discussed in Chap. 2:

- The profitable tour problem (PTP) with the objective of maximizing the difference between the collected profit and the travel cost (see Sect. 2.1).
- The prize collecting TSP (PCTSP) with the objective of minimizing the travel cost. The minimum total collected profit is set as a constraint (see Sect. 2.2).
- The orienteering problem (OP) with the objective of maximizing the total collected profit, while the available time is set as a constraint (see Sect. 2.3).

> In our opinion, the best basic model for the TTDP is the OP. It is also used by far the most in literature. The main objective is to maximize the total collected profit from visited POIs within a given time budget. The set of nodes N refers to the set of POIs. Node 1 (and node N) refers to the start node (and end node), e.g., the hotel where the tourist is staying or the station where the tourist arrives. The profit of each POI corresponds to the personal preference of the tourist and should be set in such a way that it makes sense to simply add up the individual profits collected at each POI. The multiple day TTDP can then be formulated as the TOP (see Sect. 3.1) and opening hours can be considered by using time windows in the OPTW and the TOPTW (see Sect. 3.2).

Various extensions of the TTDP can also be formulated as the variants of the OP, TOP, OPTW, or TOPTW. The TOPTW could be extended by introducing multiple time windows per POI and time windows can be different on different days. The

possibility to schedule a lunch break can also be added. This break has no fixed location or exact timing. Other possible extensions such as 'must-see POIs', which can be modeled as mandatory nodes, or POIs which cannot be visited on certain days, will significantly increase the complexity of the problem. Several POIs may also have different profits which depend on certain factors, such as weather conditions. Parks have a higher profit when the sun is shining than when it is raining. This can be modeled by duplicating the node for this POI, each with a different score and opening hours/days. Then an additional constraint should be added that only one of these two POIs can be visited.

In recent years, almost all papers in the TTDP literature use the OP (and other extended variants) to formulate the TTDP. Here, we summarize several TTDP aspects that can be modeled with variants of the OP. The below-mentioned TTDP models are summarized in Table 7.2 including the related references:

- Very similar attractions will, most likely, be given similar attraction values or profits. Some tourists prefer to visit at most one or two of these attractions, while for other tourists it has an added value to see (and compare) them all. Another possibility is that a lower valued attraction becomes more appealing when visited in combination with other POIs. In this case, the **objective function is a nonlinear function of the POIs visited**. A certain combination of POIs could produce a higher (or lower) profit than the sum of individual profits. This type of objective function is applied in the OP variant known as the **generalized OP**.
- Large countries, such as the USA, are divided into several travel regions. POIs are then grouped based on regions. **A profit** is associated with each region and **can only be gained if all POIs in one particular region are visited**. Visiting more than one region is possible subject to the duration of stay and the available budget. This problem can be modeled with the **clustered OP**.
- The OPTW and TOPTW can be used to model opening hours in the TTDP. In both OPTW and TOPTW, tourists are allowed to wait at a certain POI before it opens. However, some tourists might indicate they **never want to wait**. Therefore, an additional constraint can be added that ensures that arrivals at POIs are only scheduled when the POIs are actually open. This problem is called the **modified TOPTW**.
- Several **POIs can be extended with a certain number of attributes**, for example, they may have different time windows in different days and multiple time windows per day. This problem can also be extended by adding other attributes, such as budget limitations for entrance fees and 'max-n type constraints' for each day or for the whole trip, e.g., tourists only visit one beach during their whole trip. This problem corresponds to another extension of the TOPTW, namely the **multi-constraint TOP with multiple time windows**. This variant will be further discussed in Sect. 8.1.
- Providing up-to-date **public transportation** information should also be part of the functionalities of a PET. The travel time required to move from one POI to the next, will vary according to the transportation modes (walking or public transportation) that are considered and according to the time the tourist leaves a certain POI.

This leads to a **time dependent TOPTW (TD-TOPTW)**, also including public transport information. The TDTOPTW is discussed further in Sect. 8.3.

- When tourists visit a region, a certain route may be worthwhile to follow and might be an attraction on its own. This could for instance be a drive through a beautiful canyon. Also in a city, **scenic routes** or strolling through a street can be considered as a POI. In this case the POIs are not nodes in a network, but arcs or edges. This problem was first modeled as the **cycle trip planning problem (CTPP)**. In this case, the network consists of a set of nodes and a set of arcs where each arc corresponds with a cost and a profit. This problem is also known as the **arc OP (AOP)**.
- The CTPP can be extended to **different classes of users**. Each class may have its own preferences, such as the cultural oriented tourist, the shopper, the nature oriented tourist, etc. The objective is to maximize the sum over all user classes of the attractiveness of the itinerary selected for that class. This problem is known as the **multi-commodity OP with network design** (MOP-ND).
- When visiting POIs, the **visit duration may be different for different tourists**. For example, when visiting a museum, some tourists spend much more time on information collection. This increases the duration of their visit, but also their appreciation. In other words, the profit collected may depend on the time spent at the museum. This problem is considered in the **OP with variable profits (OPVP)**.
- In some cases, tourists spend multiple days or weeks **to visit an entire region**. Then, it makes no sense to stay in one hotel and start every day trip from the same hotel. In that case, tourists might spend the first nights in one hotel, then move to the next for a number of nights, then spend one night in again another hotel and so on. Obviously the selection of hotels should be integrated with the selection of POIs from the region. Actually, the hotel selection will have a large influence on which POIs can be reached and how much profit can be collected during the entire trip. This problem is modeled as the **OP with hotel selection (OPHS)**.

Table 7.2 TTDP variants

Problem	Related OP	References
Nonlinear objective function	Generalized OP	Geem et al. (2005)
Clustered POIs	Clustered OP	Herzog and Wörndl (2014)
No waiting	modified TOPTW	Sylejmani and Dika (2011)
Multiple attributes and constraints	MC-TOPMTW	Souffriau et al. (2013)
Public transport	TD-TOPTW	Garcia et al. (2013)
Scenic routes	CTPP or AOP	Verbeeck et al. (2014)
Different user classes	MOP-ND	Malucelli et al. (2015)
Duration dependent profits	OPVP	Yu et al. (2015)
Regional visits	OPHS	Divsalar et al. (2013, 2014)

More details on the OP applications modeling the TTDP are summarized in the latest OP survey by Gunawan et al. (2016). The definition of the TTDP is described in Vansteenwegen and Van Oudheusden (2007). Examples of electronic tourist guides (ETG) can be found in Souffriau et al. (2008), Vansteenwegen et al. (2011), and Verbeeck et al. (2014). Gavalas et al. (2014) provides a detailed survey on the TTDP.

7.3 Other Applications from Practice

The below-mentioned applications are summarized in Table 7.3 including the related references.

The **sales representative planning problem** is the route planning problem of sales representatives (or other field workers) who have the duty of visiting their customers regularly in specific time intervals. In some cases, flexibility in the visiting schedules is required and which customers should be visited during the next week is usually planned beforehand. Some customers could be mandatory customers and other customers might have specific time windows or even a different time window for every day. This problem is called the **multi-period OP with multiple time windows (MuPOPMTW)**. So the time window constraints are also extended in order to cater the multiple time windows. Moreover, additional constraints ensure the visits to mandatory customers.

Another example of the sales representative planning problem is in the pharmaceutical industry when sales representatives need to visit doctors to inform them of their products and encourage them to become active prescribers. The sales representatives need to decide which doctors to visit on a daily basis. This problem can be considered as the TOPTW. However, since patients typically have uncertain service times, an uncertain waiting time for the sales representatives may occur. This increases the complexity of the problem and the problem can be seen as the **stochastic OP with time windows (SOPTW)**. Stochastic OPs are discussed further in Sect. 8.4.

