Otto von Guericke University

Faculty of Computer Science

Master's Thesis

In-Depth Analysis and Characteristics of the Traveling Thief Problem

Author:

Julian Blank

May 2, 2016

Advisors:

Prof. Dr.-Ing. habil. Sanaz Mostaghim

Faculty of Computer Science
Otto von Guericke University

Prof. Dr. Kalyanmoy Deb

Computer Science and Engineering Department
Michigan State University

Blank, Julian:
In-Depth Analysis and Characteristics
of the Traveling Thief Problem
Master's Thesis, Otto von Guericke University, 2016.

Abstract

This thesis presents an in-depth analysis and characteristics of the quite new and complex Traveling Thief Problem, where the well-known Traveling Salesman Problem and Knapsack Problem interact. Since the Traveling Thief Problem is an interwoven system, the interaction between the components is explained and visualized in detail. The single as well as the multi-objective formulation and the different models of the problem are defined and an example scenario is presented in order to show how the objective values, time and profit, are calculated. Furthermore, we propose another model, which uses an extra constraint instead of a weight vector for composing the two objectives, and is therefore embeddable in a multi-objective ϵ-constraint algorithm. To encourage doing research on the multi-objective problem, characteristics of the two-dimensional objective space are presented and the usefulness of algorithms for the interacting subproblems is investigated. Moreover, three problem solving strategies for the single-objective problem are proposed and existing algorithms are classified into these categories. Also, some current research questions using experimental studies on small-scale instances are answered and different solutions of a case study found by state-of-the-art algorithms are visualized. All analyses focus on the interdependence and interwovenness of the components. The obtained results provide insights into characteristics of the single and multi-objective Traveling Thief Problem and new starting points for ongoing research.

Acknowledgements

I would like to thank my first advisor Prof. Dr.-Ing habil. Sanaz Mostaghim for the continuous support of my Master's Thesis. You were readily available with help and advice and, even though I was far away overseas, you never stopped supporting me. Our meetings were effective and efficient, and helped me to keep track of my current research progress. Furthermore, I would like to thank my second advisor Prof. Dr. Kalyanmoy Deb for inviting me to the Computational Optimization and Innovation (COIN) Laboratory at the Michigan State University (MSU). Our discussions about a challenging problem helped me to study different facets and steered me in the right direction whenever I needed it. Also, thank you for inviting me to the Beacon Conference.

In addition to my advisors, I must express my very profound gratitude to my whole family. Especially, I want to say thanks to my mother, Barbara Blank-Brückner, for supporting me morally and financially, and my stepfather, Matthias Brückner, for introducing me into technology in my younger days and supporting all my interests. Also, thanks to my grandparents, Johanna und Rudi Blank, for providing me a quiet place to focus on my work and giving me a feeling of comfort. Moreover, I want to thank all members of the COIN laboratory for welcoming me in the working group and for all presentations and discussions. Particularly, thanks to Gregorio Toscano Pulido and his family for your support whenever I needed it and Proteek Chandan Roy as well as Abhinav Gaur for giving me some more inspiration for my work. Furthermore, thanks to Heiner Zille, who helped me especially in the beginning of my thesis and supported me writing the proposal. Additionally, thanks to my temporary roommate Lauren Zepeda and her parents Jeri-Ann and Dave Cypher for the invitation to Christmas and giving me insights into American traditions during my time as a visiting scholar. I also thank my friends from the coffee hour and everybody who joined our group afterwards. Especially thanks to Shoko Hiruta and Irwin Moksin for our mind clearing road trips. These helped me to get a new viewpoint to my work after having some great days. Also thanks to all my friends in Germany who were aware of my situation in a foreign country and welcomed me again as if I had never been gone. Moreover, I want to say thanks to my reviewers, Tom Palmer, Gregorio Toscano Pulido, Sascha Hock and Tobias Kranz, who helped me to finish the final steps of this thesis.

Lastly, I want to mention that this work was supported by a fellowship within the FITweltweit programme of the German Academic Exchange Service (DAAD). I want to say thanks for making my exchange year at the Michigan State University possible.

Contents

List of Figures

List of Tables

Acronyms

EA Evolutionary Algorithm

GA Genetic Algorithm

KNP Knapsack Problem

MOP Multi-Objective Problem

TSP Traveling Salesman Problem

TTP Traveling Thief Problem

SH Simple Heuristic

RLS Random Local Search

1+1 EA (1+1) Evolutionary Algorithm

DH Density Heuristic

MA Memetic Algorithm

CC Cooperative Coevolution

TSMA Two Stage Memetic Algorithm

ACO Ant Colony Optimization

VRP Vehicle Routing Problem

1. Introduction

People are facing real-world problems with dependencies and interwovenness every day. Some researchers claim that there is a gap between theory and practice, because either the problem or the benchmarks do not reflect real-world characteristics [BMB13]. Therefore, to increase the complexity of benchmarks, problem instances are scaled up to come closer to real-world problems. For instance, researchers have investigated the well-known Traveling Salesman Problem with 25,000,000 cities [ACR03].

Nonetheless, assumptions and simplifications are often made when real-world problems are reduced to known and well investigated NP-hard optimization problems. For this reason, theorists are accused of solving toy problems and missing reality. Therefore, a bunch of new problems have been investigated, for example the Vehicle Routing Problem (VRP) [TV01]. Also, numerous variations of this problem exist, where either new constraints or new objectives are added. For instance, the fuel consumption as a constraint or the satisfaction of the customers as an additional objective was introduced. The VRP variants could be considered as a start to model real-world problems and make the research more practical for the industry. Although, these problems are extensions of existing problems, there are cases in the real world, where several of these problems interact interdependently. Because of the interdependence, solving each problem by its own will not lead to an optimal solution. In general, little research for interwoven and interdependent problems has been done so far.

In order to provide a problem with real-world characteristics, this thesis presents the Traveling Thief Problem (TTP), where the well-known Traveling Salesman Problem (TSP) and Knapsack Problem (KNP) interact. Research on the TTP will give insights into how to handle problems with interwovenness and therefore how to solve problems with real-world characteristics. First steps for the TTP were made and some algorithms for solving the problem were proposed. Also, a large-scale benchmark for the TTP was published and investigated by researchers.

1.1 Research Goals

The goal of this thesis is to analyze the Traveling Thief Problem in-depth. Therefore, the interdependence of the problem is investigated and the more complex multi-objective model is considered. Furthermore, the following specific goals should be achieved:

Complexity of the Traveling Thief Problem: It has to be investigated why the Traveling Thief Problem is very hard to solve. Therefore, the interdependence of the two subproblems has to be considered and should be clearly shown in order to solve the problem appropriately.

Models of the Traveling Thief Problem: Different models of the Traveling Thief Problem with different characteristics exist. The dimensionalities of the objective space and the interaction equations are different. This work aims to clarify the relation between these models and also to explain the relation between them. Furthermore, some mixed forms should be introduced and their characteristics investigated.

Multi-Objective Traveling Thief Problem: Another goal is to analyze the objective space of the multi-objective problem which is helpful in order to propose new algorithms. This thesis aims to build the bridge between single and multi-objective optimization and to outline the correlation from the two-dimensional to the one-dimensional space. It should encourage doing research on the multi-objective problem.

Problem Solving Strategies: Furthermore, this work aims to propose different problem solving strategies where the Traveling Thief Problem is modeled in different ways. Additionally, existing algorithms should be assigned to these categories.

Experimental Study and Visualization of Results: Some further experiments should answer current research questions about the problem. These insights are useful for the design of further algorithms. Moreover, the thesis aims to show some solutions of a case study. These visualizations should give an idea what good solutions look like.

1.2 Document Organization

The outline of this work is as follows: The current chapter motivates the investigations on the TTP and shows the goals of this thesis. Chapter 2 presents the background and definitions. It includes an introduction into optimization in general and defines single as well as multi-objective problems. Moreover, the two well-known problems, the TSP and KNP, are explained. These two problems are combined to the TTP, where the interdependencies and different models are described. Chapter 3 presents related work, while Chapter 4 highlights the characteristics of the TTP and its objective space. Followed by Chapter 5, where problem solving strategies are proposed and the problem itself is analyzed. Lastly, the thesis presents further consideration in Chapter 6 and is concluded in Chapter 7.

2. Background

This chapter provides the background of the TTP. Therefore, single as well as multi-objective optimization is introduced and the foundation is laid. Moreover, the subproblems and the TTP itself are explained and an example scenario is given.

2.1 Optimization

Optimization is done in order to make something as effective as possible. This can be useful in many situations and not least in our everyday life. In the following, definitions and principles of optimization are explained.

2.1.1 Single-Objective Optimization

Mathematically, single-objective optimization aims to find the minimum or maximum of a function. For example in economy the aim of a company is to maximize the profit at the end of a year. But the first challenge is to formulate the problem in equations.

Single-objective optimization problems are defined according to Equation 2.1. A function $f(\vec{x})$ is minimized or maximized. As parameter vector $\vec{x}$ is given, where the search domain is limited by the lower x_i^L and upper bounds x_i^U of each x_i. Furthermore, inequality constraints $g_j(\vec{x})$ and equality constraints $h_k(\vec{x})$ could be defined.

$$\begin{aligned} min/max \quad & f(\vec{x}) && \\ s.t. \quad & g_j(\vec{x}) \geq 0 && j = 1, 2, ..., J; \\ & h_k(\vec{x}) = 0 && k = 1, 2, ..., K; \\ & x_i^L <= x_i <= x_i^U && i = 1, 2, ..., N; \end{aligned} \tag{2.1}$$

However, the limitation is that only one objective can be optimized. If several objectives exist, they have to be composed to one objective using a composing function. A naive approach is to define a weight for each objective and calculate the weighted sum.

2.1.2 Multi-Objective Optimization

Multi-objective optimization has multiple functions to optimize and the goal is to find solutions that are as good as possible according to all given objectives [Deb01]. Since the objective space could have more than one dimension, the fitness of a solution is more difficult to determine. For this reason the problem formulation and also the domination relation of solutions are explained in the following.

The definition in Equation 2.2 is similar to Equation 2.1. But instead of only one function, multiple functions $f_m(\vec{x})$ are considered. A single-objective is equal to a multi-objective problem when $M = 1$.

$$\begin{aligned} min/max \quad & f_m(\vec{x}) && m = 1, 2, ..., M; \\ s.t. \quad & g_j(\vec{x}) \geq 0 && j = 1, 2, ..., J; \\ & h_k(\vec{x}) = 0 && k = 1, 2, ..., K; \\ & x_i^L <= x_i <= x_i^U && i = 1, 2, ..., N; \end{aligned} \tag{2.2}$$

Since there is normally more than one objective, the function $\vec{f}(\vec{x})$ projects $\vec{x}$ in a space with a dimensionality larger than one. The connection between decision and objective space is illustrated in Figure 2.1. The three-dimensional vector $\vec{x}$ is mapped to a two-dimensional vector $\vec{z}$.

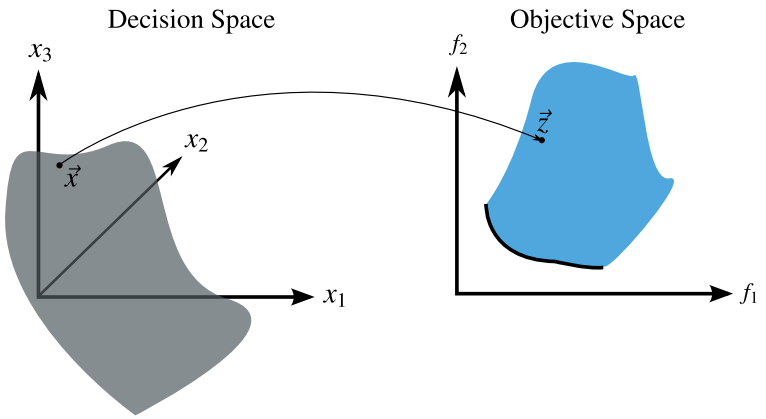


Figure 2.1: Multi-Objective Optimization: From Decision to Objective Space [Deb01]

The aim is to find good solutions with trade-offs instead of a single solution [CLV06]. In the resulting solution set no solution should dominate the other. If a minimization problem is considered, solution $\vec{u}$ dominates solution $\vec{v}$ if

$$\forall i \; f_i(\vec{u}) \leq f_i(\vec{v}) \quad \wedge \quad \exists i \; f_i(\vec{u}) < f_i(\vec{v}) \qquad i \in \{1, ...M\} \tag{2.3}$$

which means $\vec{f}(\vec{u})$ is partially less than $\vec{f}(\vec{v})$ denoted by $\vec{f}(\vec{u}) \preceq \vec{f}(\vec{v})$. Furthermore, there is a set of solutions where no solution dominates the other, the so called Pareto Optimal Set

$$P^* := \{\vec{x} \in \Omega \mid \nexists \vec{y} \in \Omega \quad \vec{f}(\vec{y}) \preceq \vec{f}(\vec{x})\} \tag{2.4}$$

where Ω is a set with all possible solutions that do not violate any constraint. The Pareto front

$$PF^* := \{\vec{f}(\vec{x}) \mid \vec{x} \in P^*\} \tag{2.5}$$

is given by the vectors in the objective space of the Pareto optimal set [CLV06]. In Figure 2.1 the Pareto front is visualized by the bold line in the objective space when f_1 and f_2 are minimized. All other solutions are dominated by solutions on that line.

