Fast Heuristics for the Multiple Traveling Thieves Problem

Shelvin Chand
School of Engineering & Information Technology
University of New South Wales
Canberra, Australia
shelvin.chand@student.adfa.edu.au

Markus Wagner
School of Computer Science
University of Adelaide
Adelaide, Australia
markus.wagner@adelaide.edu.au

ABSTRACT

The traveling thief problem (TTP) is fast gaining attention for being a challenging combinatorial optimization problem. A number of algorithms have been proposed for solving this problem in the recent past. Despite being a challenging problem, it is often argued if TTP is realistic enough because of its formulation, which only allows a single thief to travel across hundreds or thousands of cities to collect (steal) items. In addition, the thief is required to visit all cities, regardless of whether an item is stolen there or not. In this paper we discuss the shortcomings of the current formulation and present a relaxed version of the problem which allows multiple thieves to travel across different cities with the aim of maximizing the group's collective profit. A number of fast heuristics for solving the newly proposed multiple traveling thieves problem (MTTP) are also proposed and evaluated.

CCS Concepts

•**Computing methodologies** → **Heuristic function construction; Randomized search;**

Keywords

Traveling Thief Problem; Traveling Salesman Problem; Knapsack Problem; Combinatorial Optimization

1. INTRODUCTION

The traveling thief problem (TTP) [3] is fast gaining attention for being a challenging combinatorial optimization problem. The NP-Hard optimization problem combines two well-known combinatorial optimization problems, namely the traveling salesman problem (TSP) and the knapsack problem. The two components have been merged in such a way that the optimal solution for both does not necessarily correspond to an optimal TTP solution. The motivation for the TTP is to have a problem where the interactions of problem components can be investigated systematically.

GECCO '16, July 20-24, 2016, Denver, CO, USA

DOI: http://dx.doi.org/10.1145/2908812.2908841

Despite being a challenging problem, the TTP formulation is often criticized for not being realistic enough since there is only one thief who is responsible for traveling a number of cities to steal items. This is especially important since the TTP is supposed to be considered as an artificial representation of similar real world problems encountered within the domain of delivery, scheduling and routing.

In this paper we propose a new variant of the TTP called the multiple traveling thieves problem (MTTP). The MTTP is an extension of the present TTP formulation to include multiple thieves. This not only makes the problem more realistic as it considers groups of moving entities, but it also adds additional difficulty since there will be multiple tours and packing plans which will need to be simultaneously optimized. There also exists an interdependence between the different thieves' tours and packing plans which means that optimizing the components in isolation does not necessarily yield the best results.

The formulation differs from the problems within the vehicle routing domain in that not all cities have to be visited, and even for cities which are visited not all items have to be picked up. Also our formulation allows for a city to be visited by multiple thieves. The formulation is also done in such a way that the problem can be easily transitioned into a single thief TTP. This also means that the current set of benchmark instances can also be used for evaluating the performance of algorithms designed for solving MTTP.

We also propose a number of heuristic approaches for solving the MTTP. These approaches utilize some simple and intuitive operators for efficiently solving the given problem. Some operators, such as the Cross Thief Mutation operator, are designed in such a way that they can easily be used on the single thief TTP as well.

The rest of the paper is organized as follows. Section 2 gives a detailed description of the original TTP formulation. Section 3 discusses the short comings of the current problem and gives details of the newly proposed multiple traveling thieves problem. Sections 4 and 5 discuss the details of the heuristics used for solving the MTTP and the results obtained. The paper concludes with an overall summary and discussion on possible future research directions in Section 6.

2. TRAVELING THIEF PROBLEM

In the following, we first define the TTP. Then, we provide an overview of current state-of-the-art approaches.

2.1 Problem Description

We use the definition of the TTP by Polyakovskiy et

al. [14]. Given is a set of cities $N = \{1, \ldots, n\}$ and a set of items $M = \{1, \ldots, m\}$ distributed among the cities. For any pair of cities $i, j \in N$, we know the distance d_{ij} between them. Every city i, except the first one, contains a set of items $M_i = \{1, \ldots, m_i\}$, $M = \bigcup_{i \in N} M_i$. Each item k positioned in the city i is characterized by its profit p_{ik} and weight w_{ik}, thus the item $I_{ik} \sim (p_{ik}, w_{ik})$. The thief must visit all cities exactly once starting from the first city and returning back to it in the end. Any item may be selected in any city as long as the total weight of collected items does not exceed the specified capacity W. A renting rate R is to be paid per each time unit taken to complete the tour. v_{max} and v_{min} denote the maximal and minimum speeds that the thief can move. The goal is to find a tour, along with a packing plan, that results in the maximal profit.