The **resource allocation during wildfires** is considered as another application of the TOPTW. The incident management teams (IMT) need to coordinate the response efforts to wildfires. The protection requirement of an asset is defined as the amount of resources, e.g., vehicles or fire trucks, required to provide an adequate level of protection to defend an asset. Each asset has a certain value or profit that can be collected when the asset is protected. The profit is collected if and only if a sufficient number of vehicles cooperatively protects the location during the time window. This problem is a generalization of the TOPTW, namely the **cooperative OP with time windows (COPTW)**.

The smuggler search problem is a path-constrained optimal search problem in continuous space and time. This problem arises in **military search and rescue applications** where the objective is to route a searcher to find moving targets in an area of interest (AOI). Targets can be thought as carrying some type of illicit material. Limited resources (e.g., time, fuel, etc.) are expended by the searcher while performing search actions in an AOI. Rewards are collected by the searcher at each

node visited, where the reward level depends on the amount of scarce resources expended, e.g., how much time spent dwelling at a particular node. Arcs in the network are traversed while consuming the same limited resources used for reward collection. In this problem, the reward collected is not a constant, it is a function of the dwell time. This dwell time is a decision problem—how long we can stay in a particular node– and this is limited by the remaining time used for traveling. So the sum of the dwell times and the traveling times should be smaller than the available time for searching. This problem is a generalization of the OP, namely the **generalized OP with resource dependent rewards (GOP-RDR)**.

Crowdsourcing is defined as an idea of outsourcing a task, traditionally performed by an employee, to a large group of people in the form of an open call. Some companies outsource tasks to individuals who are willing to complete them for rewards. Each task should be carried out at a different location. In this problem, each individual solves this problem as an **OP** in order to determine their route or set of tasks. If public transport and time windows are taken into consideration, the problem is extended to the **multi-path OP with time windows**.

One application of the crowdsourcing is a mobile crowdsourcing application that computes the best (detour) routes for smartphone users who have spare time to perform additional tasks for a certain reward. For example, a system called NoiseTube enables users to measure their personal exposure to noise in their everyday environment by using GPS-equipped mobile phones as noise sensors. The data can be automatically sent and shared online to the public to contribute to the collective noise mapping of cities. Users might receive suggestions of (less-covered) locations in the city to walk by, in return for a certain reward. Given a large pool of individuals to perform a variety of location-specific urban tasks, they are asked to visit additional nodes that are close to their routine movements. The objective is to maximize the collected profit from visiting additional nodes while respecting time budget constraints. This can be modeled as a **TOP**. If we assume that each user has a finite list of possible routes with a known probabilistic distribution, this problem becomes an extension of the **multi-period OP with multiple time windows (MuPOPMTW)**.

One of the latest applications of the OP is the **agile earth observation satellite scheduling problem**. The earth observation satellite (EOS) is widely used in many important fields, such as remote sensing of natural resources, monitoring of nature disasters, and so on. The agile EOS is a new generation of EOSs with a higher capability to observe more targets over a period of time due to its three degrees of freedom of mobility of the camera. The scheduling problem of an agile EOS is an oversubscribed resource scheduling problem with time windows where not all tasks must be scheduled, but a different profit is collected based on each selected task. The objective is to maximize the total collected profit. In the satellite scheduling case, the travel times are time dependent. This is because the travel ('transition') time to change the look angle of the camera from one observation to the next, depends on the look angle of the first observation and thus on the observation start (and end) time. Moreover, when observing targets with a satellite, the look angle of the camera determines the quality of the image taken and thus also the profit collected. The best image quality is obtained at the 'nadir' point where the satellite observes a target

Table 7.3 Papers on other applications

Problem	Related OP	References
Sales representative planning problem	MuPOPTW	Tricoire et al. (2010)
	SOPTW	Zhang et al. (2014)
Wildfire routing problem	COPTW	Vander Merwe et al. (2014)
Smuggler search problem	GOP-RDR	Pietz and Royset (2013)
Crowdsourcing problem	TOP	Ludwig et al. (2009)
	TOP	Chen et al. (2014)
	MuPOPMTW	Chen et al. (2015)
Agile earth observation satellite scheduling	TDOP-TDPTW	Peng et al. (2018, 2019)

directly below and lower profits are collected before and after this point. Therefore, this problem can be modeled as the **OP with time-dependent travel times, time-dependent profits and time windows (TDOP-TDPTW)**.

References

Angelelli E, Archetti C, Vindigni M (2014) The clustered orienteering problem. Eur J Oper Res 238(2):404–414

Archetti C, Feillet D, Hertz A, Speranza MG (2009) The capacitated team orienteering and profitable tour problems. J Oper Res Soc 60:831–842

Archetti C, Bianchessi N, Speranza MG (2013) The capacitated team orienteering problem with incomplete service. Optim Lett 7(7):1405–1417

Archetti C, Bianchessi N, Speranza MG (2014a) The split delivery capacitated team orienteering problem. Networks 63(1):16–33

Archetti C, Speranza MG, Corberán Á, Sanchis JM, Plana I (2014b) The team orienteering arc routing problem. Transp Sci 48(3):442–457

Archetti C, Corberán Á, Plana I, Sanchis JM, Speranza MG (2015) A matheuristic for the team orienteering arc routing problem. Eur J Oper Res 245(2):392–401

Archetti C, Corberán Á, Plana I, Sanchis JM, Speranza MG (2016) A branch-and-cut algorithm for the orienteering arc routing problem. Comput Oper Res 66:95–104

Archetti C, Carrabs F, Cerulli R (2018) The set orienteering problem. Eur J Oper Res 267:264–272

Chen C, Cheng SF, Gunawan A, Misra A, Dasgupta K, Chander D (2014) TRACCS: Trajectory-aware coordinated urban crowd-sourcing. In: Proceedings of the second AAAI conference on human computation and crowdsourcing (HCOMP 2014), Pittsburgh, USA

Chen C, Cheng SF, Lau HC, Misra A (2015) Towards city-scale mobile crowdsourcing: task recommendations under trajectory uncertainties. In: Proceedings of the international joint conference on artificial intelligence (IJCAI-2015), Buenos Aires, Argentina

Divsalar A, Vansteenwegen P, Cattrysse D (2013) The orienteering problem with hotel selection: a variable neighborhood search method. Int J Prod Econ 145(1):150–160

Divsalar A, Vansteenwegen P, Sorensen K, Cattrysse D (2014) A memetic algorithm for the orienteering problem with hotel selection. Eur J Oper Res 237(1):29–49

Erdoğan G, Laporte G (2013) The orienteering problem with variable profits. Networks 61(2):104–116