For the purposes of illustration let us consider the decision of buying a new car [Deb01]. The decision is made based on two properties of the car, namely the price and horsepower. We assume that customers prefer a cheap car with much horsepower. Therefore, the price is minimized and the horsepower maximized. The customers compare advertisements and create a list with four cars:

Model	**Price in $**	**Horsepower in HP**
Car 1	8.000	50
Car 2	20.000	80
Car 3	14.000	90
Car 4	30.000	140

Table 2.1: Multi-Objective Optimization: Car Example

There are two objectives and some solutions are indifferent to each other. For instance, *Car 1* is indifferent to *Car 2* because it is cheaper but has less horsepower. The relation between *Car 2* and *Car 3* is different, because *Car 3* is cheaper and has more horsepower. In terms of multi-objective optimization, *Car 3* dominates *Car 2*. For this reason, customers should not buy *Car 2* based on the given properties since it is not part of the Pareto front.

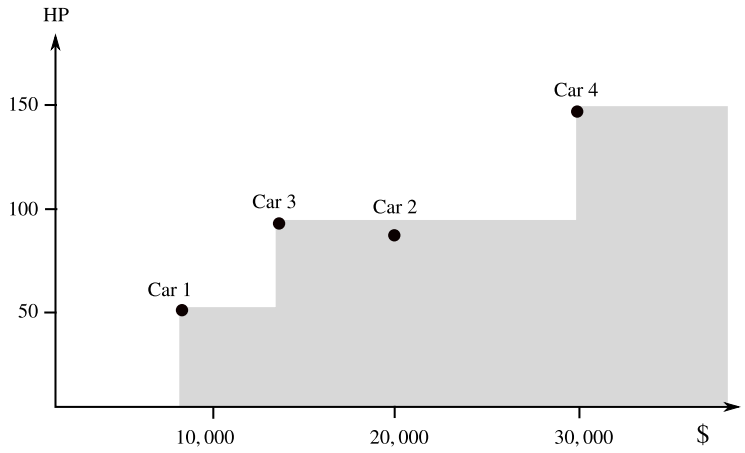


Figure 2.2: Objective Space of the Car Buying Problem Example

In Figure 2.2 the objective space of the car example is illustrated. All cars are represented by a point. The grey area contains all points which are dominated by the Pareto front of the given example. Since *Car 2* lies in the grey part, it cannot be part of the Pareto front which is built by the other cars.

2.2 Traveling Salesman Problem

In the TSP [ABCC07] a salesman has to visit n cities. The distances are given by a map represented as a distance matrix $A = (d_{ij})$ with $i, j \in \{0, .., n\}$. The salesman has to visit each city once and the result is a permutation vector $\pi = (\pi_1, \pi_2, ..., \pi_n)$ [1], where π_i is the i-th city of the salesman. The distance between two cities divided by a constant velocity v results in the traveling time for the salesman denoted by $f(\pi)$. The goal is to minimize the total traveling time of the tour:

$$\begin{aligned} min \quad & f(\pi) \\ s.t. \quad & \pi = (\pi_1, \pi_2, ..., \pi_n) \in P_n \\ f(\pi) \quad = \quad & \sum_{i=1}^{n-1} \frac{d_{\pi_i, \pi_{i+1}}}{v} + \frac{d_{\pi_n, \pi_1}}{v} \end{aligned} \tag{2.6}$$

There are $\frac{(n-1)!}{2}$ different tours to consider, if we assume that the salesman has to start from the first city and travels on a symmetric map where $d_{i,j} = d_{j,i}$.

[1] The permutation vector is denoted by π instead of $\vec{\pi}$.

2.3 Knapsack Problem

For the KNP [Lag96] a knapsack has to be filled with items without violating the maximum weight constraint. Each item j has a value $b_j \geq 0$ and a weight $w_j \geq 0$ where $j \in \{1, .., m\}$. The binary decision vector $z = (z_1, .., z_m)$ [2] defines, if an item is picked or not. The aim is to maximize the profit $g(z)$:

$$\begin{aligned} max \quad & g(z) \\ \text{s.t.} \quad & \sum_{j=1}^{m} z_j w_j \leq Q \\ & z = (z_1, .., z_m) \in \mathbb{B}^m \\ g(z) \quad = \quad & \sum_{j=1}^{m} z_j b_j \end{aligned} \tag{2.7}$$

The search space of this problem contains 2^n combinations.

2.4 Traveling Thief Problem

The TTP is a combinatorial optimization problem that consists of two interweaving problems such as TSP and KNP [IKM+15]. After explaining the two components separately, the interdependence and the different models of the problem are described.

2.4.1 Interdependencies

The TTP combines the above defined subproblems and let them interact together. The traveling thief can collect items from each city he is visiting. The items are stored in a knapsack carried by him. In more detail, each city π_i provides one or multiple items, which could be picked by the thief. There is an interaction between the subproblems: The velocity of the traveling thief depends on the current knapsack weight w, which is carried by him. It is calculated by considering all cities, which were visited so far, and summing up the weights of all picked items. The weight at city i given π and z is calculated by:

$$w(i, \pi, z) = \sum_{k=1}^{i} \sum_{j=1}^{m} a_j(\pi_k)\ w_j z_j \tag{2.8}$$

The function $a_j(\pi_k)$ is defined for each item j and returns 1 if the item could be stolen at city π_k and 0 otherwise. The current weight of the knapsack has an influence on the velocity.

[2] The binary vector is denoted by z instead of $\vec{z}$.

When the thief picks an item, the weight of the knapsack increases and therefore the velocity of the thief decreases.

The velocity v is always in a specific range $v = [v_{min}, v_{max}]$ and could not be negative for a feasible solution. Whenever the knapsack is heavier than the maximum weight Q, the capacity constraint is violated. However, to provide also the traveling time for infeasible solutions the velocity is set to v_{min}, if $w > Q$:

$$v(w) = \begin{cases} v_{max} - \frac{v_{max} - v_{min}}{Q} \cdot w & \text{if } w \leq Q \\ v_{min} & \text{otherwise} \end{cases} \tag{2.9}$$

If the knapsack is empty the velocity is equal to v_{max}. Contrarily, if the current knapsack weight is equal to Q the velocity is v_{min}. Figure 2.3 illustrates an example for the velocity function above. For heavier weights $w > 10$, the minimum velocity is set to be 0.1, while the velocity linearly decreases by the increasing amount of weights for ($w \leq Q$). To calculate the velocity at each city, the weight at each city according to π has to be known.

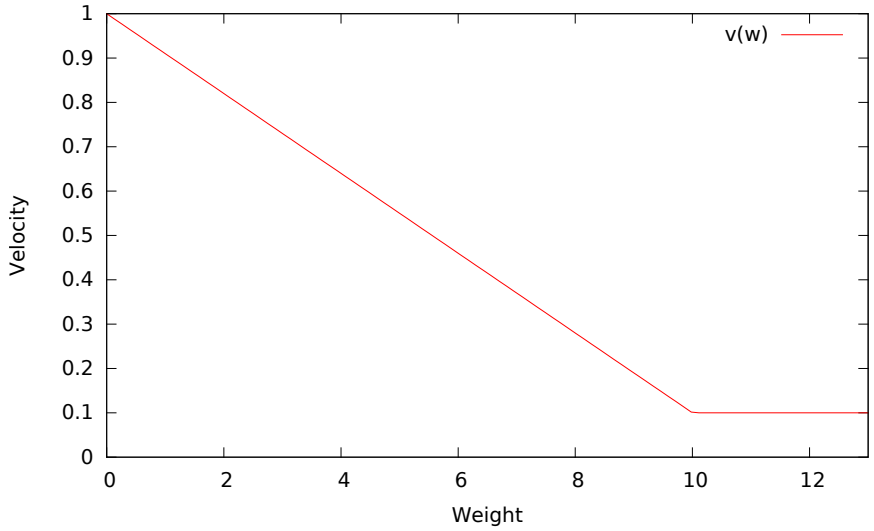


Figure 2.3: Velocity Function $v(w)$ with $v_{min} = 0.1$, $v_{max} = 1$ and $Q = 10$

Furthermore, the traveling time of the thief is calculated by:

$$f(\pi, z) = \sum_{i=1}^{n-1} \frac{d_{\pi_i, \pi_{i+1}}}{v(w(i, \pi, z))} + \frac{d_{\pi_n, \pi_1}}{v(w(n, \pi, z))} \tag{2.10}$$

The calculation is based on Equation 2.6, but the velocity is defined by a function instead of a constant value. This function takes the current weight, which depends on the index i of the tour. The current weight, and therefore also the velocity, will change on the tour by

considering the picked items defined by z. In order to calculate the total tour time, the velocity at each city needs to be known. For calculating the velocity at each city the current weight of the knapsack must be given. Since both calculations are based on z and z is part of the knapsack subproblem, it is very hard to solve the problem. In fact, such problems are called interwoven systems as the solution of one subproblem highly depends on the solution of the other subproblems.

The solution of KNP is influenced by TSP in the way the longer items are carried, their values denoted by b deteriorates. This is called depreciation. There are two ways of calculating the final profit. One contains no depreciation resulting in the traditional knapsack problem:

$$g(z) = \sum_{j=1}^{m} z_j \, b_j \tag{2.11}$$

and the other calculates the depreciation according to the following equation as proposed in [BMB13] which is modeled by an exponential function:

$$g(\pi, z) = \sum_{j=1}^{m} (z_j \cdot b_j \cdot D^{\frac{T_j(\pi,z)}{C}}) \tag{2.12}$$

where b_j is the initial value of the item j, D the depreciation rate ($0 < D < 1$) and C the depreciation constant. The function $T_j(\pi, z)$ returns the time, how long item j was carried by the thief.

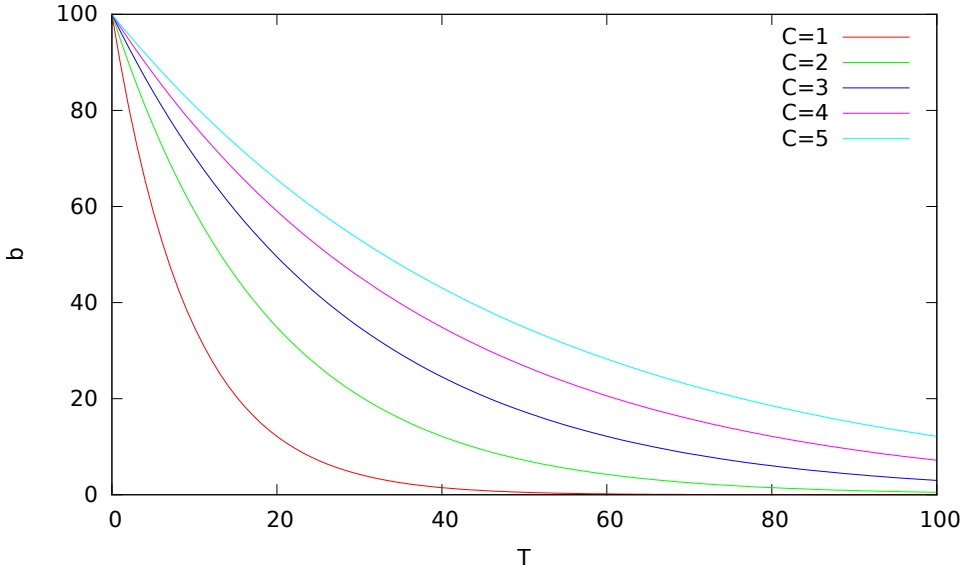


Figure 2.4: Exponential Depreciation Function for an Item with $D = 0.9$ and initial Value $b = 100$.

Some exponential depreciation functions with different depreciation constants C are illustrated in Figure 2.4. A small C means the item value loss is higher. For example if $C = 1$ and $D = 0.9$, an item i with $b_i = 100$ will lose almost the whole value after being carried 40 time units. But if $C = 5$, the item is still worth more than 40% of its value at the same point of time.

2.4.2 Models

Different models of the TTP are proposed [BMB13]. On the one hand there are multi-objective problems where the time is minimized and the profit maximized. On the other hand there are single-objective problems where time and profit are composed to one objective. Additionally, different equations for calculating the profit are used.

Originally, Model A was always a single-objective problem without depreciation, and Model B a multi-objective problem with depreciation. Furthermore, we propose Model C which will be described in Section 2.4.2.3.

Model	Version	#obj	#constraints	Profit
Model A	**single** [BMB13]	1	1	$g(z)$
	multi	2	1	$g(z)$
Model B	single	1	1	$g(\pi, z)$
	multi [BMB13]	2	1	$g(\pi, z)$
Model C	**no-depr**	1	2	$g(z)$
	depr	1	2	$g(\pi, z)$

Table 2.2: Models of the Traveling Thief Problem

Each model could be easily modified by using for example no composition of the objectives or other linking functions. However, this thesis mainly investigates the proposed models, which are printed bold in Table 2.2. To these models is referred, whenever only the model and not the objective space dimensionality or profit function type is mentioned. Note, that in [BMB13], the single-objective Model A was proposed as TTP_1 and the multi-objective Model B as TTP_2.

2.4.2.1 Model A

Model A is a single-objective problem where the influence of the two subproblems is balanced by weights. The single-objective $G(\pi, z)$ has to be maximized. The value R is used as a weight to balance the importance of the tour time and the final profit.

$$\max \; G(\pi, z) \quad = \quad 1 \cdot g(z) - R \cdot f(\pi, z) \tag{2.13}$$

The value of R is different for each problem. No depreciation is considered and $g(z)$ (cf. Equation 2.11) is used. Therefore, there is no dependency of π on the KNP component.