The objective function uses a binary variable $y_{ik} \in \{0, 1\}$ that is equal to one when the item k is selected in the city i, and zero otherwise. Also, let W_i denote the total weight of collected items when the thief leaves the city i. Then, the objective function for a tour $\Pi = (x_1, \ldots, x_n)$, $x_i \in N$ and a packing plan $P = (y_{21}, \ldots, y_{nm_i})$ has the following form:

$$Z(\Pi, P) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{ik} y_{ik} - R \left(\frac{d_{x_n x_1}}{v_{max} - \nu W_{x_n}} + \sum_{i=1}^{n-1} \frac{d_{x_i x_{i+1}}}{v_{max} - \nu W_{x_i}} \right)$$

where $\nu = \frac{v_{max} - v_{min}}{W}$ is a constant value defined by input parameters. The minuend is the sum of all packed items' profits and the subtrahend is the amount that the thief pays for the knapsack's rent equal to the total traveling time along Π multiplied by R.

We provide a brief example in the following (see Figure 1 [14]). Each city but the first has an assigned set of items. Let us assume that the maximum weight $W = 3$, the renting rate $R = 1$ and v_{max} and v_{min} are set as 1 and 0.1, respectively. Then the optimum objective value is $Z(\Pi, P) = 50$ for $\Pi = (1, 2, 4, 3)$ and $P = (0, 0, 0, 1, 1, 0)$. This means that the thief collects no items traveling from city 1 to city 3 via cities 2 and 4. Therefore, this part of the tour has a cost of 15. In the city 3 only items I_{32} and I_{33} are picked up, resulting in a total profit of 80. However, on the way from city 3 back to city 1 the thief's knapsack has a weight of 2. This reduces the speed and results in an increased cost of 15. Consequently, the final objective value is $Z(\Pi, P) = 80 - 15 - 15 = 50$.

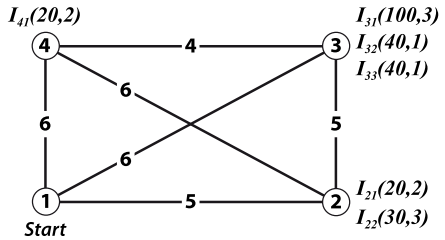

Figure 1: Illustrative example for a TTP instance [14].

2.2 Current State-of-the-Art

Polyakovskiy et al. [14] proposed the first set of heuristics for solving the TTP. Their approach was to solve the problem using a two-step procedure. The first step involved generating a good TSP tour by using the classical Chained Lin-Kernighan heuristic. The second step involved keeping the tour fixed and applying a packing heuristic for improving the solution. Their first approach was a simple heuristic (SH) which constructed a solution by processing and picking items that maximized the objective value according to a given tour. Items were picked based on a *score* value that was calculated for each item to estimate how good it is according to the given tour. They also proposed two iterative heuristics, namely the Random Local Search and (1+1)-EA, which based on certain probabilistic calculations flipped a number of packing bits. After each iteration the solution was evaluated and if an improvement was noted, the changes were kept else they were ignored.

Bonyadi et al. [4] investigated experimentally the interdependency between the TSP and Knapsack components of the TTP. They proposed two heuristic approaches named Density-based Heuristic (DH) and CoSolver. DH is again a two-phased approach similar to the SH from [14]. CoSolver is a method inspired by coevolution based approaches, which divides the problem into sub-problems where each sub-problem is solved by a different module of CoSolver. The communication between the different modules and sub-problems allows for the interdependencies to be considered. DH was outperformed by CoSolver on larger instances.

Mei et al. [11] analyzed the mathematical formulation to show that the TTP problem is not additively separable. The authors also used two separate approaches for solving the TTP. First a cooperative coevolution based approach similar to CoSolver, and then a memetic algorithm which solves the problem as a whole. The memetic algorithm outperformed cooperative coevolution. The work by Bonyadi et al. [4] and Mei et al. [11] highlight the importance of considering interdependencies between the TTP components as this will allow for the generation of more competitive solutions.

Faulkner et al. [6] investigated multiple operators and did a comprehensive comparison with existing approaches. They proposed a number of operators, such as BITFLIP and PACK-ITERATIVE, for optimizing the packing plan given a particular tour. They also proposed an insertion operator for iteratively optimizing the tour given a particular packing. They combined these operators in a number of simple and complex heuristics. The main observation was that there does not yet seem to be a single best algorithmic paradigm for the TTP. The operators, however, were quite beneficial in improving the quality of results.

Wagner [20] recently investigated the use of swarm intelligence approaches. These focus less on short TSP tours, but more on good TTP tours, which can be longer. On small instances with up to 250 cities and 2000 items, these approaches are the best fast approaches known to us to date.

3. PROPOSING A NEW VARIANT OF THE TRAVELING THIEF PROBLEM

3.1 Criticism of TTP

While the TTP is a challenging problem it is often criticized for not being realistic. One thief traveling across 500, 1000 and sometimes even 10000+ cities to collect items may result in an enticing computer science problem, but in reality this would most likely be a very inefficient way of collecting items. In reality if there is an organization or entity working in a similar line of business, profit would be an

important factor, and they would definitely want to invest resources (e.g. using additional transporting entities) in trying to make the collection of items more efficient.