Garcia A, Vansteenwegen P, Arbelaitz O, Souffriau W, Linaza MT (2013) Integrating public transportation in personalised electronic tourist guides. Comput Oper Res 40(3):758–774
Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G (2014a) A survey on algorithmic approaches for solving tourist trip design problems. J Heuristics 20(3):291–328
Geem ZW, Tseng CL, Park Y (2005) Harmony search for generalized orienteering problem: best touring in china. In: Wang L, Chen K, Ong YS (eds) Advances in natural computation, vol 3612. Lecture notes in computer science. Springer, Berlin, pp 741–750
Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332
Herzog D, Wörndl W (2014) A travel recommender system for combining multiple travel regions to a composite trip. In: Proceedings of new trends in content-based recommender systems (CBRecSys) workshop, 8th ACM conference on recommender systems, California, USA
Ludwig B, Zenker B, Schrader J (2009) Recommendation of personalized routes with public transport connections. In: Tavangarian D, Kirste T, Timmermann D, Lucke U, Versick D (eds) Intelligent interactive assistance and mobile multimedia computing: international conference, IMC 2009, communications in computer and information science, vol 53. Springer, Berlin, pp 97–107
Malucelli F, Giovannini A, Nonato M (2015) Designing single origin-destination itineraries for several classes of cycle-tourists. Transp Res Procedia 10:413–422
Peng G, Dewil R, Verbeeck C, Gunawan A, Xing L, Vansteenwegen P (2019) Agile earth observation satellite scheduling: an orienteering problem with time-dependent profits and travel times. Comput Oper Res 111:84–98
Peng G, Vansteenwegen P, Liu X, Xing L, Kong X, (2018) An iterated local search algorithm for agile earth observation satellite scheduling. In: Proceedings of 2018 spaceOps conference, Marseille, France
Pietz J, Royset JO (2013) Generalized orienteering problem with resource depedent rewards. Nav Res Logist 60(4):294–312
Souffriau W, Vansteenwegen P, Vertommen J, Vanden Berghe G, Van Oudheusden D (2008) A personalised tourist trip design algorithm for mobile tourist guides. Appl Artif Intell 22(10):964–985
Souffriau W, Vansteenwegen P, Vanden Berghe G, Van Oudheusden D (2013) The multiconstraint team orienteering problem with multiple time windows. Transp Sci 47(1):53–63
Sylejmani K, Dika A (2011) Solving touristic trip planning problem by using taboo search approach. Int J Comput Sci 8(5):139–149
Tricoire F, Romauch M, Doerner KF, Hartl RF (2010) Heuristics for the multi-period orienteering problem with multiple time windows. Comput Oper Res 37(2):351–367
Vander Merwe M, Minas JP, Ozlen M, Hearne JW (2014) The cooperative orienteering problem with time windows. Optim Online
Vansteenwegen P, Mateo M (2014) An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. Eur J Oper Res 237(3):802–813
Vansteenwegen P, Van Oudheusden D (2007) The mobile tourist guide: an OR opportunity. OR Insights 20(3):21–27
Vansteenwegen P, Souffriau W, Vanden Berghe G, Van Oudheusden D (2011) The city trip planner: an expert system for tourists. Expert Syst Appl 38(6):6540–6546
Verbeeck C, Vansteenwegen P, Aghezzaf EH (2014b) An extension of the arc orienteering problem and its application to cycle trip planning. Transp Res Part E 68:64–78
Yu J, Aslam J, Karaman S, Rus D (2015) Anytime planning of optimal schedules for a mobile sensing robot. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS 2015), Hamburg, Germany
Zhang S, Ohlmann JW, Thomas BW (2014) A priori orienteering with time windows and stochastic wait times at customers. Eur J Oper Res 239(1):70–79

Chapter 8
Other Orienteering Problem Variants

In this chapter, different variants of routing problems with profits will be discussed. Based on what is available in literature, mostly variants of the orienteering problem will be discussed. A first variant considers capacity constraints, since these appear frequently in many practical applications. Next, multi-objective orienteering problems, explicitly considering different types of profits separately are discussed. Time-dependent and stochastic travel times are also relevant for most practical applications. These are considered together with time-dependent and stochastic profits. More and more routing problems are considered together with inventory management. For routing problems with profits, this leads to the inventory orienteering problem, which is discussed below.

8.1 Capacity Constraints

In many practical applications, both for regular routing problems and routing problems with profits, some sort of capacity constraints need to be considered. In case of logistic applications, this is typically the vehicle capacity. A straightforward TOP variant with capacity constraints is called the **capacitated team orienteering problem** (CTOP). In the CTOP, each node has a demand next to a profit and the objective is to determine a route for each available vehicle in order to maximize the total profit, without violating the vehicle capacity (and time budget).

The CTOP can be extended by relaxing the assumption that a customer must be completely served. This is called the CTOP with Incomplete Service (CTOP- IS). It can be proven that the profit collected by the CTOP-IS can become as large as twice the one collected by the CTOP. Another extension of the CTOP allows a node to be served by more than one route. This problem is named the **split delivery CTOP (SDCTOP)**. By splitting deliveries, the cost of the routes and the number of vehicles

P. Vansteenwegen and A. Gunawan, *Orienteering Problems*,
EURO Advanced Tutorials on Operational Research,
https://doi.org/10.1007/978-3-030-29746-6_8

can be reduced. The SDCTOP is further extended to the **SDCTOP with minimum delivery amounts (SDCTOP-MDA)**.

For tourist applications, capacity constraints can be used to model the budget tourists want to spend on entrance fees. Moreover, capacity constraints can also be used to model that a tourist only wants to visit a given number of points of interest of a certain type. For instance, a tourist only wants to visit one museum per day or at most two parks during a multiple day trip or at least one church during the trip.

For the mathematical formulation of the problem, adding capacity constraints is relatively easy. If we start from the formulation of the TOP, Eqs. (3.1)–(3.7), capacity constraints can be added as:

$$\sum_{i=1}^{|N|} D_i y_{im} \leq CAP; \ \forall m = 1, \ldots, M. \tag{8.1}$$

In this equation, D_i represents the demand (or volume or entrance fee) of node i and CAP is the capacity of the vehicle. Remember that $y_{im} = 1$, if node i is visited in route m, and 0 otherwise. So the total demand that can be collected in each route is limited to the capacity of the vehicle. By replacing CAP with a specific CAP_m for each route m, a different capacity per day can be imposed. If the goal is to limit the capacity (budget) for all routes (days) together, the constraint changes to:

$$\sum_{m=1}^{M} \sum_{i=1}^{|N|} D_i y_{im} \leq CAP. \tag{8.2}$$

When a tourist only wants to visit a given number of points of interest of a certain type, this is called a 'max-n type' constraint. Therefore, the above mentioned capacity constraints can be generalized in a sort of knapsack constraints, where e_{imz} is the cost associated with knapsack constraint z for node i in route m and CAP_z is the budget available for knapsack constraint z:

$$\sum_{m=1}^{M} \sum_{i=1}^{|N|} e_{imz} y_{im} \leq CAP_z; \ \forall z = 1, \ldots, Z. \tag{8.3}$$

In case of budget limitations, e_{imz} represents the entrance fee for node i in route m and CAP_z the money budget available to spend.

The 'max-n type' constraints can be modeled with these knapsack constraints: e_{imz} is set to 1 if node i is of type z, and 0 otherwise; CAP_z obviously is the maximum number of visits of type z. These 'max-n type' and budget constraints can also be stated per route, e.g., visit at most one museum on the first day or spend at most 50 euro on the third day. In this case, an extra knapsack constraint should be added for that particular day, i.e., a particular value of m.

The same type of constraint can also be used when modeling nodes with multiple time windows, for instance, a point of interest that is closed during, but open before and after lunch, or with a different profit during the day or during the evening or night. In both cases a duplicate of the node can be added to the problem instance, with its own opening hours and/or profit. Obviously, a constraint should then be added that at most one node can be selected among the original node and its duplicate(s).

This way of modeling leads to a high number of capacity or knapsack constraints, which is a challenge for many solution approaches. Especially when local search techniques are applied to generate a feasible solution or improve a given solution, the challenge is to check all these constraints in a very short computation time. For example, when a local search adds an extra node to a given solution, the list of other possible insertions should be updated according to the large set of constraints. Possible insertions that would violate constraints should be marked as impossible for future selection. When nodes are removed, these automatically become available for future (re-)insertion. Furthermore, this removal leads to extra slack in some constraints, again allowing other insertions which were previously marked impossible.

This can be addressed efficiently by a technique called **neighborhood consistency checking**. A naive approach would check feasibility for every node, every time a node is added to or removed from a given solution. Such an approach would significantly increase the execution time of the local search. Hence, a faster procedure is preferred. This is done by introducing an a_{im} variable for each node-route combination. When a node i is included in route m of the given solution, a_{im} is set to 'IN'; when a node i cannot be added to route m of the given solution due to one or more of the constraints, a_{im} is set to 'NO'; when a node i should be considered for insertion in route m, a_{im} is set to 'CON'.