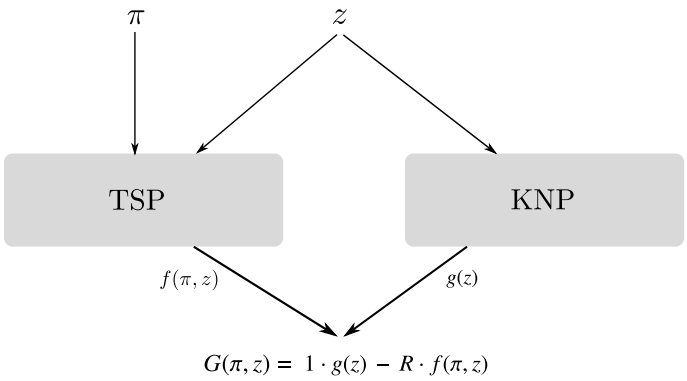


Figure 2.5: Model A: Illustration of Interdependencies

The tour π is clearly only part of the TSP component and the TSP does not share any equation with the KNP. Furthermore, the final function value is combined to one objective. However, the interwovenness is still present even if only z is a parameter for both components. Since, there is only one objective, it does not matter which variable is changed, either π or z, the objective value $G(\pi, z)$ is different.

The renting rate R provides a weight in order to combine two objectives into one. Figure 2.6 visualizes the utility function in the two-dimensional objective space. Multiple slopes with different objective values $G(\pi, z)$ exist. R defines a slope which combines all points in the objective space to one function value. Since our goal is to maximize $G(\pi, z)$, we are searching for the left most slope. Further considerations will be done in this work, but some other characteristics of the objective space need to be mentioned before.

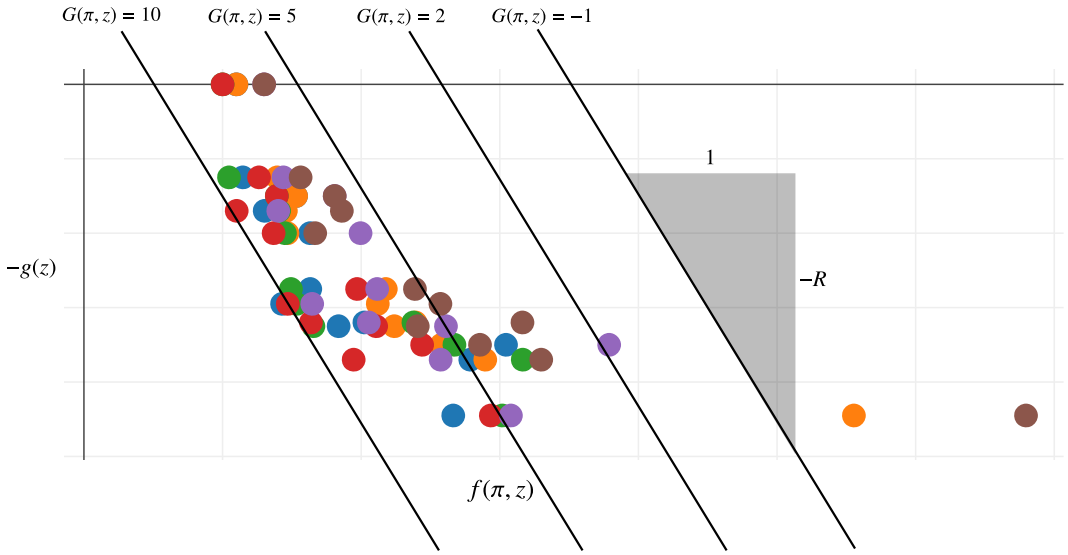


Figure 2.6: Correlation between the Single and Multi-Objective Problem

Basically, Model A does not consider depreciation and uses a function in order to combine two objectives into one. For the transition from Model A to Model B, we introduce the multi-objective Model A, which does not have a composing function and therefore two objectives (cf. Equation 2.14).

$$G(\pi, z) = \begin{cases} min & f(\pi, z) \\ min & -g(z) \end{cases} \tag{2.14}$$

All observations, which are made for the objective space on this model, are useful for Model A, because there is only the projection to the single-objective space missing. Furthermore, the multi-objective viewpoint helps to find characteristics of Model B by exploring the two-dimensional objective space.

Note, the goal is to minimize the traveling time $f(\pi, z)$ and maximize the profit $g(z)$ at the end of the tour. Instead of maximizing the profit, we convert the problem to a pure minimization problem and minimize $-g(z)$. Therefore, the y-axis will always represent the negative profit $-g(z)$. All figures in this work will show this axis labeling.

2.4.2.2 Model B

For Model B depreciation is considered and $g(\pi, z)$ (c.f. Equation 2.12) is used. As proposed for the multi-objective Model A this formulation in Equation 2.15 is a pure minimization problem.

$$G(\pi, z) = \begin{cases} min & f(\pi, z) \\ min & -g(\pi, z) \end{cases} \tag{2.15}$$

No research is done so far on this more difficult problem. Through the depreciation it is a complete interwoven system and it does not matter if the tour π or the packing plan z is changed, both objective function values are different. The interwoven system is illustrated in Figure 2.7.

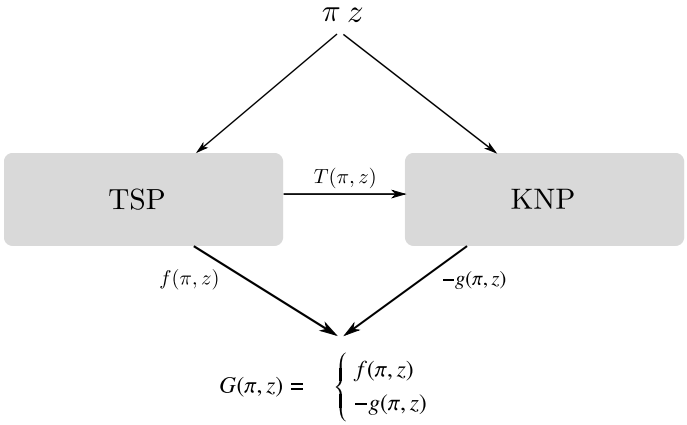


Figure 2.7: Model B: Illustration of Interdependencies

Both variables π and z are needed for the TSP as well as for the KNP component. The depreciation of the items is dependent on the carrying time $T(\pi, z)$, which is part of the TSP component. This model shows that no matter which variable is changed all components are affected. For this reason it describes a complete interwoven system.

2.4.2.3 Model C

For multi-objective problems the ϵ-constraint method is proposed to transform the multi-objective into several single-objective problems by using an extra constraint [Deb01]. We transfer this method to the TTP and introduce Model C. It is a single-objective problem, but uses a minimum profit constraint:

$$\begin{aligned} \min \quad & f(\pi, z) \\ \text{s.t.} \quad & W(z) \leq Q \\ & g(z) \geq P_{min} \end{aligned} \tag{2.16}$$

In Model C the traveling time $f(\pi, z)$ is minimized. The function $W(z)$ calculates the knapsack weight according to z. Note, either the profit constraint is $g(z) \geq P_{min}$ or $-g(z) \leq -P_{min}$. To explore the whole Pareto front, problems with different P_{min} could be solved by using a single-objective algorithm as shown in Figure 2.8.

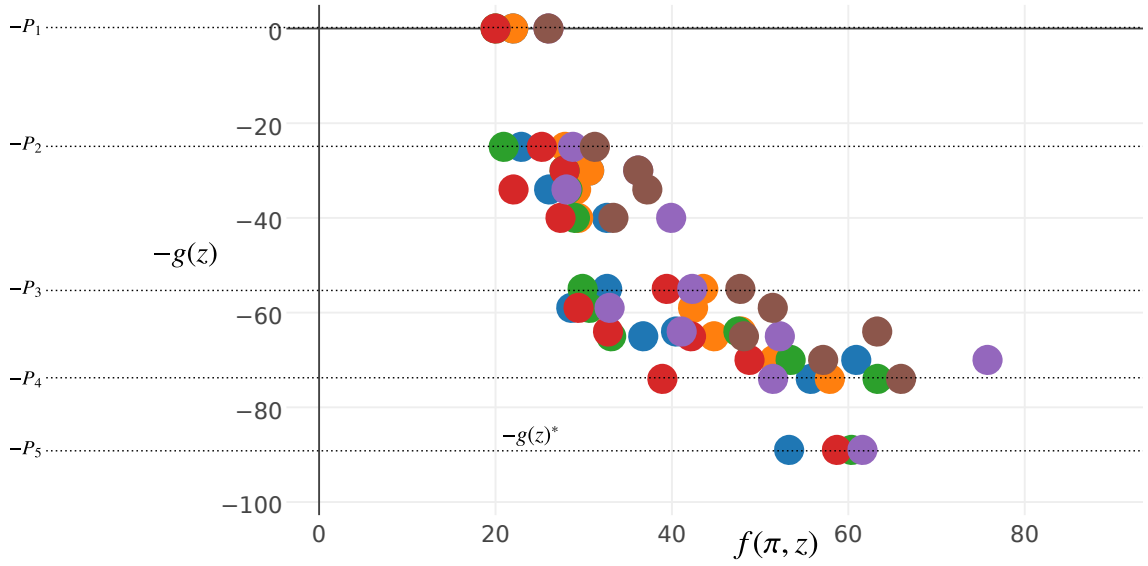


Figure 2.8: The Profit Constraints in the Objective Space

For defining P_{min} the boundaries are known to be in $[0,\ g(z^*)]$ where z^* is the optimal solution for the KNP which could be found by using knapsack solvers, for example [MPT99].

Somebody might assume Model C faces the problem of finding the best tour π given z. But by looking at Figure 2.8 it can be concluded that this assumption is not necessarily true. Let

us consider the minimum profit line $-P_3$. Solutions not on, but below the horizontal $-P_3$ line should be found in order to minimize $f(\pi, z)$. These solutions have a different z and no solutions where $g(z) = P_3$ are part of the Pareto front. Nonetheless, an approach is to fix z' where $g(z') \geq P_3$ and optimize π according to z'.

Doing research on Model C contributes directly to the multi-objective TTP, because all proposed solvers could be easily embedded in an ϵ-constraint algorithm.

2.4.3 Example Scenario

In order to illustrate the equations and interdependence, an example scenario (cf. Figure 2.9) is presented in the following. The thief starts at city 0 and has to visit city *1, 2, 3* exactly once and to return to city 0. In this example each city provides one item and the thief could decide to steal item I_j or not. For the further example we use a zero index based notation for the vectors π and z and the first city is at index 0.

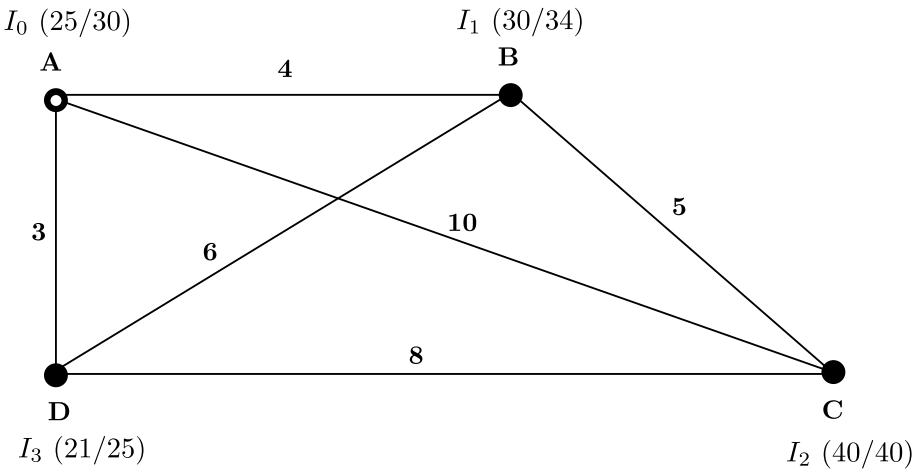


Figure 2.9: An Example Scenario for the Traveling Thief Problem

A permutation vector, which contains all cities exactly once, and a binary picking vector are needed to calculate the objectives. Even though, this is a very small example with four cities and four items the solution space consists of $(n-1)! \cdot 2^m = 6 \cdot 16 = 96$ combinations.

In order to understand how the objectives are calculated, an example hand calculation for the tour *[0,2,1,3]* and the packing plan *[0,1,0,1]* is done as follows: The thief starts with the maximum velocity, because the knapsack is empty. He begins his tour at city *0* and picks no item there. For an empty knapsack $w(0, \pi, z) = 0$ the velocity is $v(0) = v_{max} = 1.0$. The distance from city 0 to city 2 is 9.0 and the thief needs 9.0 time units. At city 2 the thief will not pick an item and continue to travel to city 1 with $w(1, \pi, z) = 0$ and therefore with v_{max} in

i	π_i	$w(i,\pi,z)$	$v(w_{\pi_i})$	$d_{\pi_i,\pi_{i+1}}$	$t_{\pi_i,\pi_{i+1}}$	$\sum$
0	0	0	1	9	9	-
1	2	0	1	5	5	9
2	1	30	0.6625	5	7.5472	14
3	3	51	0.42625	3	7.0381	21.547
4	-	-	-	-	-	**28.585**

Table 2.3: Hand calculations for *[0,2,1,3] [0,1,0,1]* on the Example Scenario

additional 5.0 time units. Here he picks item I_1 with $w_1 = 30$ and the current weight becomes $w(2,\pi,z) = 30$, which means the velocity will be reduced to $v(30) = 1.0 - (\frac{1.0-0.1}{80}) \cdot 30 = 0.6625$. For traveling from city 1 to city 3 the thief needs the distance divided by the current velocity $\frac{5.0}{0.6625} \approx 7.5472$. At city 3 he picks I_3 with $w_3 = 21$ and the current knapsack weight increases to $w(3,\pi,z) = 30 + 21 = 51$. For this reason the velocity decreases to $v(51) = 1.0 - (\frac{1.0-0.1}{80}) \cdot 51 = 0.42625$. For returning to city 0 the thief needs according to this current speed $\frac{3.0}{0.42625} \approx 7.0381$ time units. Finally, we sum up the time for traveling from each city to the next $\sum_{k=0}^{n-1} t_{\pi_k,\pi_{k+1}} = 9 + 5 + 7.5472 + 7.0381 = 28.5853$ to calculate the whole traveling time.