3.2 Multiple-Traveling Thief Problem

In response to the above criticism we propose the multiple-traveling thieves problem (MTTP). To the best of our knowledge there currently does not exist any variant to the original TTP problem. Given that a single thief's profit depends on the time traveled, which depends on the distance traveled and the items packed, it is to be expected that in some situations multiple thieves can yield a higher total profit if they travel *lighter*. While this does increase the possibility of greater total profit it also introduces some challenges: now that there are multiple thieves, the optimization algorithm will need to simultaneously optimize several tours and packing plans.

In its most basic form the MTTP consists of t thieves traveling m cities with the goal of collecting items of value that are situated at each of these locations. There are a total of q items which are located across the m cities. All thieves have to start and end their tour at a common location which for this problem is the very first city as it is assumed that the thieves are all working together and they need to return to the initial city to divide the loot. Each thief can visit a set of j cities, given that $j \leq m$. Multiple thieves can visit the same city but each item can only be picked up by one thief. A thief can choose to visit a city and pick up one or multiple items or not pick up any items at all. Visiting all cities is not mandatory.

Each thief can pick up as many items as possible as long as he/she respects the total knapsack capacity, which means that the thieves do not get assigned individual knapsack capacities. A real-world motivation for this can look as follows. Let us assume that the thieves want to transport the stolen goods via ship to some other country and the ship has a capacity limit. Since the thieves know the routes and packing plans before starting the journey they can pick appropriate vehicles among themselves: so a thief with smaller route might pick a smaller truck and another might pick a bigger one. Also, this design decision allows us to seamlessly transition between different number of thieves. For example, one thief might use almost all the capacity, whereas another thief just briefly goes to one city, picks one item, and then comes back. While this strategy might sound a bit artificial and not very much real-world like, it is allowed in our formulation. This ability to transition is important, as the MTTP should, just like the TTP, be as well-structured and general as possible: the core motivation behind the TTP and the MTTP is to facilitate the investigation of the different interacting components of the Knapsack Problem and the Traveling Salesperson Problem.

3.3 Similarities with Other Problems

The closest problem within literature to our MTTP is the vehicle routing problem (VRP). The VRP is concerned with finding optimal routes for a fleet of vehicles delivering or collecting items from different locations [8]. Often, all vehicles have a common starting and end point (depot). In addition, it is common that each city is to be visited exactly once and by only one vehicle. The VRP first appeared in the paper by Dantzig and Ramser [5] who were interested in finding optimal routes for gasoline delivery trucks. Since then a number of variants have been proposed. The capacitated VRP extends the classical VRP by imposing a constraint such that the total load transported by a vehicle cannot exceed its capacity. The multi-depot VRP [7, 12, 16] considers multiple points from which clients can be serviced. The VRP with time windows [18] adds another layer of realism as there exist time windows within which service should be made to a particular city and arriving early or late can result in penalties. There also exist other variants such as stochastic VRP [16], dynamic VRP [15], VRP with time-dependent prize collection [2], and with selective pickup and delivery [19], among countless other extensions.

It is important to note that our MTTP, just like the original TTP, is unlike many capacitated vehicle-routing problems in the area of Green Logistics (see, e.g., the survey article [9]). In our case, we consider in addition to the routing problem not only a load-dependent feature, but also the NP-hard optimization problem of deciding which items are to be stolen by the thieves.

Some obvious differences between the classical VRP and our proposed MTTP are:

- The classical VRP requires that all cities be visited exactly once while in the MTTP the thieves can choose not to visit a city. There can be various reasons for not visiting a city such as, for example, the city does not hold high value items, or that the travel time involved in visiting the city is not worth the gain.
- For the classical VRP all items must be delivered and/or picked up from all the cities. In the MTTP, as mentioned in the previous difference, not all cities have to visited and thus naturally not all items have to picked up. In addition, even for cities which are visited it may be possible that not all items are picked or in some cases none of the items are picked. This is all part of the profit maximization scheme.
- In the classical VRP the weight of the items does not play a part in slowing down or speeding up delivery or pickup. In the MTTP the order in which the items are picked has a huge impact even if the route taken is the shortest.

Based on the above discussion it is clear that MTTP not only makes the TTP problem much more realistic but also adds an additional layer of difficulty. It also adequately fills the gap left by the VRP and its variants, as the MTTP is the systematic combination of two combinatorial problems.

4. MTTP LOCAL SEARCH ROUTINES

In this section, we present a number of local search routines used to construct packing plans and tours.

4.1 Solution Representation

A single solution for the MTTP can consist of several tours and packing plans. We use the following representation for solutions. For consistency all tour vectors are of the same size and all packing plan vectors are also of the same size. For a single thief the cityIDs (city identifiers within the vector) are represented as positive values if the thief is visiting these cities and as negative values if the thief is not visiting particular cities. Similarly for the packing plan, if the thief is picking an item from a city, it is represented by 1 and if no items are picked or if the thief is not visiting that city then it is represented as 0.