When a local search heuristic performs an insertion of node i in route m, the value a_{im} becomes 'IN' and has to be propagated to other constraints to determine which values a_{jn} change to 'NO' and filter these from the set of possible insertions. During this propagation, only the constraints that involve a visit to node i in route m are considered. The reverse problem occurs when a node is removed from the solution: the corresponding a_{im} variable becomes 'CON', meaning that the node i can be considered for re-insertion later. Due to the removal of this node, more slack becomes available in all constraints that involve a visit to node i in route m. For each node j in these constraints, it should be verified if a_{jn} can be changed from 'NO' to 'CON', based on all other constraints in which node j is involved in route n.

The CTOP, some benchmark instances and some solution approaches are discussed in, e.g., Archetti et al. (2009, 2013b), and Tarantilis et al. (2013). The CTOP-IS, SDCTOP, and SDCTOP-MDA are discussed in Archetti et al. (2013a, 2014a), and Wang et al. (2014), respectively. More details about the multiconstraint team orienteering problem (with multiple time windows) and neighborhood consistency can be found in Souffriau et al. (2013).

8.2 Multi-objective Routing Problems with Profits

In Sect. 7.2, the planning of individual tourist trips, the TTDP, is modeled by variants of the single objective orienteering problem (OP). Although, in practice, a tourist typically has different preferences for different types of interest (nature, churches, typical squares and streets, shopping, museums, etc.). The single objective OP thus involves an aggregation of the different preferences or objectives into a single objective. A major drawback of this approach is that each preference requires a predefined weight in the resulting single objective. Determining these weights to represent tourist preferences is obviously very challenging and, moreover, it significantly influences the final tourist trip proposed to the tourist.

An alternative way of modeling the TTDP is the multiple-objective orienteering problem. This would allow to model the different preferences separately instead of bringing those together in one objective. In that case, the solution algorithm needs to determine a set of non-dominated or Pareto-optimal solutions, instead of a single (near) optimal solution. A solution is dominated by another solution when the other solution scores better on at least one objective and not worse on any of the other objectives. By filtering out the dominated solutions, the number of solutions presented to the tourist can be reduced significantly. At the end, it is then up to the tourist to select one of those Pareto-optimal solutions. This is probably still a difficult task for a tourist and thus a drawback of this approach. Probably some help in exploring the set of solutions and selecting a final option would be required.

Another bi-objective extension, in this case of the orienteering problem with time windows (OPTW), is considered when modeling mobile freelancers who have to integrate irregular tasks into their daily schedules. Typically, freelancers have a number of tasks at various locations and times, with different levels of importance. In the bi-objective OPTW, the selection and scheduling of these tasks is considered and one objective is to schedule more (rewarding) tasks (gain more income) and the other objective is to enjoy more free time (quality of life).

> Actually, also the PTP itself could be considered as a bi-objective problem, since the profit is maximized and the travel cost is minimized. However, in the PTP both objectives are just added together in one objective. If both objectives are considered separately, this problem is called the traveling salesperson problem with profits (TSPP).

Algorithms dealing with multi-objective problems typically generate and improve an entire population of solutions, maintaining the non-dominated solutions. For the bi-objective OP, a so-called Pareto ant colony optimization algorithm and a Pareto variable neighborhood search method are used to generate a population of solutions. Both methods also include path relinking procedures.

More details on the multi-objective OP, the two solution approaches to address the bi-objective orienteering problem, and some benchmark instances can be found in Schilde et al. (2009). In Matl et al. (2017), a large neighborhood search generating

a set of non-dominated solutions for the bi-objective OPTW is discussed. Another way of tackling multi-objective problems is to solve a series of single objective subproblems, where all but one objectives are transformed into constraints. Such an approach is used to address the TSPP. More details about the TSPP and this solution approach is available in Bérubé et al. (2009b).

8.3 Time Dependency

In the regular OP, the travel time between two nodes is assumed to be a constant value. However, in many practical situations, the travel time actually depends on the network properties, which may affect the travel time between two nodes. The most common example is congestion. On different times of the day, traveling between two customers will require a different travel time. Another example is the travel time of public transport, which typically includes some waiting time before the bus or train arrives. This waiting time depends on the arrival time at the stop/station and thus on the departure time from the previous location. The OP where the travel time between two nodes depends on the departure time at the first node is called the **time pependent OP (TDOP)**. Obviously, this problem is NP-hard.

8.3.1 Mathematical Formulation with Time Dependent Travel Times

Now we present a mixed integer programming model for the TDOP. The model extends the OP model described in Sect. 2.3. Consider a set of nodes $N = \{1, \ldots, |N|\}$ where each node $i \in N$ is associated with the non-negative profit P_i. The start and end nodes are fixed to nodes 1 and $|N|$, respectively. Since the travel times are time dependent, the mathematical formulation should keep track of the time. Moreover, the travel times will require a time index. In this formulation, this is modeled by considering a discrete number of time slots for traveling between each pair of nodes. Each time slot is indicated by the index t in the formulation. The travel times are modeled based on an intercept and a slope, different for each considered time slot t. The TDOP can be formulated as an integer programming model with the following decision variables and parameters:

- $x_{ijt} = 1$: if a vehicle travels from node i to j with a departure time in time slot t, 0 otherwise;
- w_{ijt}: the departure time in time slot t when traveling from node i to j;
- θ_{ijt}: slope coefficient of the linear time-dependent travel time for arc (i, j);
- η_{ijt}: intercept coefficient of the linear time-dependent travel time for arc (i, j);
- τ_{ijt}: lower limit of time slot t for arc (i, j);
- t_{ij}: number of time slots for arc (i, j).

$$\text{Maximize} \sum_{i=2}^{|N|-1} \sum_{j=2}^{|N|} \sum_{t=1}^{t_{ij}} P_i x_{ijt} \tag{8.4}$$

The objective function (8.4) is to maximize the total collected score.

$$\sum_{j=2}^{|N|} x_{1j1} = \sum_{i=1}^{|N|-1} \sum_{t=1}^{t_{i|N|}} x_{i|N|t} = 1 \tag{8.5}$$

Constraints (8.5) ensure that the route starts and ends at nodes 1 and $|N|$, respectively.

$$\sum_{i=1}^{|N|-1} \sum_{t=1}^{t_{ih}} x_{iht} = \sum_{j=2}^{|N|} \sum_{t=1}^{t_{hj}} x_{hjt} \leq 1; \ \forall h = 2, \ldots, |N| - 1 \tag{8.6}$$

Constraints (8.6) ensure each node is visited at most once.

$$\sum_{i=1}^{|N|-1} \sum_{t=1}^{t_{ih}} [w_{iht} + (\theta_{iht} w_{iht} + \eta_{iht} x_{iht})] = \sum_{j=2}^{|N|} \sum_{t=1}^{t_{hj}} w_{hjt}; \ \forall h = 2, \ldots, |N| - 1 \tag{8.7}$$

Constraints (8.7) guarantee that the departure time of a succeeding node is equal to the sum of the departure time of the previous node together with the travel time between these two nodes. These constraints do not allow waiting time.

$$x_{ijt} \tau_{ijt} \leq w_{ijt} \leq x_{ijt} \tau_{ij(t+1)}; \ i = 1, \ldots, |N| - 1, j = 2, \ldots, |N|, \forall t \tag{8.8}$$

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} \sum_{t=1}^{t_{ij}} \left[\theta_{ijt} w_{ijt} + \eta_{ijt} x_{ijt}\right] \leq T_{max} \tag{8.9}$$

Constraints (8.8) categorize the departure time in the right time slot which is done by multiplying the departure time with its corresponding θ and η in constraint (8.9). Constraint (8.9) also enforces the limited travel time.

$$w_{1i1} = 0; \ \forall i = 1, \ldots, |N| \tag{8.10}$$

Constraints (8.10) ensure that a route starts in the first time slot.

$$0 \leq w_{ijt} \leq T_{max}; \ \forall i, j = 1, \ldots, |N|, \forall t \tag{8.11}$$

Constraints (8.11) guarantee that all departure times are less than or equal to T_{max}.