The final profit without depreciation is calculated by summing up the values of all items which is $34 + 25 = 59$. So the variable *[0,2,1,3] [0,1,0,1]* is mapped to the point $(28.59, 59.0)$ in the objective space. If we assume $R = 1.516$ the single-objective value is $59 - 1.516 \cdot 28.59 \approx 15.658$.

The Pareto front contains 10 solutions and our hand calculation is printed bold at Table 2.4.

Tour	Packing	$f(\pi,z)$	$g(z)$	$G(\pi,z)$
[0, 1, 2, 3]	[0, 0, 0, 0]	20.0	0.0	-30.32
[0, 3, 2, 1]	[0, 0, 0, 0]	20.0	0.0	-30.32
[0, 1, 2, 3]	[0, 0, 0, 1]	20.93	-25.0	-6.73
[0, 3, 2, 1]	[0, 1, 0, 0]	22.04	-34.0	0.587
[0, 3, 2, 1]	[0, 0, 1, 0]	27.36	-40.0	-1.478
[0, 2, 1, 3]	**[0, 1, 0, 1]**	**28.59**	**-59.0**	**15.658**
[0, 3, 2, 1]	[1, 1, 0, 0]	32.75	-64.0	14.351
[0, 1, 2, 3]	[0, 0, 1, 1]	33.11	-65.0	14.805
[0, 3, 2, 1]	[0, 1, 1, 0]	38.91	-74.0	15.012
[0, 2, 1, 3]	[1, 1, 0, 1]	53.28	-89.0	8.228

Table 2.4: Pareto front of the Example Scenario

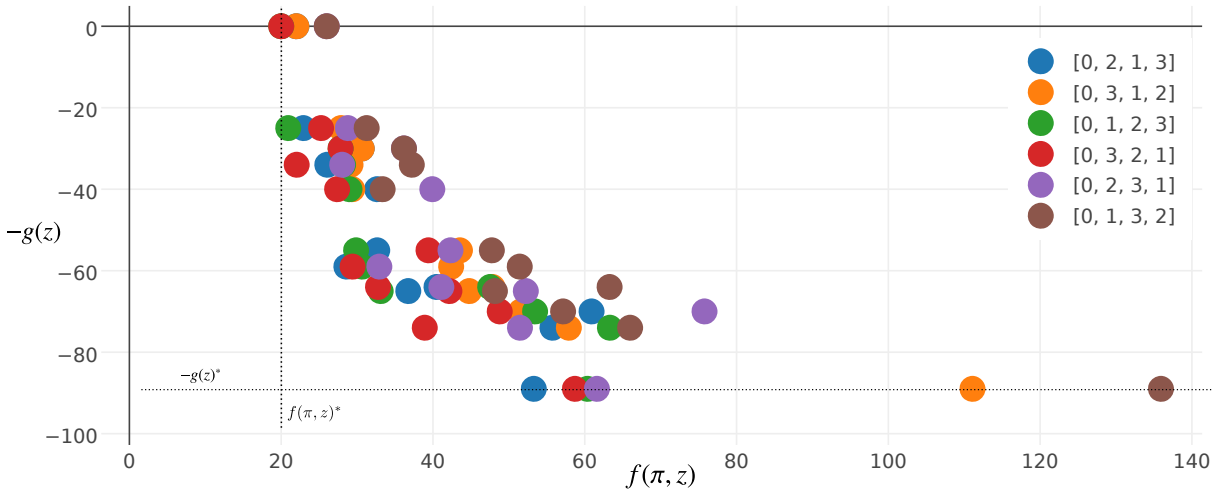


Figure 2.10: Pareto front of the Example Scenario colored by Tour

Additionally, Figure 2.10 shows the objective space colored by tour. The vertical line denoted by $f(\pi,z)^*$ illustrates solutions where the time of the thief is minimized. Furthermore, all solutions on the horizontal line denoted by $-g(z)^*$ have the maximum profit. More detailed studies of this objective space are done during this publication.

3. Related Work

The TTP was introduced by Bonyadi [BMB13]. He postulated the necessity of research on interwoven problems and provided the mathematical formulation of the TTP. Also, the single and multi-objective version of the problem was introduced. Bonyadi laid the foundation for further research.

Afterwards, a new large-scale benchmark was proposed [PBW+14]. The benchmark is based on the TSPLIB [Rei90] instances which are available for TSP. Since the TTP also needs the KNP part, items with weight and profit attributes are added to the cities. Three different weight-value correlations and ten different capacity categories are considered. Instances with $1, 3, 5$ and 10 items per city are provided. All this together builds a benchmark set with 9,720 different instances. Furthermore, methods where π is always the near optimal tour π^* calculated by the Lin-Kernighan Heuristic [ACR03] are described. A Simple Heuristic (SH) is proposed which uses a fitness value for each item and greedily picks them. Moreover, a Random Local Search (RLS) and an (1+1) Evolutionary Algorithm (1+1 EA) is applied to the TTP, where hill climbing is performed on a knapsack which was filled up to the maximum capacity. The mutation operators are different in both algorithms and could be found in [PBW+14]. 1+1 EA showed the best results for small, RLS for medium and SH for very large problem instances.

Also, the coworker approach *CoSolver* was introduced in [BMPW14]. One component given the other one is optimized in several iterations. It was compared with another heuristic based approach called Density Heuristic (DH), which is similar to SH. *CoSolver* outperformed DH in average and showed a smaller standard deviation.

Additionally, some investigations regarding the interdependence of the TTP were made by Yi Mei et al [MLY14b]. Basically, meta-heuristic algorithms are used for the first time. On the one hand Cooperative Coevolution (CC) is applied where π and z are considered separately and are only combined for calculating the function value. On the other hand a Memetic Algorithm (MA) is proposed which is an Evolutionary Algorithm (EA) with an embedded

local search. Both algorithms are tested on a small-scale benchmark with the result that MA outperformed CC.

Thereafter, some improvements of MA are made using some complexity reduction strategies [MLY14a]. The algorithm uses a Two Stage Local Search and is called Two Stage Memetic Algorithm (TSMA). Randomly a local search is performed, where the tours of both parents are combined using a TSP crossover operator. By fixing the tour, the packing plan is created based on an item heuristic and improved through an EA. Finally, the offspring is added to the population. Experiments are made on the large scale benchmark [PBW+14] and TSMA showed better results than RLS and 1+1 EA. Since the item picking heuristic is an essential part, further improvements were made on this heuristic [MLSY15].

A similar method is described in [Wac15] and called Hybrid Genetic Algorithm. It is a genetic algorithm with an embedded local search as well. The implementation performs a crossover on either π with a local search on z, or z with a local search on π. In comparison to TSMA a local search is performed for every offspring. A detailed study of different parameters such as population size, survivor size and mutation rate is provided. Because of the frequently performed local searches this algorithm performed worse than RLS and 1+1 EA by execution time. Nonetheless, Hybrid Genetic Algorithms outperformed Pure Genetic Algorithms.

Another meta-heuristic approach, namely Ant Colony Optimization (ACO), was investigated in [Bir15]. For the TSP, ACO algorithms showed good results in the past. If this approach is also appropriate for the TTP, is part of this publication. Birkedal proposed two versions of his algorithm where ants are used to create π. The packing plan z is calculated by an improved version of DH called GDH. In the first version the length of the tour is used for the pheromone deposition and therefore the two subproblems are considered completely isolated. In the second version the normalized function value $G(\pi, z)$ provides the value for deposition. The interwovenness is considered, because GDH estimates the function value of π by solving the embedded KNP problem using a greedy heuristic. The second version performs consistently better than the first one and showed also a smaller standard deviation.

4. Characteristics of the Traveling Thief Problem

In the following chapter, characteristics of the TTP are discussed and additionally considerations about the multi-objective space are made.

4.1 Discrete Variables

The TTP provides two discrete variables. The tour is given by a permutation vector π. The permutation property must hold otherwise a constraint is violated. Therefore, methods based on evolution must use recombination operators that consider the permutation. Even though π is part of the TSP and recombination methods were studied intensively [OSH87], the interwovenness to another problem is of course not included in those studies. Moreover, a vector z defines if an item is picked or not. It is a binary vector which does not allow packing an item partially. Both discrete variables do not allow smooth continuous changes and therefore the search in the variable space is more complex than for continuous optimization problems.

4.2 Non-linear Problem

In opposition to the TSP and KNP, the TTP cannot be formulated in a linear equation. The distance to the next city is divided by the current velocity. Since the thief gets slower when an item is picked, the velocity is a variable and not a constant. The velocity is dependent on the current weight and for that both variables π and z are needed. The variables are in the denominator and it could only be linearized by using approximation techniques. For this reason, Mixed Integer Linear Programming (MILP) algorithms could not be applied to solve the original problem optimally. Even though these methods are time-consuming, optimal solutions would help to investigate characteristics of the TTP.

4.3 Expensive Reevaluations

For the subproblems many algorithms are based on fast reevaluations:

- TSP: When a TSP-Swap is performed, the new tour length can be calculated by two additions and two subtractions.
- KNP: When an item is added or removed, the resulting profit can be calculated by adding or subtracting the profit and weight, and checking the maximum capacity constraint.

Through the interwovenness both calculations are not possible any more. Even though sometimes an efficient partial reevaluation could be done, mostly it cannot. When a swap on π is performed, the weight vector changes because π depends on z. A weight change is followed by changing the traveling times between the cities. The same fact holds for a bit flip on z. This fact is a crucial difference compared to the subproblems alone.

4.4 Symmetry of π

For the TTP a symmetric map is defined. Whenever the knapsack is empty ($z = z^{empty}$), the symmetry of π holds. Let us assume we have the tour π and the symmetric tour π^{sym}.

Then

$$\begin{aligned} G(\pi, z^{empty}) &= G(\pi^{sym}, z^{empty}) \\ (f(\pi, z^{empty}), 0) &= (f(\pi^{sym}, z^{empty}), 0) \end{aligned} \tag{4.1}$$

holds. When the thief starts to pick an item, the velocity decreases from that point onwards. The order of the cities is obviously different from π to π^{sym}. For that reason the overall traveling time changes. In general

$$\exists\, z = (z_1, .., z_m) \in \mathbb{B}^m \quad f(\pi, z) \neq f(\pi^{sym}, z) \tag{4.2}$$

which means the traveling time of symmetric tours is different and therefore the objective value as well. Of course, degenerated problems where no items exist or the maximum capacity is zero are excluded. When a TSP solver is used, which optimizes basically the TTP for z^{empty}, only one tour π is returned. If this tour is fixed, the symmetric tour should be considered too, because of the symmetry on z^{empty}. Choosing randomly one of the both optimal tours could have fatal consequences for the further search. For instance, if we take in our scenario example the tour $[0, 1, 2, 3]$ instead of $[0, 3, 2, 1]$, we are only able to find two points of the real Pareto front (cf. Table 2.4).

4.5 Single versus Multi-objective Optimization Problem

Mostly Model A was considered so far, where the TTP is reduced to a single-objective problem using Equation 2.13. The rent rate R is introduced to balance time and profit. Since this is dependent on the map and values of the items, R has to be defined for each problem instance. For example, if the thief could steal high valuable items, the resulting profit $g(z)$ is also large. Furthermore, the range of the traveling times is unknown, because the distance between the cities varies. In order to define an appropriate R, at least the boundaries are helpful.

The weight value in benchmark [PBW+14] is defined as

$$R = \frac{g(z^*)}{f(\pi^*, z^*)} \tag{4.3}$$

where z^* is the optimal packing plan for the knapsack and π^* the optimal or almost optimal tour for the TSP. As an estimation for the minimum tour time of the Pareto front, the optimal tour ignoring the knapsack part is used and $f(\pi^*, z^*)$ calculated.

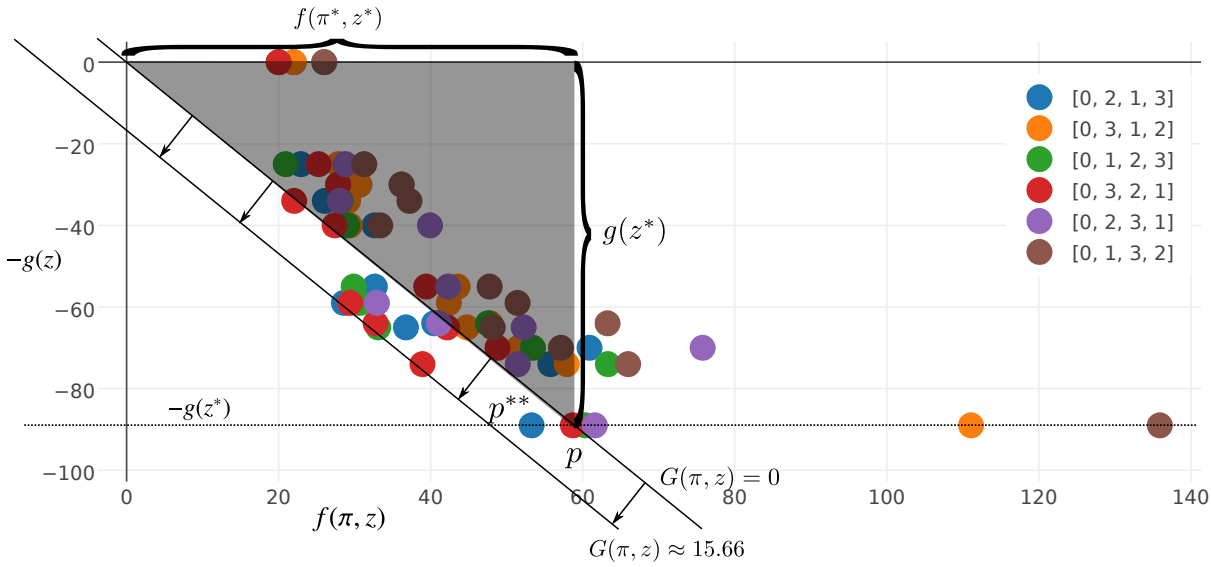


Figure 4.1: Rent Rate R for the Example Scenario

For our example scenario the best picking plan is $[1, 1, 0, 1]$ and the best tour according to this picking plan $[0, 1, 2, 3]$ or the symmetric best. It is proposed [FPSW15] to use the best tour calculated by the Lin-Kernighan Heuristic [ACR03] which returns randomly the best or the symmetric best, if we make the assumption to start at city *0*. Better results could be achieved by testing both tours for the fixed picking plan and taking the faster one. This is only necessary, because - if at least one item is picked - the symmetry is no longer guaranteed. Since the tour $[0, 3, 2, 1]$ is faster on the given packing plan, we will continue with this tour.