4.2 Initialization

Since this is a new problem it was important that we experiment with a range of alternative initialization procedures. We have implemented four alternative initialization procedures. The tour initialization is based on an optimal or near optimal (in terms of shortest overall distance) tour generated using the Chained Lin-Kernighan heuristic (CLK) [1].[1]

The following four approaches cover a range of different scenarios allowing us to discover a good starting point:

- Init1: Assign complete CLK tour to a single thief. The initial tours for the remaining thieves do no involve any travel. All thieves start with an empty packing plan. This is equivalent to starting with a TTP solution.
- Init2: Iterate through the CLK tour assigning each city randomly to one thief. Each city is visited by one thief and all thieves start with an empty packing plan.
- Init3: Iterate through the CLK tour assigning each city randomly to one thief. A city can be visited by multiple thieves provided the city has more than one item. All thieves start with an empty packing plan.
- Init4: Each thief is assigned a chunk of the CLK tour. Assuming there are n thieves, the size of each chunk would be $(totalCities/n)$. Each city is only visited by one thief. Iterative greedy packing using PACKITERATIVE [6] is then applied to the individual tours. PACKITERATIVE considers the profits and weights of the items, and also their distance to the final city based on the provided tour.

4.3 Drop City Operator

The DROPCITY operator is a very simple and intuitive approach. The aim of the thieves is to maximize the overall profit so not only would they want to visit cities which are highly profitable, but they would also want to avoid cities which have the opposite effect. Visiting some cities may not be reasonable if the city contains low value items or when the distance involved in traveling is too high and the cost is not worth the gain.

Our proposed distance-reducing operator takes in a solution together with cityID and thiefID. It checks if the thief is visiting the city specified by the cityID. If it is, then the city is removed from the tour and the items picked from that city are flushed out of the thief's packing plan.

This can then be done iteratively to remove cities which are not yielding the desired gain to make up for the additional travel time.

4.4 Cross Thief Mutation

The CROSSTHIEFMUTATION is used for transferring tour and packing information between thieves. It takes as input a cityID and then identifies which thief (if any) is visiting that city. In case there are multiple thieves visiting that city, the first thief is selected. This thief is called $\text{thief}_{\text{old}}$ The next step is to randomly choose a new thief ($\text{thief}_{\text{new}}$) which may or may not be visiting that city in its current tour. The packing plan for the corresponding city from $thief_{old}$ is then transferred to $\text{thief}_{\text{new}}$. Once the packing information has been transferred the corresponding city is removed from $\text{thief}_{\text{old}}$ tour. It also means that $\text{thief}_{\text{new}}$ is now visiting that city.

This can help optimize the tour and packing plans simultaneously for multiple thieves. It can also be used as a crossover operator in the original TTP formulation.

4.5 Additional Routines

In our study we also include operators present within the current literature. Modification of these approaches for MTTP was simple as we only added a simple check to ensure only active items and tour elements were modified i.e. those corresponding to the locations the thief was visiting while other non-visiting locations were left untouched.

4.5.1 Bit Flip

The BITFLIP operator was originally proposed in [6]. The operator iterates through a given packing plan and sequentially flips the packing bits. If an improvement is seen in fitness the changes are kept else they are ignored. This process is then repeated for the next bit. In short, the bit flip operator optimizes the packing plan given a particular tour.

4.5.2 Insertion

The INSERTION operator [6] is used to modify and optimize the tour based on a given packing. The operator works by removing a city from a given location in the tour and inserting it at a different location. It may be possible that some valuable items are picked at the beginning of the tour due to the order of the cities in the route. Hence the insertion operator tries to delay the pickup of such items by rearranging the tour such that these items are picked towards the end so that there is minimal impact on the speed of the thief for the earlier parts of the tour.

A good TTP solution can be characterized by the packing of fewer or lighter items at the beginning of the tour and the packing of higher value items at the end of the tour. The insertion operator tries to achieve exactly this.

4.5.3 1+1 EA

The (1+1)-EA is a simple hill-climber used by Polyakovskiy *et al.* [14]. Similar to the BITFLIP operator, the (1+1)-EA also tries to optimize the packing plan given a particular tour. It iterates through a packing plan and inverts each item independently with probability of $1/q$. The updated solution is then evaluated to see if fitness improved. If an improvement is noted then the changes are kept, else they are ignored. The above steps are repeated several times to improve fitness.

5. LOCAL SEARCH ALGORITHMS

In order to isolate the effects of the different search routines, we first define a set of straight-forward local search algorithms:

- Heuristic S1: initialize, then BITFLIP for each thief;
- Heuristc S2: initialize, then (1+1)-EA for each thief;
- Heuristc S3: initialize, then INSERTION for each thief;
- Heuristc S4: initialize, then DROPCITY for each thief;
- Heuristc S5: initialize, then CROSSTHIEFMUTATION once for each city;

The first four procedures are repeated for all thieves allowing us to optimize each tour and packing plan in a step-by-step

[1]As available at http://www.tsp.gatech.edu/concorde/downloads/downloads.htm

approach. Individual attention is given to each component of the overall solution. The CrossThiefMutation uses a more global approach by which a number of tours and packing plans are optimized simultaneously.