8.3.2 Solution Approaches for Time Dependent Travel Times

Many approaches have been developed to address time dependent travel times for the OP. Table 8.1 summarizes these algorithms and also mentions the application for which these algorithms were developed.

The first algorithm studies the TDOP in the context of a transportation network with given source and destination nodes. The proposed dynamic programming approach is based on the idea of network planning and dynamic node labeling. The ACS includes a time-dependent local search procedure and an efficient local evaluation metric. A strong characteristic of the algorithm is that the insertion step contains a fast evaluation of the possible insertion of a node. The diversification is implemented in the typical way by depreciating the pheromone trails during the construction procedure. The theme park navigation problem is implemented for a large leisure facility. To obtain solutions within acceptable computational time, four different metaheuristics are developed and compared: a restart greedy append construction heuristic (Greedy), a restart variable neighborhood descent (VND), a basic version of ILS (basic ILS), and a modified ILS with adaptive perturbation size and probabilistically intensified restart (adaptive ILS).

The next four approaches combine time dependent travel times with time windows for each node of the OP (TDOPTW). The first application considered in two consecutive papers is the personalized electronic tourist guides (PET) for the city of San Sebastian in Spain. The PET tries to maximize the tourists' satisfaction in near real-time by considering time dependent travel times due to the use of public transportation. Moreover, opening hours (time windows) are considered as well. The leave time of the first node will determine the transportation mode (and the waiting time in case public transport is used) and thus the time dependent travel time. In the first paper a hybrid algorithm combining two heuristics is presented. The first

Table 8.1 Solution approaches for the TDOP and applications

Reference	Problem	Algorithm	Application
Li (2012)	TDOP	Dynamic programming	Transportation network
Verbeeck et al. (2014a)	TDOP	Ant colony system	–
Gunawan et al. (2014)	TDOP	Iterated local search	Theme park navigation problem
Garcia et al. (2010)	TDOPTW	Hybrid iterated local search	Personalized electronic tourist guides
Garcia et al. (2013)	TDTOPTW	Hybrid iterated local search	Personalized electronic tourist guides
Abbaspour and Samadzadegan (2011)	TDOPTW	Adapted genetic algorithm	Tourist trip design problem
Gavalas et al. (2014b)	TDTOPTW	Time dependent CSCRoutes and the SlackCSCRoutes	Tourist trip design problem

heuristic focuses on the calculation of the average travel time between all pairs of nodes. Based on these averages a solution is calculated, actually avoiding the time dependency of the travel times. Afterwards, a repair procedure is required to incorporate the real travel times between two visits. The second heuristic in the hybrid algorithm is the iterated local search (ILS). In the second paper addressing the PET, this ILS approach is further adapted to deal with multiple routes (or multiple day visits) instead of only one (TDTOPTW).

The last two papers address the TTDP by modeling it as a TDOPTW. The first paper considers three different modes of transportation: walking, bus, and subway. Obviously, this implies time dependent travel times. In order to solve the problem, a Genetic Algorithm is proposed. The second paper considers multiple routes (TDTOPTW) when addressing the TTDP. Two cluster-based algorithms, the Time Dependent CSCRoutes (TDCSCRoutes) and the SlackCSCRoutes, are proposed. The algorithms employ an insertion step that considers time dependent travel times caused by waiting times for public transport. TDCSCRoutes selects a node to be inserted based on the insertion cost, while SlackCSCRoutes involves a more global criterion as it considers the effect of the insertion on the whole route. The algorithms are tested on data for the city of Athens (Greece).

8.3.3 Benchmark Instances for Time Dependent Travel Times

Benchmark instances for the TDOP are based on the TOP benchmark instances. Inputs required for this data are the types of arc (highway, morning congestion, etc.), the speed matrix, and the adapted TOP instances. The speed matrix contains a speed for each time slot and based on the type of arc. This first set of instances is then further modified and randomly generated instances are added together with two real-world instances. For this modified set, the original time dependent travel time is discretized in different intervals. The randomly generated set is generated by varying values of $|N|$ and T_{max}. The two real-world instances are based on two popular theme parks in Asia. Each node represents an attraction or ride, the score of each attraction is derived from user preference data. The time dependent travel times depend on the queues (waiting times) before the attraction, which are assumed to be known.

Table 8.2 presents an overview of the available benchmark TDOP instances, which are available in http://www.mech.kuleuven.be/en/cib/op#section-23. While many different works have dealt with benchmark OP, TOP, OPTW, and TOPTW instances, it is worth mentioning that benchmark TDOP instances are not widely used. Until now, each paper focuses on its own instances.

Table 8.2 Benchmark instances for the TDOP

Reference		Number of instances	Number of nodes $\|N\|$
Verbeeck et al. (2014a)		9	32
		9	21
		9	33
		10	100
		3	66
		9	64
		10	102
Gunawan et al. (2014)	MODIFIED	7×4	21–102
	RANDOM	4×4	10–40
	REAL	2×1	17–40

8.3.4 Time Dependent Profits

Another type of time dependency when considering the OP, are time dependent profits. The most used time dependent profits consider a profit decreasing over time. This problem is known as the **traveling repairman problem with profits** (TRPP), where a repairman visits a subset of nodes in order to collect profits, which decrease over time. The TRPP occurs as a routing problem after disasters such as earthquakes or tsunamis. For example, consider the following situation. Immediately after such a disaster, there are a number of villages with an urgent need for medicine. The sooner the medicine reaches a village, the more people can be rescued, so the higher the profit. Time dependent profits are also considered in the **OP with time dependent rewards** (OP-TDR). Also in this problem, the profit of each node monotonously decreases over time, meaning that each visit is naturally scheduled as early as possible for maximizing the collected profit.

Recently, time dependent profits are also considered more generally in case of satellite scheduling. When observing targets with a satellite, the look angle of the camera determines the quality of the image taken and thus also the profit collected. Since the satellite is constantly moving, the observation start time will determine the amount of profit collected. The best image quality is obtained at the 'nadir' point where the satellite observes a target directly below and its camera angle is equal to zero. Lower profits are collected before and after this point with a minimum of only half of the target profit at the edge of the visible time window. In this case each 'visit' should not be scheduled as early as possible in order to maximize the collected profit, but as close as possible to the point in time corresponding to the maximal profit of that visit.

Actually, in this satellite scheduling case, next to the profits, also the travel times are time dependent. This is because the travel ('transition') time to change the look angle of the camera from one observation to the next, depends on the look angle of the first observation and thus on the observation start (and end) time. An algorithm

available to address both time dependencies is the bidirectional dynamic programming based iterated local search (BDP-ILS) algorithm. It contains the typical ILS framework, with an insert procedure that avoids a full feasibility check. To fully exploit the time dependent profits, a bidirectional dynamic programming approach is integrated. This approach efficiently evaluates a solution or an insert move based on a cumulative forward and backward profit evaluation.

8.3.5 Related Literature

The papers mentioned in the tables above (Tables 8.1 and 8.2) present more details on the TDOP, solution approaches, and benchmark instances. The paper of Fomin and Lingas (2002) provides a formal definition of the TDOP and states it is NP-hard. For more details about calculating parameters $theta_{ijt}$, η_{ijt}, and t_{ij} we refer to the work of Verbeeck et al. (2014a).