Using the optimal packing plan the thief needs ≈ 58.70 time units and has a profit of 89.0. Therefore the renting rate is:

$$\begin{aligned} R &= \frac{g(z^*)}{f(\pi^*, z^*)} \\ &= \frac{g([1,1,0,1])}{f([0,3,2,1],[1,1,0,1])} \\ &= \frac{89}{58.70} \approx 1.516 \end{aligned}$$

The found solution, denoted as p in Figure 4.1, has always the single-objective value $G(\pi^*_{z^*}, z^*) = 0$. In this case the solution is not part of the Pareto front, because another point p^{**} with the picking vector $[1,1,0,1]$ and the tour $[0,2,1,3]$ needs less time than the optimal tour. This method tries to find the solution with the highest profit and minimum time denoted as p^{**} in Figure 4.1.

Moving the line to the left increases and to the right decreases $G(\pi, z)$. Since we are maximizing $G(\pi, z)$, the most left point, which lies on the slope, is the optimum according to R. If the single-objective problem is considered, the algorithm might miss the knee in the Pareto front.

We used a similar equation for our proposed benchmark (c.f. Section 5.8).

$$R = \frac{g(z^*)}{f(\pi^*, z^*) - f(\pi^*, z^{empty})} \tag{4.4}$$

Here the slope takes the time decrease on π^* from z^{empty} to z^* into account. Therefore, the subproblems are balanced more precisely. But it is still based on the assumption that $f(\pi^*, z^*)$ is close to the optimal solution on the $-g(z^*)$ horizontal line.

4.6 Optimization by using the optimal Tour

Some algorithms are fixing the tour by using the optimal or near optimal solution of the TSP. But, as the example scenario shows, there are points on the Pareto front which do not have the optimal tour. Even though, the search space could be reduced by using this method, solutions which might be part of the optimal Pareto front are ignored. Since for the TSP exist symmetric solutions as well, the first question is, which of the two optimal tours to choose.

In Figure 4.2 all solutions with tours, which are represented at least once in the Pareto front, are shown. Each solution, which is part of the Pareto front, has a black circle in the middle. Regarding the defined weight vector $R = 1.516$ the blue point p^* in the Pareto front with $G(\pi, z) = 15.66$ is optimal. When the solution space is limited by looking only at solutions with the optimal tour, which means only green and red points, the solution p' with

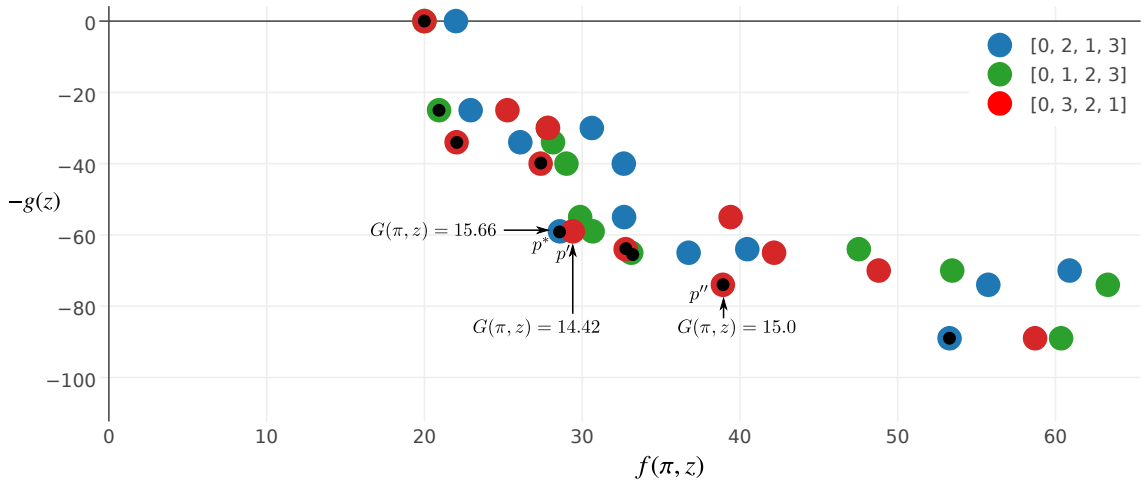


Figure 4.2: Objective Space denoted with $G(\pi, z)$

$G(\pi, z) = 14.42$ instead of p^* is found. Furthermore, there is another solution p'' where $G(\pi, z) = 15.0$ with the optimal tour. When the problem is solved exhaustively in a single-objective way by only looking at solutions with optimal tours, p'' is found. Point p'' is part of the Pareto front and a reasonable decision. But, solution p^* cannot be found at all. If the search space is reduced to variables with only optimal tours, even an exhaustive search limited by this assumption might not return solutions of the Pareto front.

4.7 Shape of the Pareto front

Most of the experiment results showed convex Pareto fronts. However, the example in Section 2.4 shows a non-convex front (cf. Figure 4.3). There is a solution p' and another solution p''. Any point in between these two points must have a value which is below the directly connected line. Obviously, the solution p''' is above that line which means the Pareto front is not convex. This also holds, if p'''' is chosen as a second point and p' kept as a first. Then all selected solutions have the same tour $[0, 3, 2, 1]$ and only the picking vectors z vary. This means, even if π is fixed and only z modified, the resulting front might be non-convex. In fact, optimizing Model A by using a rent rate to compose the problem will never find solutions in the non-convex part.

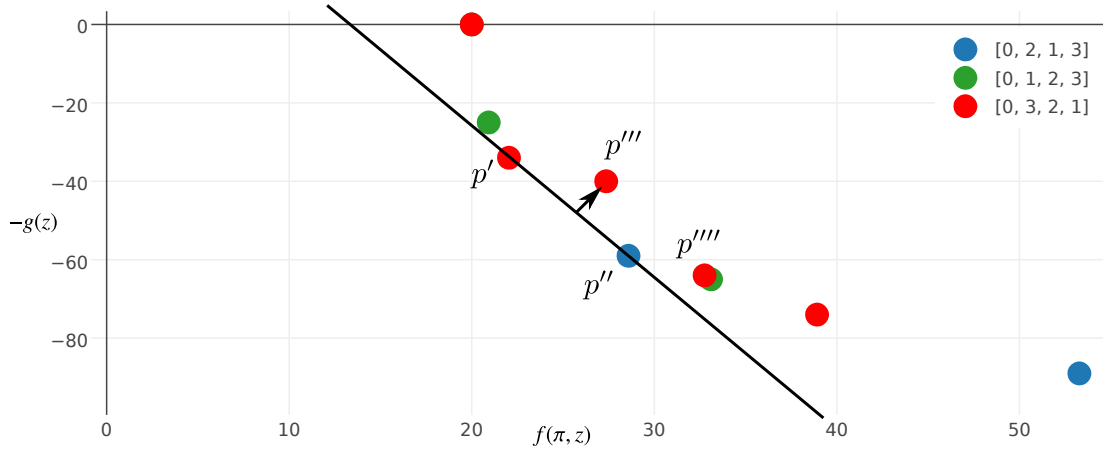


Figure 4.3: Non-Convex Pareto front

4.8 Right Shift of Tours with empty Knapsacks

If we consider only the TSP, we are optimizing the front where $g(z) = 0$. For all these solutions the knapsack is empty, which means the velocity during the tour is equal to v_{max}. If the thief starts to pick an item, he gets slower during the tour because of the velocity decrease. For this reason all tours are shifted to the right when items are picked (c.f. Figure 4.4).

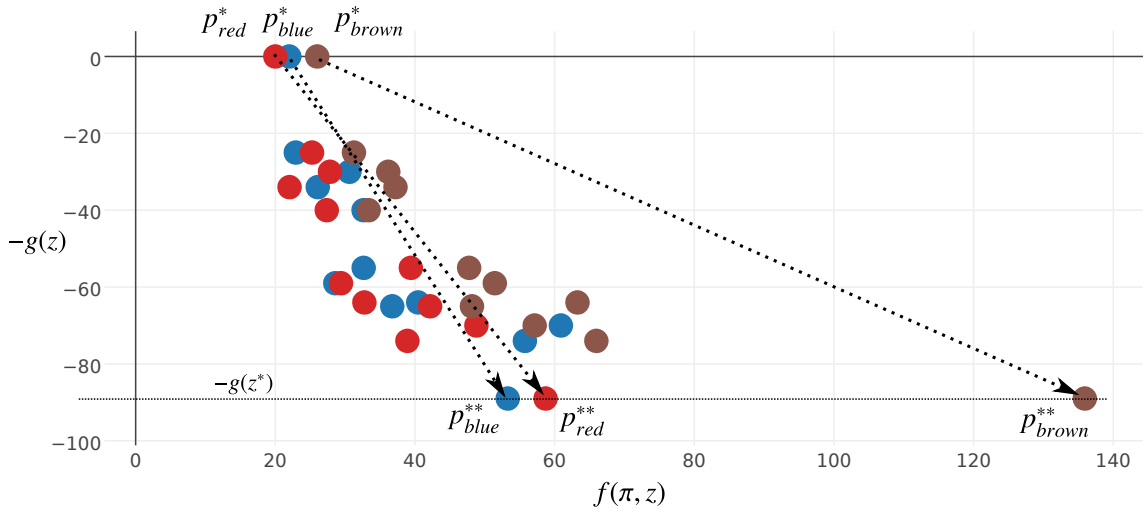


Figure 4.4: Right Shift of Solutions in the Objective Space

The solution p^*_{red} is the optimal tour without picking any item. If the tour π_{red} is fixed, the highest profit, which could be achieved using this tour, is on the horizontal $-g(z^*)$ line. This holds for any arbitrary π. However, p_{blue} starts on the right of p_{red} but has a smaller shift to the right. For this reason, a tour which is not fast without picking any item on $g(z) = 0$, could become part of the Pareto front when items are picked.

5. In-Depth Analysis of the Traveling Thief Problem

This chapter provides an in-depth analysis of the TTP. All studies are based on small-scale instances in order to get some new insights into how the interwovenness of the problem works.

5.1 Problem Solving Strategies

We propose three different strategies how to tackle the single-objective problem. The challenge is clearly to consider the interwovenness without changing the original optimization goal.

5.1.1 Combined

First, the problem can be seen as a combined problem where π and z and therefore also the TSP and KNP have the same importance. In general, this allows both variables to be changed at the same time.

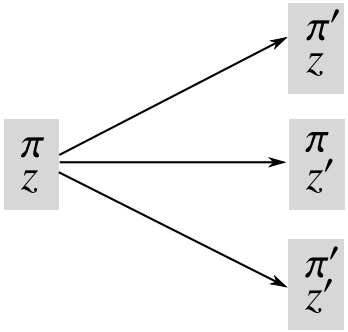


Figure 5.1: Problem Solving Strategy: Combined Approach

There are three different possible changes of both variables (c.f. Figure 5.1). Even though it sounds very intuitive, this is the major difference from the other models. It allows π and z to change to π' and z'. This approach can be implemented in various ways, for example in a point search or an evolutionary algorithm.

5.1.2 Bi-level

Another possibility is to create a hierarchy among the variables and look at the TTP as a bi-level problem. A bi-level problem divides all variables into two classes, namely the lower-level and the upper-level variables, where each level could have constraints [CMS07]. The idea is to fix all upper-level variables and to optimize the lower level. Applied to the TTP two different hierarchies are possible.

Firstly, the TSP can be in the upper level and the KNP in the lower level. For different π, z is maximized by not violating the knapsack capacity Q.

$$\begin{aligned} \max_{\pi} \quad & G(\pi, z') \\ \text{s.t.} \quad & z' = \arg\max_{z} \; G(\pi, z) \\ & \qquad \text{s.t.} \quad W(z) \leq Q \end{aligned} \tag{5.1}$$

Secondly, the KNP can be in the upper level and the TSP in the lower level. Since we are fixing z before, the capacity constraint is part of the upper level. According to z, the optimal π has to be found.

$$\begin{aligned} \max_{z} \quad & G(\pi', z) \\ \text{s.t.} \quad & W(z) \leq Q \\ & \pi' = \arg\max_{\pi} G(\pi, z) \end{aligned} \tag{5.2}$$

In both models the lower level is solved many times and therefore an efficient method should be used. Since there is a NP-hard problem in the upper and one in the lower level, both models are equally challenging.

5.1.3 Bank-to-Bank

Because each hierarchy is used alternatingly, Bank-to-Bank works without having the same hierarchy for the whole optimization process. Always one of both variables is fixed and the other one is optimized (c.f. Figure 5.2).