Since we expect the different routines to be able to escape each other's local optima, we also define the following slightly more complex heuristics:

- Heuristic C1: *repeat* initialization *until time is up*;
- Heuristic C2: initialization, then *repeat* CrossThiefMutation once for each city; DropCity for each thief *until time is up*;
- Heuristic C3: initialization, then *repeat* CrossThiefMutation (to distribute cities); Bitflip (to maximize packing); DropCity (to drop unprofitable cities), Insertion (to rearrange cities) *until time is up*;

The motivation for C1 with its completely independent initializations without any subsequent hill-climber is that its equivalent in [6] by far outperformed more complex approximate approaches to the TTP. By considering our MTTP equivalent, we can investigate the importance of a good initial CLK tour. The heuristic C2 is based purely on our new operators. This allows us to collectively establish the efficiency of the new operators. In heuristic C3 we combine the search routines available to us. While using multiple operators may be beneficial, the amount of time spent per operator will be lower than in the dedicated simple heuristics due to the rotation.

These heuristics allow us to investigate by how much good TTP solutions can be improved when considering them in a MTTP scenario, where multiple thieves are allowed and where it is not necessary to visit every city.

5.1 Experimental Setup

For our investigations, we use the set of TTP instances defined by Polyakovskiy et al. [14][2]. In these instances, the two components of the problem have been balanced in such a way that the near-optimal solution of one sub-problem does not dominate over the optimal solution of another sub-problem. The characteristics of the original 9,720 instances vary widely. In short, they are based on instances from the TSPlib by Reinelt [17], and on different types of knapsacks as described by [10]. Also, the knapsack capacities and the items per city vary. Lastly, for each instance, the renting rate R that links both subproblems is chosen in such a way that at least one TTP solution with zero negative objective value exists.

For our experiments, we use the 72 instances as chosen by Faulkner et al. [6]. This is a representative subset to cover small, medium, and large size instances with different characteristics:

- six different numbers of cities (spread out roughly logarithmically): 195, 783, 3038, 11849, 33810, and 85900;
- two different numbers of items per city: 3, and 10;
- all three different types of knapsacks;
- two different sizes of knapsacks (capacities): 3 and 7 times the size of the smallest knapsack.

For our MTTP study, we can reuse these existing TTP instances without any modifications.

[2]As available at http://cs.adelaide.edu.au/~optlog/research/ttp.php

We run all algorithms for a maximum of 10 minutes per instance. Due to their randomized nature, we perform 30 independent repetitions of the algorithms on each instance. All computations are performed on machines with Intel Xeon E5430 CPUs (2.66GHz) and Java 1.8. Note that our code and results are available online.[2]

We evaluate the different algorithms and heuristics as follows. First, we calculate the average objective scores for each approach on each instance over the independent runs. We first rank the heuristics for each instance, and then we average these rankings based on the aspect that we are interested in, e.g., the number of thieves or the algorithm.

5.2 Study I: Initializations

In this first study, we investigate the effects of the number of thieves involved and of the different initialization strategies. Figure 2 shows the average ranks achieved by the different combinations across all 72 instances. We can observe that an increase in the number of thieves beyond two results in a worse rank. This is the case since all traveling thieves share the same start and end city, and thus the total travel costs increase. This is also reflected in the study of the initialization schemes, where Init1 (only one travelling thief) performs better than Init2 and Init3 (all thieves travel, but do not pick any items). Init4 performs the best, as it creates good packing plans for all traveling thieves.

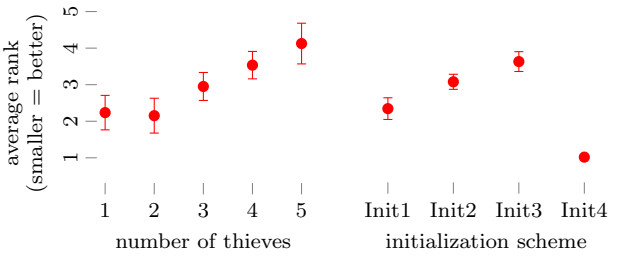

Figure 2: Study I: Initialization. Shown are the average ranks and their standard deviations.

5.3 Study II: Local Search Routines

Next, we investigate the benefits of the local search routines presented in Section 5. Based on the results of our previous study, we use two thieves and Init4.

In Figure 3 we show again the average ranks of the different approaches across all 72 instances. While all search routines can improve upon the initial solution, the effect of the routines presented in [6] (used in our S1–3) is marginal. In contrast to this, our new problem-specific operators used in S4 and S5 result in massive improvements.