A tool to solve any TDOP instance can be downloaded at http://www.mech.kuleuven.be/en/cib/op#section-23.

More details about the OP with decreasing profits, or the traveling repairman problem, can be found in Dewilde et al. (2013).

A detailed discussion on the OP with both time dependent travel times and time dependent profits, applied to satellite scheduling, can be found in Peng et al. (2019). Some benchmark instances are available there as well.

8.4 Stochasticity

In situations where congestion may occur, the travel times between nodes are not only time-dependent, but also difficult or even impossible to predict in a deterministic way. In that case, the travel time is not a fixed value but the value is subject to variations and the travel time is said to be stochastic. As a result of the ever-increasing traffic in most areas, the OP variants with stochastic travel times have recently received more attention. The orienteering problem with stochastic travel times is called the **OP with stochastic weights (OPSW)**. Also the service times, required to loading or unloading the vehicle at the customer, are typically stochastic in practice. When both service and travel times are stochastic, the problem is called the **OP with stochastic travel and service times (OPSTS)**.

8.4.1 Problems with Stochastic Travel Times

Obviously, the formulation of the OPSTS starts from the formulation of the OP, but considers the stochastic travel and service times. These stochastic times imply that

the actual arrival at a customer (called the 'realization') will be different from the expected/planned/average arrival time. Therefore, it is possible that the realization takes place later than T_{max} (or outside the opening hours of the customer). In that case, the profit cannot be collected, but some kind of penalty is imposed. This penalty can represent an actual fee determined in a contract or a loss of goodwill. Anyway, in case a planned visit to a customer needs to be canceled, there will be a penalty.

Consider a set of nodes $N = \{1, \ldots, |N|\}$. The start and end nodes are fixed to nodes 1 and $|N|$. With each node $i \in N$ is associated a non-negative profit P_i and now also a penalty e_i. The penalty e_i is incurred if i is selected to be served but the actual visit to customer i would occur after the deadline T_{max}. Let T_{ij} be a non-negative random variable representing the time required to travel from i to j. We assume that the distribution on T_{ij} is known for all i and j. Let S_i be a non-negative random variable representing the service time at customer i. We assume that the distribution of S_i is known for all i. Let the random variable A_i be the arrival time at customer i. For a realization of A_i, $\tilde{A}_i$, we let $R(\tilde{A}_i)$ be a function representing the 'profit' earned at customer i when arriving to i at time $\tilde{A}_i$. We assume that $R(\tilde{A}_i) = P_i$ for $\tilde{A}_i \leq T_{max}$ and e_i otherwise.

The expected reward of a route *omega* is equal to the following equation in which P stands for the probability:

$$\sum_{i \in \omega} [P(A_i \leq T_{max}) P_i - (1 - P(A_i \leq T_{max})) e_i]. \tag{8.12}$$

The objective is to determine the route ω^* such that the expected reward of ω^* is larger than or equal to the expected reward of ω for every ω.

Some variants of the OPSTS have also been studied. Obviously, the OPSW can be formulated as the OPSTS, but without considering the stochastic service times. The OPSW is also formulated by a two-stage recourse model. The first stage is about finding a route. In the second stage, the realizations are revealed and recourse costs are imposed.

In the so called **dynamic stochastic OP (DSOP)**, the dependencies between travel times and the risk preference of the user are considered. The risk profiles of the user are measured as the probability of completing the route within the time budget. Travel times are modeled with random variables that follow a given time-varying distribution.

Another obvious variant considers time windows (opening hours) for each customer. This problem is called the **stochastic OP with time windows (SOPTW)**. It should be noted that the combination of uncertain travel (and service) times and strict time windows has a significant impact on the variability of the solution. This is theoretically and empirically shown in different studies. In this problem, the waiting time is modeled as a random variable dependent on the arrival time and the queue length upon arrival. The objective is to construct an a priori route that maximizes the expected profit collected on a given day.

8.4.2 Solution Approaches for Stochastic Travel Times

Different solution approaches have been developed for the OPSTS and are summarized in Table 8.3. First, a variant of VNS is proposed for the OPSTS. The VNS is able to obtain good solutions, compared to dynamic programming, within a few seconds, while the computation times for dynamic programming grow exponentially with increasing deadlines. In this study, benchmark instances are generated as well. The instances are based on the benchmark OP instances by including the distributions of the travel times and some penalty values. The second study focuses on approximating the objective function more efficiently and to minimize loss in accuracy by implementing Monte Carlo sampling and hybrid methods combining Monte Carlo sampling and an analytical solution. The hybrid methods can offer a reasonable approximation of the objective function in only a fraction of the time.

The third and fourth studies address the DSOP, thus considering dependencies between travel times and the risk preference of the user. In order to tackle this problem, a local search algorithm is developed that combines variable neighborhood search and simulated annealing. Experiments are conducted also on a real-world instance from a theme park navigation problem. The hybrid approach is able to improve the initial solution generated by a greedy insertion algorithm up to 30%. This work is extended in the fourth study. An optimization based approach that employs ideas from the sample average approximation technique, namely mixed integer linear programming-sample average approximation (MILP-SAA), is proposed. This approach is compared with the local search approach in the previous study on the same benchmark instances. The MILP-SAA outperforms the local search approach both on the synthetic instances and on the real-world instance.

The fifth study focuses on the OPSW. So, the travel costs, travel time (including service time) or fuel consumption on the arcs is considered stochastic due to factors such as weather circumstances and congestion. Sample average approximation (SAA) using Monte Carlo simulation is used to solve the problem. Candidate solutions are first generated by a random sample. Obviously, the SAA can only solve small instances within reasonable computation time. Therefore, an OPSW heuristic is proposed based on a randomization concept and a score measure. Computational results show the benefit of using the proposed approaches. Again it is shown that using the OPSW heuristic results in a higher expected profit compared to applying a heuristic for the deterministic OP in an uncertain environment.

The last two studies consider the SOPTW (or OPSWTW). For the SOPTW, an a priori route is determined that should maximize the expected profit collected on a given day. During the execution of this a priori route, two recourse actions are considered. The first recourse determines whether a customer should be skipped based on the arrival time at that particular node. The second one determines how long we should wait when arriving at a certain queue. A VNS heuristic is available to solve benchmark SOPTW instances. The algorithm is a variant of the first VNS algorithm, used for the OPSTS by including the time window constraints. The lower and upper bounds for the SOPTW are obtained by solving the deterministic OPTW

Table 8.3 Solution approaches for the OPSTS

Reference	Variant	Algorithm
Campbell et al. (2011)	OPSTS	VNS
Papapanagiotou et al. (2014)	OPSTS	Hybrid methods and Monte Carlo
Lau et al. (2012)	DSOP	VNS and SA
Varakantham and Kumar (2013)	DSOP	MILP-SAA
Evers et al. (2014)	OPSW	SAA and Monte Carlo
Zhang et al. (2014)	SOPTW	VNS
Verbeeck et al. (2016)	OPSWTW	SACS

using a dynamic programming approach. On average, the SOPTW solution improves the deterministic approach solution by 9.2%. For the OPSWTW, a stochastic ACS (SACS) is available. It takes into account the expected reward and the stochastic travel times using an estimation algorithm. Actually, it also considers time dependent travel times. The solutions are on average 23.8% better than executing the optimal solution sequence of the regular OPTW in a stochastic (and time-dependent) environment.

8.4.3 Benchmark Instances with Stochastic Travel Times

Benchmark OPSTS instances are available and generated based on the benchmark OP instances by including the distributions of the travel times and some penalty values. Also some OPSW benchmark instances are available and generated by modifying the well-known OP instances. The SOPTW benchmark instances are generated from well-known VRPTW instances. The details of how to modify known instances can be found in the related literature mentioned below.