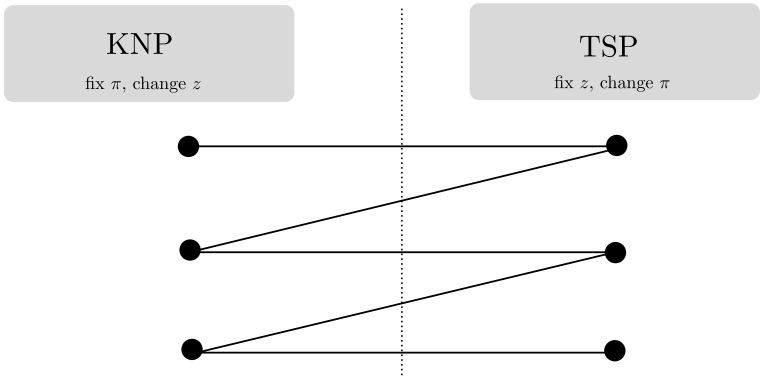


Figure 5.2: Problem Solving Strategy: Bank-to-Bank

Either π is fixed and only the packing plan z, or z is fixed and only π is modified. However, in opposition to the combined approach either π or z is modified, but not both at the same time.

5.1.4 Classification of algorithms

All three approaches are handling the problem in a certain way. Therefore, we classify the algorithms of the related work into the proposed categories:

Strategy	Algorithm
Combined	MA [MLY14b]
	TSMA [MLY14a]
Bi-level	RLS [PBW+14]
	SH [PBW+14] / DH [BMPW14]
	1+1 EA [PBW+14]
	ACO [Wac15]
Bank-To-Bank	CoSolver [BMPW14]
	CC [MLY14b]

Table 5.1: Problem Solving Strategies: Classification of existing Algorithms

Sometimes the assignment is controversial, because another concept is used as well. For example TSMA clearly allows both variables to be changed. When a crossover is performed, π and z from the offspring could be different from both parents. But the local search, which is executed during the crossover with a specific probability, is similar to a bi-level strategy, because π is fixed and a heuristic search for z is done. However, we assigned the category according to the overall concept.

5.2 Are Traveling Salesman Problem and Knapsack Problem algorithms useful?

In the past decades a lot of research was done on the TSP and KNP. Undoubtedly, algorithms and knowledge about the subproblems have to be reviewed. Therefore, degenerated cases of the TTP, where one of the two subproblems is clearly dominating, should be investigated. Two different viewpoints for the degenerated cases are possible depending on which model is considered:

- Single-Objective Model A: The rent rate R balances the influence of both subproblems.
- Multi-Objective Model A: Specific points in the objective space could be calculated using algorithms for the subproblems.

On the one hand, if the rent rate $R \approx \infty$, the slope in the bi-objective space is almost a vertical line (c.f. Figure 5.3). The goal is to find solutions on the left most vertical line. Since $R \approx \infty$, the profit is negligible and only the time $f(\pi, z)$ is important. A solution where $\min f(\pi, z)$ has to be found, which is nothing else than solving only the TSP. Basically, when $R \approx \infty$ then the dependency on z is not given any more and the function could be simplified to $f(\pi) = f(\pi, z^{empty})$.

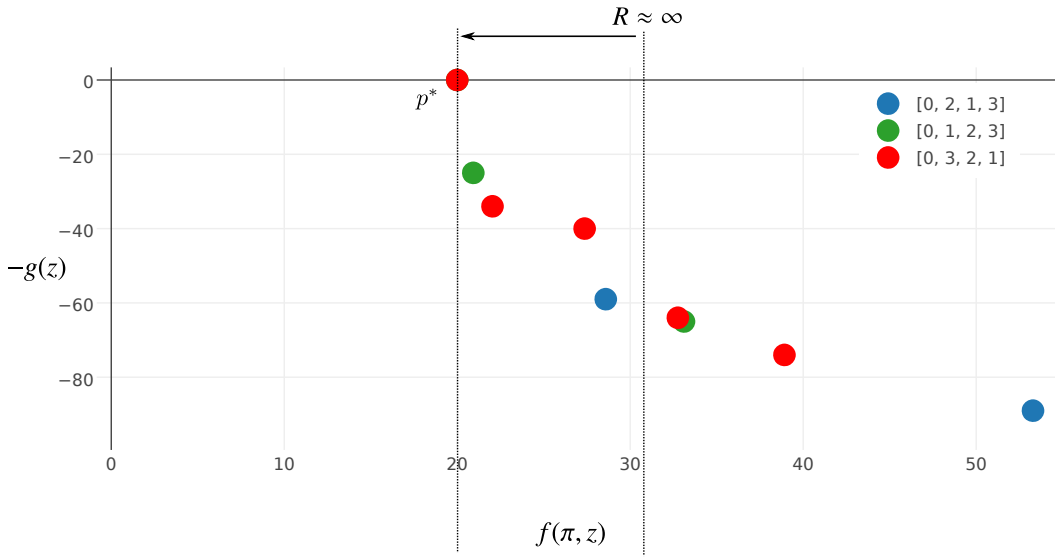


Figure 5.3: Traveling Thief Problem degenerated to Traveling Salesman Problem

On the other hand, we are able to find the left corner point p^* in the objective space. If we consider Model B for the TTP, this point cannot be dominated by any other point. All exact solvers for the TSP provide at least one solution of the optimal Pareto front for the TTP.

Another degenerated case is given for $R = 0$ where the traveling time of the thief has no influence on the function value. Solving the problem using an existing exact KNP solver, for example [MPT99], will also provide the optimal solution for the TTP. Solutions on the horizontal line have the same single-objective value (c.f. Figure 5.4). By solving only KNP, the variable z^* is provided which defines a solution on the lowest horizontal line. Interestingly, here the corner point p^{**} could not be calculated easily, as it was possible for the former corner point. All solutions with z^* and any arbitrary π are on the $-g(z^*)$ line. But for calculating p^{**}, we are searching for the fastest tour according to z^*. Because this problem involves distances between cities with conditions, it is not possible to put in a distance matrix.

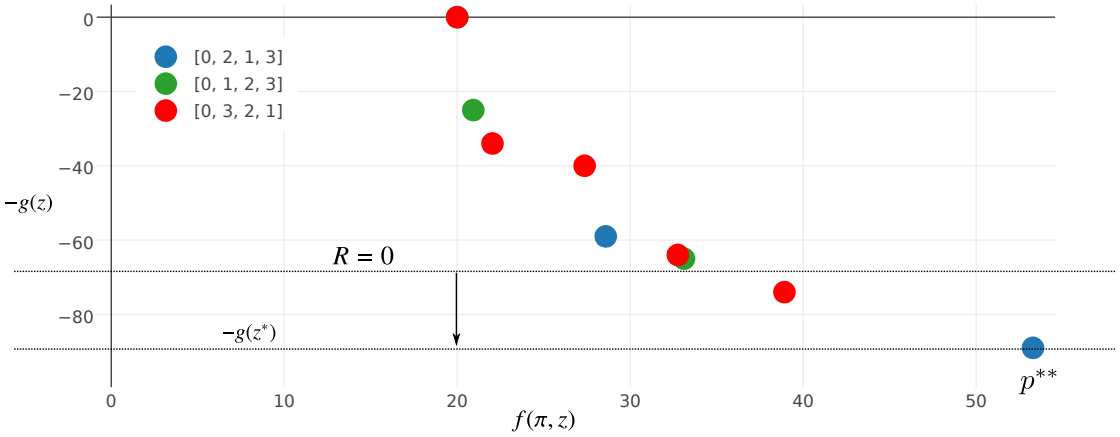


Figure 5.4: Traveling Thief Problem degenerated to Knapsack Problem

Let us for the purpose of illustration consider the following variables based on the example scenario in Section 2.4.3. The thief picks items according to *[0,0,0,1]* which means item 3 is picked. When $\pi_1 = [0, 1, 2, 3]$ and $\pi_2 = [0, 3, 1, 2]$ the traveling times between city 1 and city 2 are different. For π_1 the thief did not pick any item and goes with the maximum velocity. Since the distance between city 1 and 2 is 5, he needs 5 time units. For π_2 the thief picked item 3 at city 3. Therefore he has $w = 21$ and a velocity $v = 1.0 - \frac{1.0-0.1}{80} \cdot 21 = 1 - 0.23625 = 0.76375$ which is less than v_{max}. He needs $\frac{5}{0.76375} = 6.547$ time units for the same distance. In fact, only π and not z changed for both calculations, but the time units between the same cities are different. If $z \neq z^{empty}$ no time matrix exists and TSP solvers could not be used.

Nonetheless, the range of the feasible objective space is partially provided by methods for both subproblems. TSP solvers calculate the fastest tour and provide therefore the lower bound of the traveling time. Note, if $v_{max} \neq 1.0$, it has to be divided by v_{max}. The upper bound could only be found if we solve the longest path problem as well and divide the tour length by v_{min}. Moreover, the lower bound of the profit is known to be zero and the upper bound could

be calculated by knapsack solvers. This holds also when depreciation is applied. However, a fast tour and a good packing plan is an excellent starting point for the TTP as well.

Furthermore, the degenerated cases also show that TTP must be NP-hard. Since TSP is known to be NP-hard and we are able to reduce TTP to TSP, TTP must be NP-hard as well. The same holds for the KNP.

5.3 Is it beneficial to pick items as late as possible?

The intuition is that items should be picked as late as possible to slow down the thief as little as possible. Several experiments were made to compare algorithms and this fact was assumed. However, no concrete experiment confirms that assumption so far.

Therefore, we used an example problem instance with 100 cities and calculated random tours. We applied an evolutionary algorithm only on the packing plan for 100 generations with a population size of 200. This ensures to find solutions which are better than random ones, but of course the optimum is not guaranteed. We analyzed only the best found solution. The relative times and weights on the tour on 30 different random π's were calculated. The results are shown in Figure 5.5.

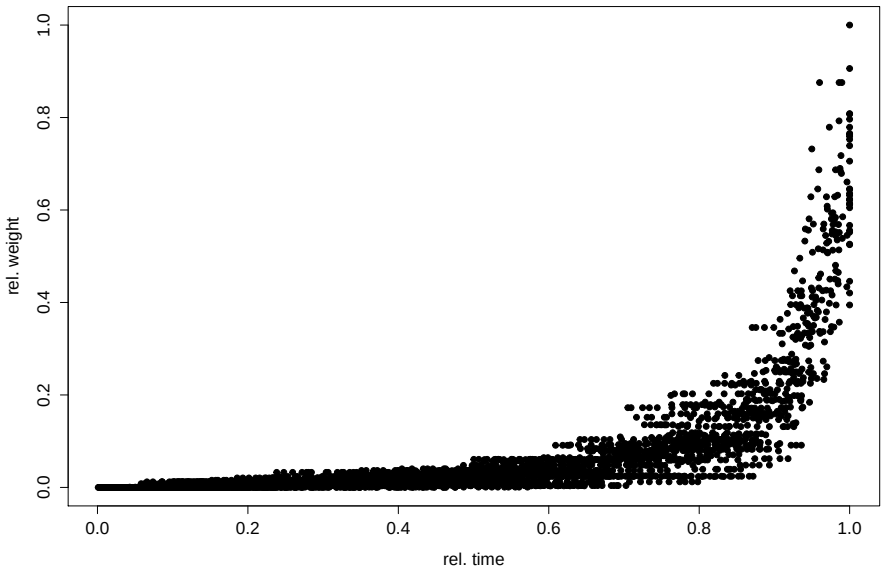


Figure 5.5: Weight of the Knapsack during a Tour

Clearly, the weights are increasing slowly in the beginning, but fast in the end. That avoids picking heavy items too early and having therefore a velocity decrease. However, also in the beginning items are picked, but they are light compared to items in the end.

Furthermore, we did the same experiment with depreciation. Therefore, we took the same problem instance, but added an exponential depreciation to all items. The depreciation rate

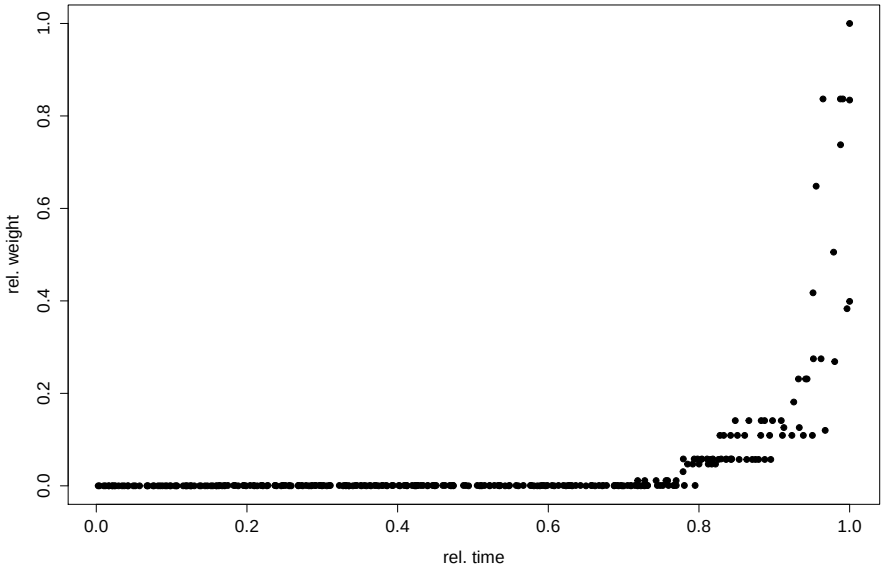


Figure 5.6: Weight of the Knapsack during a Tour when Depreciation is considered

was $b = 0.95$ and the constant $C = 1000$. As Figure 5.6 shows, the effect is strengthened. Almost no items are picked in the beginning but only in the end. Nonetheless, it is difficult to define the depreciation variables. On the one hand a continuous movement to the result without depreciation was observed by changing the variables to less depreciation. On the other hand if the depreciation effect was too strong, no items were picked and a degenerated problem instance was made.

The problem is to define π to allow a packing scheme, which let the thief pick light items in the beginning and the heavy items in the end. Additionally, the tour should not have a long detour compared to π^*.