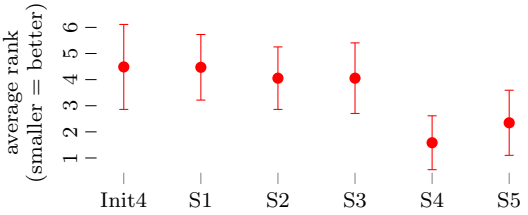

Figure 3: Study II: Local Search Routines. All use two thieves and Init4. Shown are the average ranks and their standard deviations.

5.4 Study III: Complex Algorithms

Lastly, we investigate our most complex approaches C1–3. Again, we use Init4 as the initialization scheme.

In Figure 4 we show the results of the algorithms when the number of thieves is varied. Just as before, we see that the number of thieves has an important influence on the final objective scores. Even though using two thieves seems to be a sweet spot on average, the standard deviation is again very high. C1 is the best performing algorithm, with a relatively little standard deviation. C3, however, sometimes outperforms it. Interestingly, the restarting algorithm C1 achieves the lowest rank on average, which is similar to the observation made in [6], where a comparable simple restarting algorithm performed best on average. We take this as an indicator that the TSP part of the MTTP still is of very high importance.

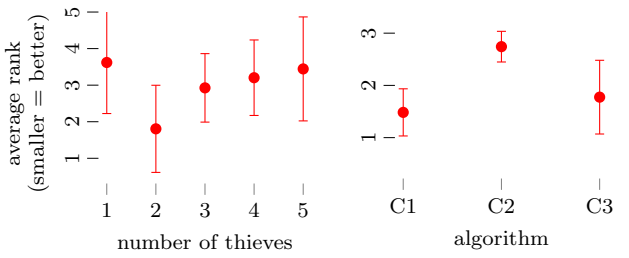

Figure 4: Study III: Complex Algorithms. All use Init4. Shown are the average ranks and their standard deviations.

As this is a very compressed view on the outcomes, we show a decision tree in Figure 5 that classifies the best algorithm for the given instances. For example, we can see that for larger instances (right half of the tree) C1 should be used. The more complex C3 with its additional operators performs better on smaller instances (left half of the tree). Also, we can see that when the number of items increases, then additional thieves are of benefit (*-4*, *-5*).

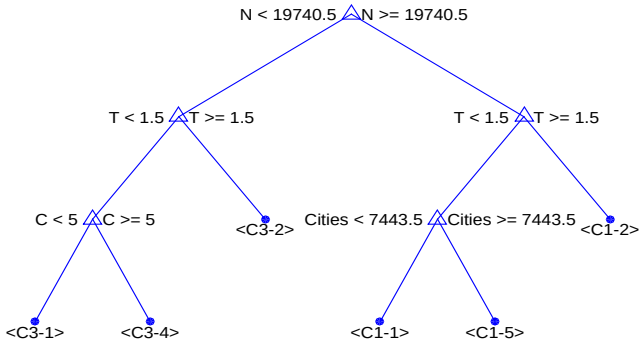

Figure 5: Decision Tree for the algorithm and thief selection.

For a single solution, we show in Figure 7 how it was evolved by C3. Firstly, we can see that the green thief's tour was reduced significantly in the first iteration and even further until the final iteration. We conjecture that this thief could be dropped completely to further increase efficiency, but the mutation step could not have been performed by our operators. This conjecture is supported by the fact that the average objective score for C3 with two thieves is indeed higher than the score for C3 with three thieves. Secondly, we can see that often large proportions of the profits available in a city are stolen (shown as large dots), while the corresponding weight dots remain often small. Thirdly, we can see that the dot sizes change over time (showing local improvement of the packing plan), and that the thieves exchanged cities.

To show how the algorithms perform differently on different instances, we list the average objective scores achieved for the smallest and largest instances in Figure 7. We can see that for the smallest instances the complex algorithm C3 with two or three thieves dominates the other configurations. For the largest instances, however, the number of thieves seems to be much more important than the choice of the algorithm. In addition, we can see that larger number of thieves are beneficial for the pla85900 instances when the weights are bounded and strongly correlated. Otherwise, two to three thieves achieve the best results. Interestingly, negative scores are achieved four times for five thieves, showing that the algorithm could not transition from the five thief-setup to the more efficient four thief-setup.

Lastly, we would like to comment that the configuration C1 with one thief is equivalent to the best-performing TTP algorithm reported in [6]. That particular algorithm is always outperformed by our MTTP approaches, regularly by 50% to 100%.

6. CONCLUDING REMARKS

In this paper we proposed the Multiple Traveling Thieves Problem, which is a variant of the recently proposed Traveling Thief Problem. The motivation for this variant is that two assumptions were made in the TTP's definition: the number of thieves was limited to one and all cities had to be visited. By removing these constraints, the problem becomes more comparable to real-world scenarios. At the core, the MTTP remains a simple combination of two well-known combinatorial optimization problems.

The MTTP consists of multiple thieves which can visit a series of cities containing valuable items in the hope of collectively maximizing their profit. The problem formulation has been discussed in detail and a number of heuristic approaches have been used to solve the newly proposed MTTP. Together with this a number of problem specific operators have been proposed, which may also be useful for future research within the domain of the original TTP.