Two sets of instances are available online. The benchmark instances for the SOPTW are available at http://ir.uiowa.edu/tippie_pubs/61/ and the instances for the OPSWTW are available at http://www.mech.kuleuven.be/en/cib/op/. These are realistic benchmark instances of varying size and properties. These instances are constructed based on an actual large road network in Belgium, the Netherlands, and Luxembourg with historic travel time profiles for every road segment.

8.4.4 Stochastic Profits

In the **OP with Stochastic Profits (OPSP)**, uncertainties about the collected profits are considered. The profits associated with the nodes are stochastic with a known distribution. Each node $i \in N$ (not considering the start and end nodes) typically has

a Normally distributed random profit $\tilde{P}_i$ with mean μ_i and σ_i^2. Given a target profit level, K, the objective function of the OPSP is to maximize the probability that the sum of the profits associated with the selected nodes is greater than or equal to K without violating the available time budget constraint T_{max}.

The objective function can then be as follows:

$$\text{Maximize } P\left(\sum_{i=2}^{|N|-1}\sum_{j=2}^{|N|}\tilde{P}_i X_{ij} \geq K\right). \tag{8.13}$$

An exact algorithm is available to solve this problem by dynamically including the sub-tour elimination constraints. So, initially, the problem is solved without sub-tour elimination constraints and then those constraints are only added to eliminate the sub-tours that appear in the solution. The algorithm is tested on four different benchmark sets. It is shown that the stochastic and deterministic solutions are quite different from each other. Moreover, the difference between the optimal stochastic objective value and the value of solving the deterministic OP with expected profits is high. Larger OPSP instances are solved using a bi-objective Genetic Algorithm. It tries to maximize the mean profit and to minimize the variance of the profit at the same time.

8.4.5 Related Literature

The papers mentioned in Table 8.3 above each present more details on OPs with stochastic travel times and solution approaches. Benchmark instances are available in Campbell et al. (2011), Evers et al. (2014), Zhang et al. (2014), and Verbeeck et al. (2016). Actually, the paper of Verbeeck et al. (2016) discusses the stochastic and time dependent OP with time windows and focuses on the impact of time windows on the complexity of the time-dependent stochastic travel time calculation.

The OP with stochastic profits is discussed in more detail in Ilhan et al. (2008).

8.5 Inventory Orienteering Problem

Recently, a lot of attention on regular vehicle routing problems goes to the so-called inventory routing problems (IRPs). Typically, these routing problems also consider the handling and inventory costs at the customers. This corresponds to logistic service providers not only distributing products from a central depot to their customers, but also managing the inventories at these customers. In the literature, this is called a 'vendor managed inventory' (VMI) problem, where the logistic service provider coordinates both the routing of the vehicles for the distribution and the timely replenishment of inventories at the customers. In general, the idea of integrating routing

and inventory management allows to reduce the costs compared to the traditional strategy of the separated vehicle routing, done by the supplier, and the management of inventory, done by the customers.

The cyclic inventory routing problem (CIRP) is a well-known variant of the general IRP. The CIRP is an appropriate optimisation model for a VMI policy when customer demand rates are stable and the planning horizon is infinite. For the CIRP, the objective function is to minimize the long term transportation and inventory costs. When thinking about an inventory routing variant of routing problems with profits, a specific variant of the CIRP should be mentioned: the single-vehicle cyclic inventory routing problem (SV-CIRP). For this problem, the demand rate is constant and a cyclic distribution plan should be developed for a single vehicle starting and ending at a single depot. The goal of the SV-CIRP is to maximize the collected profits minus the total cost, i.e., the addition of transportation and inventory costs, by determining the quantity to be delivered to the selected customers and the vehicle routes, while avoiding stock-outs. The SV-CIRP considers a set of potential customers, each with a demand rate, inventory and handling costs, and a fixed profit when selected. Other distribution aspects to be considered are the travel times between customers, the travel cost, the vehicle cost, the average speed, and the capacity of the vehicle. The main variable is the cycle time, i.e., the time between two deliveries to each customer. An important property of the SV-CIRP is that the single vehicle is allowed to make multiple trips from the depot within one cycle.

In the literature, the SV-CIRP is also called the '**inventory orienteering problem**' (IOP) and it is considered as the inventory routing variant of the OP. If transportation, inventory, and handling costs are added to the standard OP, the result will be the SV-CIRP, except that for the SV-CIRP no upper bound for the cycle time is fixed beforehand. However, certainly the IOP would better model reality than the SV-CIRP, since the IOP can limit the cycle time to a given upper bound, e.g., a number of hours or a working day. Actually, this upper bound would only make the problem easier to solve, due to a smaller number of possible solutions. Since the objective is to maximize profit minus costs and these costs include transportation costs, the problem could also be considered as an 'inventory profitable tour problem'. However, the transportation costs considered in the SV-CIRP are slightly different from the typical travel costs considered in the PTP, since these also include a fixed vehicle operating cost.

The SV-CIRP considers a set N of potential customers ($i = 1, \ldots, |N|$). In this case, the depot has index $i = 0$. Each customer has an inventory cost (s_i in euro/tonnes hour), a handling cost (h_i in euro/delivery), a demand rate (d_i in tonnes/h), and a profit (P_i in euro/h). This profit is collected when the customer is included in the distribution plan. The time required to travel between two customers i and j, or between any customer and the depot, is given by t_{ij} and is considered constant. Other aspects related to distribution that need to be taken into account are the fixed vehicle operating cost (w in euro/h), the average speed of the vehicle (v in km/h), the travel cost (tc in euro/km), and the vehicle capacity (CAP in tonnes). It is assumed that every customer has an infinite inventory capacity. An important decision variable will be the cycle time CT: the time between two deliveries to each customer.

The goal of the SV-CIRP is to minimize the total cost (*Cost* in euro/h), by determining the quantity to be delivered to each customer and the vehicle routes, while avoiding stock-outs at the selected customers. *Cost* takes into account all the above mentioned costs and rewards and can be expressed as a function of the cycle time CT (with S as the set of selected customers and T_{tot} as the total travel time):

$$Cost(CT) = w + \frac{\sum_{i \in S} h_i}{CT} + CT * \frac{\sum_{i \in S} (d_i * s_i)}{2} + \frac{tc * v * T_{tot}}{CT} - \sum_{i \in S} P_i. \tag{8.14}$$

In the SV-CIRP, due to the multiple trips (k trips) a vehicle can make during one cycle, each starting from the depot, the total travel time per cycle T_{tot} is determined by the sum of all trip travel times. Each trip must respect the vehicle capacity. This multiple trip possibility makes the problem much more complicated. To make the problem somewhat easier to understand, a simple SV-CIRP example is provided, discussed in Zhong and Aghezzaf (2012) and Vansteenwegen and Mateo (2014), and is illustrated in Fig. 8.1.

In the example, the fixed cost of using a vehicle is 20 euro/h and it travels at 50 km/h at a travel cost of 1 euro/km. The vehicle capacity is 50 tonnes. The demand rates and rewards per customer as well as the trip times are indicated in Fig. 8.1. For each customer, the holding cost is 0.5 euro/tonnes hour and the fixed handling cost is 20 euro/cycle. A feasible solution consists out of the following three trips (the depot is indicated with a '0'): (0, a, b, 0); (0, d, e, 0); (0, g, 0). A cycle time of 16.67 h minimizes the total cost for this solution to $[20 + (100/16.67) + (16.67 \times 8 \times 0.5/2) + (1 \times 50 \times 15/16.67) - 192 =] - 87.67$ euro/h. A better solution in this case has only two trips: (0, a, b, c, 0); (0, d, f, g, 0). A cycle time of 12.5 h minimizes the total cost for this solution to $[20 + (120/12.5) + (12.5 \times 8 \times 0.5/2) + (1 \times 50 \times 12/12.5) - 212 =] - 109.4$ euro/h.