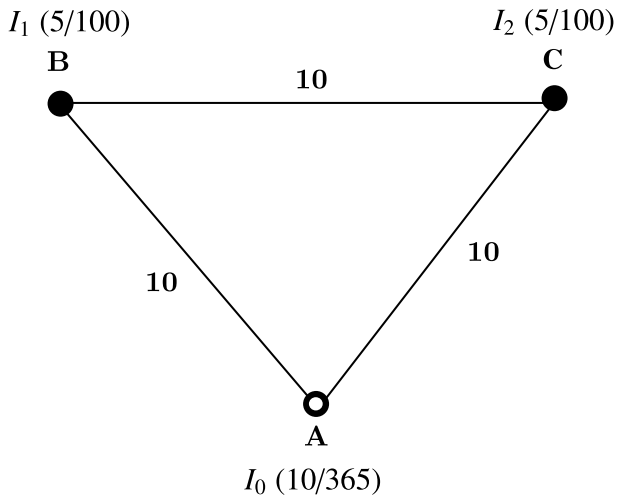
5.4 Why is greedy too greedy?

Our experiments showed that evolutionary algorithms outperform all greedy approaches on small-scale examples. For this reason we analyzed the inaccuracy, when items are picked greedily. We found an interesting effect which we present in the following example scenario.

For large-scale examples the greedy algorithm usually sorts the items according to a heuristic. Then the items are picked until the next item does not improve the objective value $G(\pi, z)$. We can spend a lot more function evaluations for each item in small-scale examples. For this reason, our algorithm considers all items, which the knapsack does not contain in the current situation, and packs the next item with the highest $G(\pi, z)$ improvement. This means in the beginning all m items, then $m - 1$, $m - 2$ and so on are considered, until no improvement is made. The complexity of the algorithm is $O(m) = m^2$ and π needs to be fixed beforehand.

Even though this greedy approach is more precise in selecting the next item, it did not find the optimum very often.

For this reason let us consider another example. It has a symmetric map with only two different possible tours (c.f. Figure 5.7).



Notation: I_i $(weight/value)$
Maximal Capacity: 10
R = 1.0

Figure 5.7: Example for Greedy Algorithms

If all items are ranked, item I_0 shows the best heuristic value $H(0)$. The profit-weight rate $H(i) = \frac{b_i}{w_i}$ is much better than for the other items:

$$\begin{aligned} H(1) = H(2) &< H(0) \\ \frac{100}{5} &< \frac{365}{10} \\ 20 &< 36.5 \end{aligned}$$

All feasible solutions are listed in Table 5.2. Since this example is symmetric, only one possible tour needs to be considered. The thief starts with an empty knapsack $G([A, B, C], []) = -30$ and has to choose between $G(\pi, [0]) = 65.0$, $G(\pi, [1]) = 53.64$ or $G(\pi, [2]) = 61.82$. It does not matter which next item is packed, the objective value is increased. However, the highest improvement is reached by picking item I_0 in this case. If this decision is made, the knapsack is completely full and the algorithm terminates immediately.

But the best solution is $G([A, B, C], [1, 2]) = 71.82$ and it does not contain the most promising item I_0. Even though this is more or less a constructed example, we have seen the same effect

π	z	$G(\pi, z)$
$[A, B, C]$	[]	-30.0
$[A, B, C]$	[0]	65.0
$[A, B, C]$	[1]	53.64
$[A, B, C]$	[2]	61.82
$[A, B, C]$	[1, 2]	**71.82**
$[A, C, B]$	[]	-30.0
$[A, C, B]$	[0]	65.0
$[A, C, B]$	[1]	61.82
$[A, C, B]$	[2]	53.64
$[A, C, B]$	[1, 2]	**71.82**

Table 5.2: All feasible Solutions of the Greedy Example

on larger problems, where the first greedily picked item is worse than other solutions without it.

Decisions which are made too early might be disadvantageous and the algorithm should be able to undo them. Since greedy algorithms make only irreversible decisions, they might stuck in local optima. Additionally, there are greedily found solutions which could be improved by removing items in the end. Therefore a post-pruning of the item set is useful.

Moreover, we made further studies based on a divide and conquer algorithm that divides all items and combines them to promising subsets. But on every level and for every merge, items are excluded for the further search and never considered again. Even though the complexity of the algorithm is logarithmic to the number of items, the results were not promising enough to study further.

Irreversible decisions are helpful to exclude parts of the search space. When those decisions are based on heuristics, this procedure could be disadvantageous for the TTP if the true optimum should be found. This effect is also attributable to the interwovenness of the TTP.

5.5 Case Study

It is a challenge to visualize large-scale instances for the TTP. Therefore, we took the *berlin52_n51_bounded-strongly-corr_01* from [PBW+14] in order to look at different solutions found by algorithms. The problem instance contains 52 vertices, which represent locations in Berlin. Every vertex - except the first - provides an item to pick. The weights and values of the items are correlated to each other and the knapsack capacity is about 9% of the sum of all items. This section is not a comparison of algorithms, but illustrates how results look like and allows drawing conclusions from them.

We used the same evolutionary algorithm as described in Section 5.3 to find a good result on the optimal π^* and symmetric optimal π^{**} TSP tours. The result is shown in Figure 5.8. The thief starts at the red vertex. When the vertex and edge is black, an item is stolen at the city, if

it is grey, not. All 2-opt optimal tours and also the optimal tour will not have any intersection between two edges.

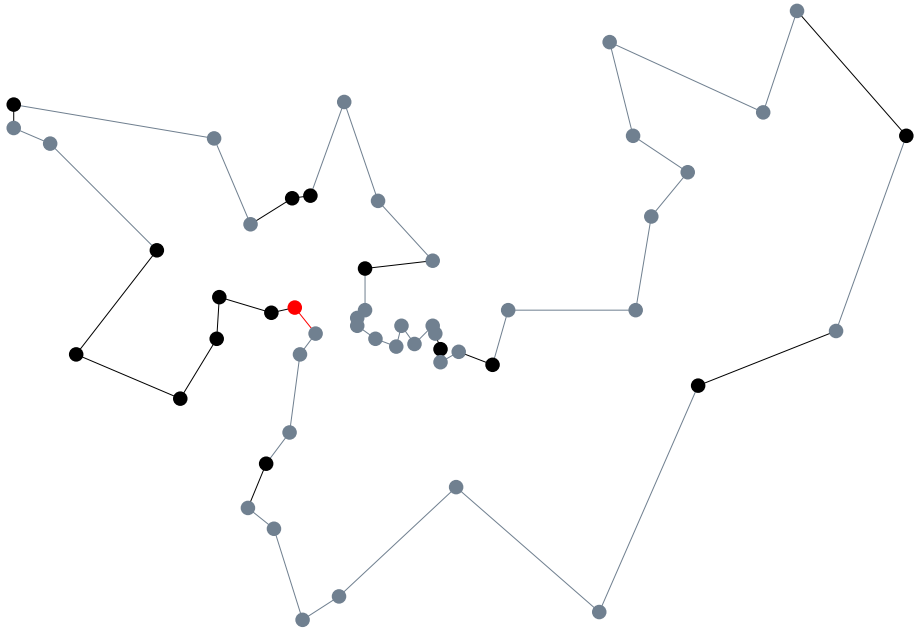


Figure 5.8: Berlin52: Optimized on π^* with $G(\pi, z) = 4016.77$

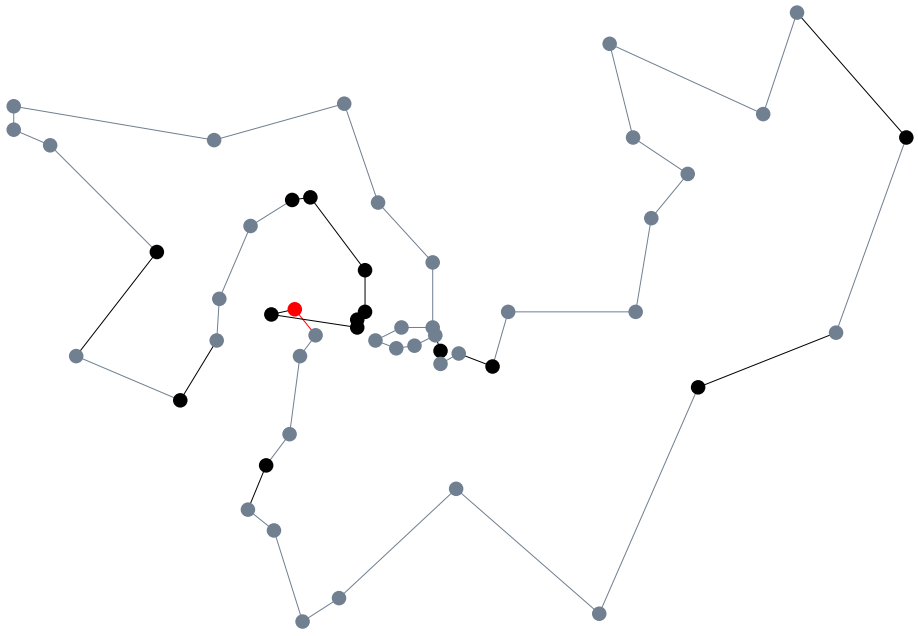


Figure 5.9: Berlin52: Solution found by TSMA with $G(\pi, z) = 4265.25$

Additionally, we used TSMA from Yi Mei et al [MLY14a] in order to see an effect when π is changed. The function value could be improved by choosing another tour π as shown in Figure 5.9. The found tour is not 2-opt optimal since there is an intersection at the end of the

tour close to the starting city. However, the beginning of the tour is similar to π^*, the end is different.

Furthermore, we implemented a bi-level approach. A sliding window swapping method is applied in the upper level on π and hill climbing on z in the lower level. We found another solution with $G(\pi, z) = 4399.59$ (c.f. Figure 5.10) using this algorithm.

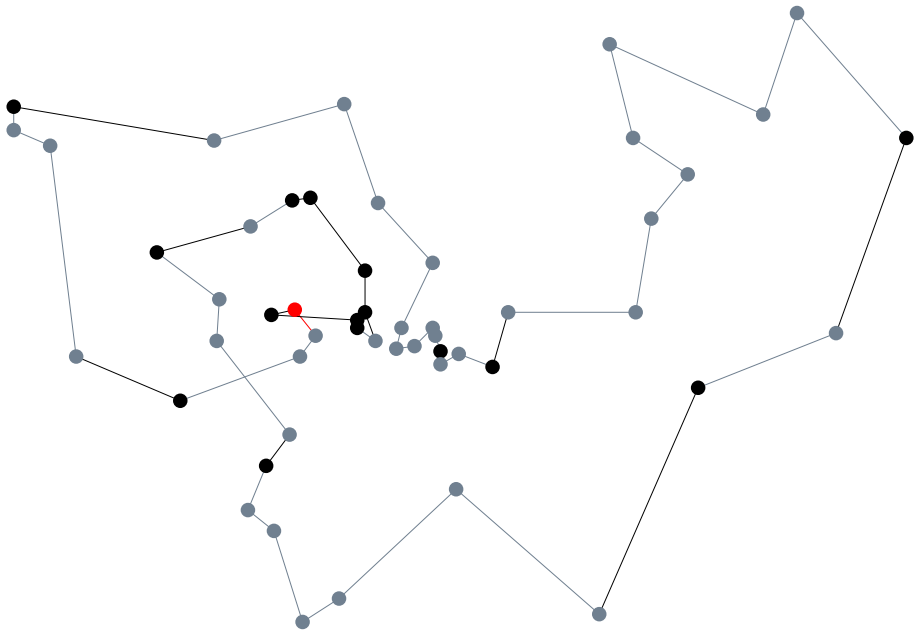


Figure 5.10: Berlin52: Best found Solution with $G(\pi, z) = 4399.59$

The result shows a tour with different directions and orders of the cities as before. The thief starts completely different and goes first to the east instead of to the west. Therefore, the end of the tour is different as well.

Even though all three figures look quite similar, they are very different in terms of the permutation vector representations. How to handle this issue, should be part of further studies. Moreover, we noticed the question is how to adapt the optimal tour π to the item distribution. All π's, which were created by the nearest neighbor approach, did not show good results. Therefore, the TSP solver could be used to create a fast tour, but the tour needs to be adapted in order to collect the items in a good order.

5.6 Clusters of Cities

In order to have other real-world problem instances, clustered city scenarios should be investigated, as it was done for the TSP [Hel14]. Let us consider the following example with three different clusters (c.f. Figure 5.11). The fastest tour brings the clusters in an order and when the distance between the clusters is not negligible, each cluster will be visited only once. Intuitively either *[C_1, C_2, C_3]* or *[C_3, C_2, C_1]* is a good order to travel through the clusters, but these solutions do not consider the assignment of the items. When all promising items are

in C_2, both permutations are disadvantageous, because C_2 should be the last cluster to visit, in order to keep the velocity high until the end. If the permutation vector representation is used, clusters are represented by many cities. It is very unlikely to change the cluster order by considering the vector city wise. Maybe cluster algorithms have to be applied. Our benchmark contains problem instances with 3, 5 and 10 clusters to study the results of existing approaches or to apply new algorithms.