The MTTP has been formulated as such that the existing TTP instances can be reused for benchmarking the performance of algorithms. All code and results presented in this paper is also available online to encourage further research within this domain.[2]

Based on the investigated instances, one observation is that by introducing additional thieves, the objective score can be increased significantly, but not arbitrarily. Also, the restart algorithm that was very successful in the TTP case was outperformed by MTTP approaches as this time the problem-specific hill-climbers could mitigate the problem of the importance of the initial tours.

Future research on problems with interdependent components is in need of systematic theoretical and experimental investigations. For example, Nallaperuma et al. [13] systematically investigated and exploited features of TSP instances. A first step specific to the TTP was recently taken by Wu et al. [21] who investigated the impact of the renting rate on instance hardness.

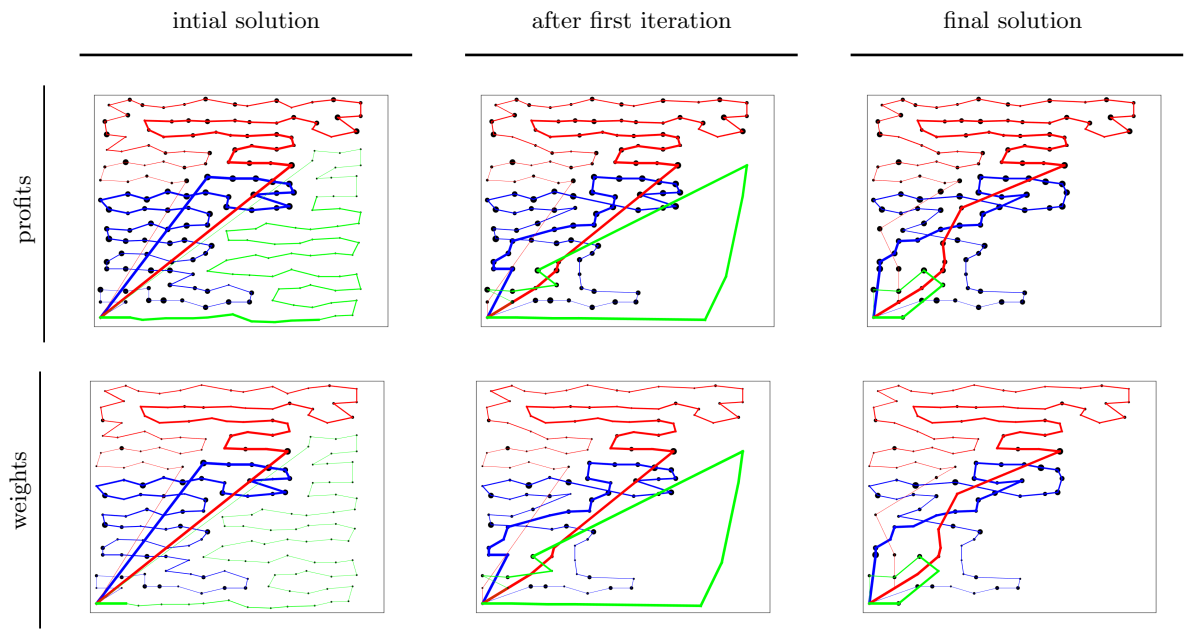

Figure 6: Evolution of a solution for rat195_n1940_uncorr_03.ttp, using algorithm C3 and three thieves. The objective scores from left to right are 72294, 151947, and 192556. The best result reported in [6] is 157882. Top row: for each thief the line thickness increases relative to each thief's total profit, the diameter of the city dots shows the proportion of the available profit in that city that was stolen. For example, a thin line shows a light traveling thief, and a small dot means that little or nothing in that city was stolen. The bottom row shows the same with a focus on the weights of the stolen items.

References

[1] D. Applegate, W. J. Cook, and A. Rohe. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003.

[2] D. Black, R. Eglese, and S. Wohlk. The time-dependent prize-collecting arc routing problem. *Computers & Operations Res.*, 40(2):526 – 535, 2013.

[3] M. R. Bonyadi, Z. Michalewicz, and L. Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *IEEE CEC*, pages 1037–1044, 2013.

[4] M. R. Bonyadi, Z. Michalewicz, M. R. Przybylek, and A. Wierzbicki. Socially inspired algorithms for the TTP. In *GECCO*, pages 421–428. ACM, 2014.

[5] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1): 80–91, 1959.

[6] H. Faulkner, S. Polyakovskiy, T. Schultz, and M. Wagner. Approximate approaches to the traveling thief problem. In *GECCO*, pages 385–392. ACM, 2015.

[7] R. Kulkarni and P. Bhave. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20(1):58 – 67, 1985.

[8] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European J. of Operational Research*, 59(3):345 – 358, 1992.