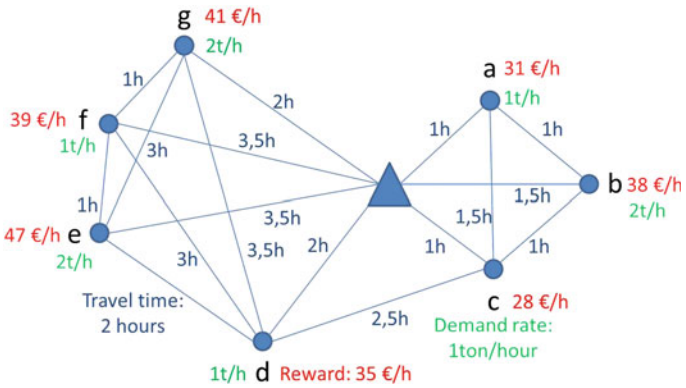

Fig. 8.1 A simple SV-CIRP example with 7 customers

When solving the SV-CIRP, a number of questions need to be answered in an integrated way: Which customers should be selected? Which cycle time would minimize the combination of inventory, handling, and routing costs? How should the selected customers be divided in multiple trips for a single vehicle? This turns out to be a very challenging problem.

More details on the SV-CIRP (or IOP), benchmark instances and solution approaches can be found in Zhong and Aghezzaf (2012), Aghezzaf et al. (2012), Vansteenwegen and Mateo (2014), and Lefever et al. (2016).

References

Abbaspour RA, Samadzadegan F (2011) Time-dependent personal tour planning and scheduling in metropolises. Expert Syst Appl 38(10):12439–12452

Aghezzaf EH, Zhong Y, Raa B, Mateo M (2012) Analysis of the single-vehicle cyclic inventory routing problem. Int J Syst Sci 43:2040–2049

Archetti C, Feillet D, Hertz A, Speranza MG (2009) The capacitated team orienteering and profitable tour problems. J Oper Res Soc 60:831–842

Archetti C, Bianchessi N, Speranza MG (2013a) The capacitated team orienteering problem with incomplete service. Optim Lett 7(7):1405–1417

Archetti C, Bianchessi N, Speranza MG (2013b) Optimal solutions for routing problems with profits. Discrete Appl Math 161(4–5):547–557

Archetti C, Bianchessi N, Speranza MG (2014a) The split delivery capacitated team orienteering problem. Networks 63(1):16–33

Bérubé JF, Gendreau M, Potvin JY (2009b) An exact e-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits. Eur J Oper Res 194:39–50

Campbell AM, Gendreau M, Thomas BW (2011) The orienteering problem with stochastic travel and service times. Ann Oper Res 186(1):61–81

Dewilde T, Cattrysse D, Coene S, Spieksma F, Vansteenwegen P (2013) Heuristics for the traveling repairman problem with profits. Comput Oper Res 40:1700–1707

Evers L, Glorie K, van der Ster S, Barros AI (2014) A two-stage approach to the orienteering problem with stochastic weights. Comput Oper Res 43:248–260

Fomin FV, Lingas A (2002) Approximation algorithms for time-dependent orienteering. Inf Process Lett 83:57–62

Garcia A, Vansteenwegen P, Arbelaitz O, Souffriau W, Linaza MT (2013) Integrating public transportation in personalised electronic tourist guides. Comput Oper Res 40(3):758–774

Garcia A, Arbelaitz O, Vansteenwegen P, Souffriau W, Linaza MT (2010) Hybrid approach for the public transportation time dependent orienteering problem with time windows. In: Corchado E, Romay MG, Savio AM (eds) Hybrid artificial intelligence systems. Lecture notes on artificial intelligence, vol 6077. Berlin, Germany, pp 151–158

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G, Vathis N (2014b) Efficient heuristics for the time dependent team orienteering problem with time windows. In: Gupta P, Zaroliagis C (eds) Proceedings of the first international conference on applied algorithms (ICAA 2014). Lecture notes in computer science, vol 8321. Springer, pp 152–163

Gunawan A, Yuan Z, Lau HC (2014) A mathematical model and metaheuristics for time dependent orienteering problem. In: Proceedings of the 10th international conference of the practice and theory of automated timetabling (PATAT 2014), pp 202–217

Ilhan T, Iravani SMR, Daskin MS (2008) The orienteering problem with stochastic profits. IIE Trans 40(4):406–421

Lau HC, Yeoh W, Varakantham P, Nguyen DT, Chen H (2012) Dynamic stochastic orienteering problems for risk-aware applications. In: Proceedings of the twenty-eighth conference annual conference on uncertainty in artificial intelligence (UAI-12). AUAI Press, Corvallis, Oregon, pp 448–458

Lefever W, Aghezzaf EH, Hadj-Hamou K (2016) A convex optimization approach for solving the single-vehicle cyclic inventory routing problem. Comput Oper Res 72:97–106

Li J (2012) Research on team orienteering problem with dynamic travel times. J Softw 7(2):249–255

Matl P, Nolz P, Ritzinger U, Ruthmair M, Tricoire F (2017) Bi-objective orienteering for personal activity scheduling. Comput Oper Res 82:69–82

Papapanagiotou V, Montemanni R, Gambardella LM (2014) Objective function evaluation methods for the orienteering problem with stochastic travel and service times. J Appl Oper Res 6(1):16–29

Peng G, Dewil R, Verbeeck C, Gunawan A, Xing L, Vansteenwegen P (2019) Agile earth observation satellite scheduling: an orienteering problem with time-dependent profits and travel times. Comput Oper Res 111:84–98

Schilde M, Doerner KF, Hartl RF, Kiechle G (2009) Metaheuristics for the biobjective orienteering problem. Swarm Intell 3(3):179–201

Souffriau W, Vansteenwegen P, Vanden Berghe G, Van Oudheusden D (2013) The multiconstraint team orienteering problem with multiple time windows. Transp Sci 47(1):53–63

Tarantilis CD, Stavropoulou F, Repoussis PP (2013) The capacitated team orienteering problem: a bi-level filter-and-fan method. Eur J Oper Res 224(1):65–78

Vansteenwegen P, Mateo M (2014) An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. Eur J Oper Res 237(3):802–813

Varakantham P, Kumar A (2013) Optimization approaches for solving chance constrained stochastic orienteering problems. In: Perny P, Pirlot M, Tsoukiàs A (eds) Proceedings of the third international conference in algorithmic decision theory (ADT2013). Lecture notes in computer science, vol 8176. Springer, pp 387–398

Verbeeck C, Sörensen K, Aghezzaf EH, Vansteenwegen P (2014a) A fast solution method for the time-dependent orienteering problem. Eur J Oper Res 236(2):419–432

Verbeeck C, Vansteenwegen P, Aghezzaf EH (2016) Solving the stochastic time-dependent orienteering problem with time windows. Eur J Oper Res 255:699–718

Wang X, Golden BL, Gulczynski D (2014) A worst-case analysis for the split delivery capacitated team orienteering problem with minimum delivery amounts. Optim Lett 8(8):2349–2356

Zhang S, Ohlmann JW, Thomas BW (2014) A priori orienteering with time windows and stochastic wait times at customers. Eur J Oper Res 239(1):70–79

Zhong Y, Aghezzaf EH (2012) Effective local search approaches for the single-vehicle cyclic inventory routing problem. Int J Serv Oper Inf 7(4):260–279