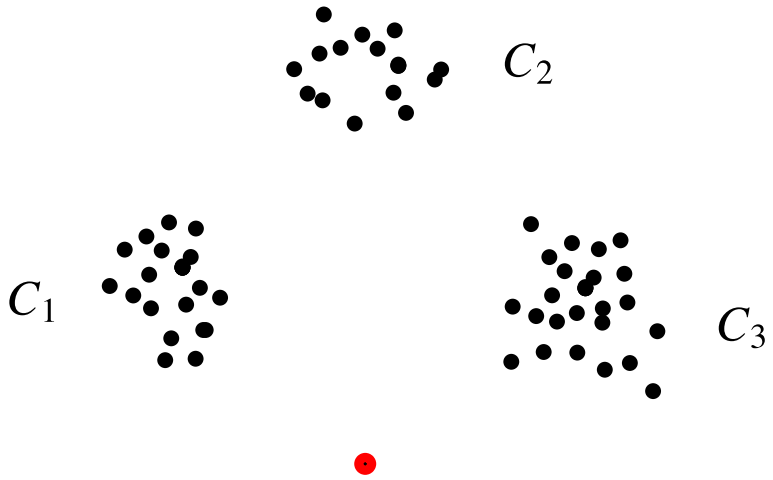


Figure 5.11: Another Example where Cities are arranged in Clusters

5.7 Mutation on π: Simple-Swap versus TSP-Swap

Different recombination operators used by evolutionary algorithms were investigated. Swapping two cities is the smallest change which is possible on π. But there are two different types of swaps:

- Simple-Swap: Swap two entries in an array as it is done for sorting algorithms.
- TSP-Swap: Reverse the array in between the two swapping indices and insert it at the same position.

First, let us discuss the influence on the objective values. The example in Figure 5.12 shows both swaps. The Simple-Swap introduces four different edges, (3,7), (7,5), (6,4) and (4,8) and removes (3,4), (4,5), (5,6) and (6,7). All items, which were picked at city 4, are picked later, and items at city 7 earlier.

The TSP-Swap is applied in order to optimize the tour 2-opt for the TSP. The edges (3,7), (4,8) are added and (3,4), (7,8) are removed. In fact, fewer edges are affected. But the order of cities is reversed, and therefore the items of the cities 4,5,6,7 are picked at a different point of time. For this reason, there is another influence on profit and time.

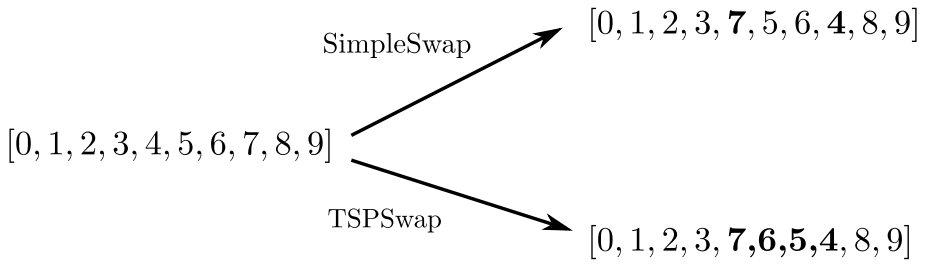


Figure 5.12: Recombination Operator: Simple-Swap versus TSP-Swap

The question is what swap improves the function value $G(\pi, z)$ most? As an essential operation, which is used in many algorithms such as hill climbing or evolutionary algorithms, it should be chosen appropriate.

Table 5.3 shows the result on the same problem as used in our case study (c.f. Section 5.5). For 100 different π's the packing plan was fixed to z^*. Then 100,000 mutations were applied to π and the relative difference to the parent was calculated.

The average improvement of the TSP-Swap is much higher than of the Simple-Swap, but the standard deviation as well. So the Simple-Swap seems to have less influence on the example problem instance. However, the maximal improvement is found with TSP-Swap and therefore this operator might be more promising for algorithms. Another advantage of the TSP-Swap is that there is a transition to the symmetric tour when the swap is performed from the second to the last index.

Operator	Simple-Swap	TSP-Swap
Mean	0.00403	**0.0164**
SD	**0.0992**	0.195
Min	**-0.580**	-0.875
Max	1.203	**3.570**

Table 5.3: Berlin52: Relative Improvement of $G(swap(\pi), z^*)$ compared to $G(\pi, z^*)$

This experiment should provide an idea of the influence and raise awareness of these two different operators. Further insights will be given when these operators are used in algorithms.

5.8 New Benchmark

We propose a new TTP benchmark in order to make efficient research on small-scale examples possible. Small problem instances are easier to visualize and therefore to analyze. In sum our benchmark contains 45 different problems. This quite small benchmark allows executing algorithms on all instances and should standardize the chosen problems for further publications. Since the other benchmark contains 9,720 instances, some publications considered different problems and so the results are not directly comparable.

Our parameter setup looks like the following:

- Cities: 5, 10, 20, 50, 100
- Items per city: 1, 5, 10
- Maximum capacity: small, medium, large

Both, the city positions and the correlation of items, are completely random. We offer our benchmark in the TSPLIB format, as proposed in [PBW+14], and also in the JSON format.

Problems for all proposed models exist and a benchmark for Model C is introduced hereby. We hope to establish a collection of the best results found so far, in order to create a database that could be used for further experiments.

6. Further Considerations

Below, further considerations are presented. Experiments on small-scale examples, which directly tackle a research question, are useful in order to make conclusions.

6.1 Representation of π and z

The most popular representation of a tour π is a permutation vector, and the 0-1 knapsack packing plan is mostly defined by a binary vector. So far, the TTP research has stuck to those two representations because algorithms and knowledge of the subproblems could be used. However, tours which look in a graph quite similar have completely different permutation vectors. As described in Section 4.4, tours are only symmetric when no item is picked. Mostly this is not the case, which means π and the symmetric tour π^{sym} are not mapped to the same objective value.

Let us consider the optimal tour $\pi^* = [x_0, x_1, x_2, x_3, x_4]$ and the symmetric optimal tour $\pi^{**} = [x_0, x_4, x_3, x_2, x_1]$. For TSP algorithms both tours show the same result. For the TTP it is beneficial to pick heavy items as late as possible. But, if the best and also heaviest item is assigned to x_4, the tour π^* is more promising than π^{**}. However, the permutation vectors are completely different and only the starting city is the same. The tours π^* and π^{**} share exactly the same edges, but the directions are different. Other graph representations, such as adjacent matrices, were investigated for the TSP and might be useful for the TTP as well.

Also, z is often treated independently of π, but it is in reality highly dependent on it. Therefore, a combination of both variables into one representation might face the interwovenness problem. If the decoding type changes, new crossover and mutation operators have to be developed.

6.2 What does change with Depreciation?

For most multi-objective studies in this publication the depreciation was not considered. But Model B uses the profit function $g(\pi, z)$ which means π has an effect on the resulting profit. With depreciation the objective space does not have horizontal alignments, because the final profit $g(\pi, z)$ is different for every π. Model A uses $g(z)$ which does not have π as a parameter and therefore the final profit for all z's does not depend on π. This dependency complicates the influence of the subproblems and increases the interwovenness. The question is, how does the influence affect the efficiency of existing algorithms and how could this effect be considered?

6.3 Influence of Problem Parameters

The benchmark proposed in [PBW+14] and our benchmark considers different parameter setups. For each TSPLIB problem the following parameters are set:

- n number of cities
- m number of items assigned to a city
- *corr* correlation between weight and value of each item
- Q maximum capacity of the knapsack
- v_{min} and v_{max}

Knowledge about the influence of these parameters would help to create further problem instances. For example it is known that a high correlation between weight and value for items does create more difficult KNP problems. But if they are uncorrelated, some items will clearly dominate others and therefore some cities are more important for the thief. This has an influence on π again. So which correlation type does create more difficult instances for the TTP?

When the maximum knapsack capacity Q is increased, each item has less influence on the final time of the thief, because the velocity decrease is dependent on Q. What does that mean for the final result? The velocities are for the whole benchmark $v_{min} = 0.1$ and $v_{max} = 1$ which means in the worst case the time between two cities becomes 10 times the distance. This is clearly one important parameter for the interwovenness and limits the effect of the KNP on the TTP. But how does the velocity range affect the problem? What happens if $v_{max} = 100$? Some small-scale experiments would help to get some insights concerning these research questions and the solutions should be analyzed.

6.4 Local Optima

Another interesting question is the existence of local optima for the TTP. Some of our experiments showed that evolutionary algorithms got stuck and no further generation was able to improve the best individual. However, we knew a better solution exists. Some studies on their existence might raise awareness for new algorithms. A distance measure in the search space should be defined, in order to measure that two solutions are close in the objective, but not in the search space. This means studies have to be made on the distance between two permutation and two binary vectors. A simple addition of those two distances might not be precisely enough, because the interwovenness affects both variables.

An example problem shall be small enough to be solved exhaustively, but large enough to be a realistic TTP example. Then the best solutions are compared according to the search space distance. If the distance is large between these solutions, we could assume local optima are a problem. Another possibility is to apply a hill climbing algorithm from many random points in the search space and have a look at the results. But the hill climbing must be done on π and z. Thereafter, the search space distance of all found solutions is measured as it would be done for the exhaustive result.

However, studies on local optima are useful in order to figure out which type of algorithm is more promising.

7. Conclusion

After introducing the TTP, showing the interaction of its subproblems, and presenting their interdependencies, different models of the problem are explained. Each model is defined mathematically and the characteristics of the objective space are discussed. In this work, conclusions about the bi-objective space of the TTP are made and the correlation between the single and multi-objective problem is explained. Furthermore, as a transition from the single to the more complex multi-objective problem, another model, namely Model C, is proposed. It has only one objective, but one extra constraint. To encourage research to be done on this problem, we publish additionally a benchmark with small-scale instances. Moreover, three different problem solving strategies are proposed: first, the combined approach, where both variables can change at any time; second, the bi-level method, where one variable is fixed and the other is optimized; and third, the bank-to-bank approach, where alternately either π or z is optimized. Also, some experimental studies and visualizations show different facets of the problem and answer some current research questions.

The TTP combines two NP-hard problems into one new problem. Clearly, the complexity is not just the sum of these two subproblems. It is more, because the interwovenness does not allow applying state-of-the-art algorithms for the subproblems. In this case one plus one is more than two and further studies should be done on the overflow part. Research done on the TTP also contributes to all interwoven problems which might be created. The interwovenness needs to be investigated in order to solve more real-world problems where several problems interact together. In the future, these problems should be investigated in more detail and algorithms which provide a meta-heuristic on how to handle the interaction of interdependent components should be proposed.

Bibliography

[ABCC07] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007. (cited on Page 6)

[ACR03] David Applegate, William Cook, and Andre Rohe. Chained lin-kernighan for large traveling salesman problems. *INFORMS J. on Computing*, 15(1):82–92, January 2003. (cited on Page 1, 17, and 21)

[Bir15] Rasmus Birkedal. Design, implementation and comparison of randomized search heuristics for the traveling thief problem. Master's thesis, Technical University of Denmark, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, DK-2800 Kgs. Lyngby, Denmark, compute@compute.dtu.dk, 2015. (cited on Page 18)

[BMB13] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *IEEE Congress on Evolutionary Computation*, pages 1037–1044. IEEE, 2013. (cited on Page 1, 9, 10, and 17)

[BMPW14] Mohammad Reza Bonyadi, Zbigniew Michalewicz, Michal Roman Przybylek, and Adam Wierzbicki. Socially inspired algorithms for the travelling thief problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 421–428, New York, NY, USA, 2014. ACM. (cited on Page 17 and 27)

[CLV06] Carlos A. Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., 2006. (cited on Page 4 and 5)

[CMS07] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization, 2007. (cited on Page 26)

[Deb01] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001. (cited on Page vi, 4, 5, and 13)

[FPSW15] Hayden Faulkner, Sergey Polyakovskiy, Tom Schultz, and Markus Wagner. Approximate approaches to the traveling thief problem. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 385–392, New York, NY, USA, 2015. ACM. (cited on Page 21)

[Hel14] Keld Helsgaun. Solving the clustered traveling salesman problem using the lin-kernighan-helsgaun algorithm. Technical report, Roskilde University, 2014. (cited on Page 35)

[IKM+15] Hisao Ishibushi, Kathrin Klamroth, Sanaz Mostaghim, Boris Naujoks, Silvia Poles, Robin Purshouse, Günter Rudolph, Stefan Ruzika, Serpil Sayin, Margaret M. Wiecek, and Xin Yao. *Multiobjective Optimization for Interwoven Systems*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. (cited on Page 7)

[Lag96] Michail G. Lagoudakis. The 0-1 knapsack problem – an introductory survey, 1996. (cited on Page 7)

[MLSY15] Yi Mei, Xiaodong Li, Flora Salim, and Xin Yao. Heuristic evolution with genetic programming for traveling thief problem. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 2753–2760, 2015. (cited on Page 18)

[MLY14a] Yi Mei, Xiaodong Li, and Xin Yao. Improving efficiency of heuristics for the large scale traveling thief problem. In *Simulated Evolution and Learning - 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings*, pages 631–643, 2014. (cited on Page 18, 27, and 34)

[MLY14b] Yi Mei, Xiaodong Li, and Xin Yao. On investigation of interdependence between sub-problems of the travelling thief problem. *Soft Computing*, pages 1–16, 2014. (cited on Page 17 and 27)

[MPT99] Silvano Martello, David Pisinger, and Paolo Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Manage. Sci.*, 45(3):414–424, 1999. (cited on Page 13 and 29)

[OSH87] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 224–230, Mahwah, NJ, USA, 1987. L. E. Associates, Inc. (cited on Page 19)

[PBW+14] Sergey Polyakovskiy, Mohammad Reza Bonyadi, Markus Wagner, Zbigniew Michalewicz, and Frank Neumann. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 477–484, New York, NY, USA, 2014. ACM. (cited on Page 17, 18, 21, 27, 33, 38, and 40)

[Rei90] G. Reinelt. TSPLIB - A t.s.p. library. Technical Report 250, Universität Augsburg, Institut für Mathematik, Augsburg, 1990. (cited on Page 17)

[TV01] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. (cited on Page 1)

[Wac15] Christoph Wachter. Solving the travelling thief problem with an evolutionary algorithm. Master's thesis, TU Wien, 2015. (cited on Page 18 and 27)

I hereby declare that this thesis was entirely my own work and that any additional sources have been duly cited.

Haibach, April 25st, 2016