[9] C. Lin, K. Choy, G. Ho, S. Chung, and H. Lam. Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, 41(4, Part 1):1118 – 1138, 2014.

[10] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.

[11] Y. Mei, X. Li, and X. Yao. On investigation of interdependence between sub-problems of the travelling thief problem. *Soft Computing*, 20(1): 157–172, 2014.

[12] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jimenez, and N. Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Indust. Engineering*, 79:115 – 129, 2015.

[13] S. Nallaperuma, M. Wagner, and F. Neumann. Analyzing the effects of instance features and algorithm parameters for max min ant system and the traveling salesperson problem. *Frontiers in Robotics and AI*, 2(18), 2015.

[14] S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann. A comprehensive benchmark set and heuristics for the traveling thief problem. In *GECCO*, pages 477–484. ACM, 2014.

[15] H. Psaraftis. Dynamic vehicle routing: Status and prospects. *Annals of Op. Res.*, 61(1):143–164, 1995.

[16] S. Raff. Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63 – 211, 1983.

[17] G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. *ORSA J. on Computing*, 3(4):376–384, 1991.

[18] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[19] C.-K. Ting and X.-L. Liao. The selective pickup and delivery problem: formulation and a memetic algorithm. *Production Econ.*, 141(1):199 – 211, 2013.

[20] M. Wagner. Stealing items more efficiently with ants: a swarm intelligence approach to the traveling thief problem. 2016. Internal technical report.

[21] J. Wu, S. Polyakovskiy, and F. Neumann. On the impact of the renting rate for the unconstrained nonlinear knapsack problem. In *GECCO*, 2016. Accepted.

rat195_n582_bounded-strongly-corr_03.ttp

thieves	C1	C2	C3
1	86533	78701	127370
2	64363	58907	112227
3	47955	44324	116202
4	40188	27362	117529
5	43432	37331	124634

pla85900_n858990_bounded-strongly-corr_03.ttp

thieves	C1	C2	C3
1	113197420	109082303	109969587
2	113942500	112068117	106088624
3	97381542	91776781	93551105
4	116408497	115948204	115770143
5	132255225	130916660	131085574

rat195_n582_bounded-strongly-corr_07.ttp

thieves	C1	C2	C3
1	110115	106289	167538
2	140640	112489	209945
3	126921	124093	216463
4	128572	119029	206717
5	113327	110846	185562

pla85900_n858990_bounded-strongly-corr_07.ttp

thieves	C1	C2	C3
1	168908454	166793490	165359355
2	282558620	277124430	279297952
3	313458440	311989181	310322117
4	335087490	333625458	333826907
5	340099283	338943956	338067872

rat195_n582_uncorr-similar-weights_03.ttp

thieves	C1	C2	C3
1	27908	27008	53665
2	28667	25824	62476
3	17595	15668	56073
4	7429	5002	53580
5	-186	-5233	55118

pla85900_n858990_uncorr-similar-weights_03.ttp

thieves	C1	C2	C3
1	48875311	48185946	47519548
2	64900668	64513574	64711850
3	58473002	58448237	58240737
4	55626375	55184745	54993949
5	54199603	53754131	54101145

rat195_n582_uncorr-similar-weights_07.ttp

thieves	C1	C2	C3
1	48322	45356	70418
2	80304	72343	95672
3	50828	48580	78727
4	30856	24880	66348
5	23146	20469	56360

pla85900_n858990_uncorr-similar-weights_07.ttp

thieves	C1	C2	C3
1	81245670	80944687	80203676
2	162125642	161613298	160927085
3	127396590	126765527	126933157
4	137398900	136316408	136217411
5	134202199	134098046	132545553

rat195_n582_uncorr_03.ttp

thieves	C1	C2	C3
1	57006	54559	68478
2	64117	61748	78527
3	37708	27953	61757
4	15021	6790	53308
5	-1208	-7486	43172

pla85900_n858990_uncorr_03.ttp

thieves	C1	C2	C3
1	71602197	70792529	71198150
2	103843224	102632216	102531341
3	70526890	70465286	70874201
4	50480499	49479155	49539461
5	36231332	35367872	35314878

rat195_n582_uncorr_07.ttp

thieves	C1	C2	C3
1	71267	68777	81600
2	99377	92394	109198
3	83754	78559	90572
4	50065	42988	78760
5	45901	36464	66318

pla85900_n858990_uncorr_07.ttp

thieves	C1	C2	C3
1	102737545	102050121	101702667
2	179210906	178195395	178457134
3	159849495	158812255	157945818
4	136021157	134870968	134842329
5	141747002	140960220	140627998

Figure 7: Average objective scores achieved. The number of cities is 195 and 85900, the number of items is 582 and 858990, there are three different types of knapsacks, and the knapsack capacity varies in two instance-specific steps (03 and 07). The cells are formatted to show the performances from lowest (blue) to highest (red) objective score in each block of 15 cells. Negative values show that it would have been best not to travel at all—something that was not allowed in the original TTP formulation, but it is now in our MTTP formulation.