Stealing items more efficiently with ants: a swarm intelligence approach to the travelling thief problem

Markus Wagner
Optimisation and Logistics, The University of Adelaide, Australia,
markus.wagner@adelaide.edu.au

Abstract. The travelling thief problem (TTP) is an academic combinatorial optimisation problem in which its two components, namely the travelling salesperson problem (TSP) and the knapsack problem, interact. The goal is to provide to a thief a tour across all given cities and a packing plan that defines which items should be taken in which city. The combining elements are the knapsack's renting rate that is to be paid for the travel time, and the thief's slowdown with increasing knapsack use. Previously, successful algorithms focussed almost exclusively on constructing packing plans for near-optimal TSP tours. Even though additional hill-climbers are used at times, this strong initial bias prevents them from finding better solutions that require longer tours that can give rise to more profitable packing plans. Our swarm intelligence approach shifts the focus away from good TSP tours to good TTP tours. In our study we observe that this is effective and computationally efficient, as we outperform state-of-the-art approaches on instances with up to 250 cities and 2000 items, sometimes by more than 10%.

Keywords: MAX-MIN Ant System, Travelling Thief Problem

1 Introduction

The travelling thief problem (TTP, [2]) is fast gaining attention for being a challenging combinatorial optimisation problem. This NP-hard optimisation problem combines two well-known combinatorial optimisation problems, namely the travelling salesperson problem (TSP) and the knapsack problem. The two components have been merged in such a way that the optimal solution for each of them does not necessarily correspond to an optimal TTP solution. The motivation for the TTP is to have a problem where the interactions of interdependent problem components can be investigated systematically.

So far, constructive heuristics, simple and complex hill-climbers, and also more sophisticated co-evolutionary approaches have been applied to the TTP. The drawbacks of these approaches are that they either focus almost exclusively on good TSP tours, or that they cannot navigate the search space neither effectively nor efficiently. So far, minimally problem-specific local searches that alternate between solving the TSP and KP components appear to perform best.

In this article, we propose the use of swarm intelligence based on ant colony optimisation in order to solve the TTP's tour part. The packing part is then computed heuristically for each tour, and after each iteration the best solution is improved further using additional hill-climbers. Even though the local search in the tour generation still focusses on the TSP part, the individuals in the swarm are assessed based on the solution's TTP objective score. The use of swarm intelligence allows us to explore different tours in a collaborative fashion, and we are no longer limited by a single "current" tour. As we shall see later, this added flexibility and shift from good TSP tours to good TTP tours is very beneficial.

We proceed as follows. Section 2 gives a brief description of the original TTP formulation and discusses related work. In Section 3, we describe our approach for stealing items more efficiently with the help of swarm intelligence. We present and discuss the results of our experimental study in Section 4, before we conclude with remarks on possible future research directions.

2 Traveling Thief Problem

2.1 Problem Description

We use the definition of the TTP by Polyakovskiy et al. [11]. Given is a set of cities $N = \{1, \dots, n\}$ and a set of items $M = \{1, \dots, m\}$ 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 $M_i = \{1, \dots, m_i\}$, $M = \bigcup_{i \in N} M_i$. Each item k positioned in the city i is characterised 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. υ_{max} and υ_{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 $\Pi = (x_1, \dots, x_n)$, $x_i \in N$ and a packing plan $P = (y_{21}, \dots, y_{nm_i})$ has the following form:

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

where $\nu = \frac{\upsilon_{max} - \upsilon_{min}}{W}$ 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.

2.2 Current State-of-the-Art

Polyakovskiy et al. [11] proposed the first set of heuristics for solving the TTP. Their approach was to solve the problem using two steps. The first step involved generating a good TSP tour by using the classical Chained Lin-Kernighan heuristic [1]. The second step involved keeping the tour fixed and applying a packing heuristic for improving the solution.

Bonyadi et al. [3] and Mei et al. [7] investigated experimentally and theoretically the interdependency between the TSP and knapsack components of the TTP. They proposed heuristic approaches including coevolutionary ones and a memetic algorithm. The latter called MATLS considered the interdependencies in more depth and outperformed cooperative coevolution.

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 PACKITERATIVE, for optimising the packing plan given a particular tour. They also proposed INSERTION for iteratively optimising the tour given a particular packing. They combined these operators in a number of simple and complex heuristics that outperformed existing approaches.

Recently, a relaxed version of the TTP was presented by Chand and Wagner [4] as reaction to the criticism that the TTP is not realistic. In the new version of the problem multiple thieves are allowed to travel across different cities (not necessarily across all) with the aim of maximising the group's collective profit.

Note that even when the tour is kept fixed, packing is NP-hard [10].

3 Using ants to steal items

While swarm intelligence does not easily offer provable performance guarantees, it does give us a means of working on the tour part of the TTP, on top of which we can run other heuristics. Effectively, our approach is a bi-level one, where the ants are assessed based on the TTP solution for which they created the tour.

The packing heuristic of our choice is the fast and effective PACKITERATIVE [6]. It considers the items' profits and weight, and also their distances to the final city based on the provided tour. Its characteristic feature is that it performs a binary search on an internal parameter in order to fine-tune the packing.

Our implementation is built upon Adrian Wilke's ACOTSPjava 1.0.1,[1] which is based on Thomas Stützle's ACOTSP 1.0.3. The overall logic of the used swarm intelligence package remains unchanged, and our modifications are minimal.

In Algorithm 1 we show the simplified overview of our swarm intelligence approach. The TTP-specific injections are mainly in two places:

1. Whenever a tour is generated, a packing plan for it is generated using PACKITERATIVE. The tour's objective score, which is normally the total distance travelled, is replaced by the TTP solution's objective score.
2. At the end of each iteration, we run hill-climbers on the best solutions in order to achieve further improvements. We call this "boosting".

Note that we make a rather strong assumption in the first injection: as the ants' tours are assessed using PACKITERATIVE, we assume that the packing heuristic is optimal. While this is not the case, we have observed in [6] that PACKITERATIVE

[1] ACOTSPjava: `http://adibaba.github.io/ACOTSPJava/`, last accessed 28 Feb 2016.

Algorithm 1 ACOTSP for the Travelling Thief Problem (injections in italics)

while (termination condition not met)
  Construct tours using ants.
  Construct for each tour a packing plan using PackIterative, resulting in a TTP objective score. If the tour has been assessed before, we skip the packing step and retrieve the score from a cache.
  Perform local search on tours (if activated).
  Update ACO statistics.
  Boost solutions using (1+1)-EA, Insertion, Bitflip (if activated).
  Pheromone trail update.

quickly produces very good approximations of the optimal packing across a wide range of instances.

Lastly, note that we improve the runtime of the first modification by caching and retrieving `<tour,objective score>` tuples, as PackIterative is deterministic. Also, we rotate the tours before running the packing heuristic, so that they always start and end in the first city.

4 Experimental Study

4.1 Experimental Setup

For our investigations, we use the set of TTP instances defined by Polyakovskiy et al. [11].[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. For our experiments, we use 108 instances with the following characteristics:

- nine different numbers of cities (spread out roughly logarithmically): 51, 76, 100, 159, 225, 280, 574, 724, 1000;
- two different numbers of items per city: 3, and 10;
- all three different types of knapsacks: uncorrelated, uncorrelated with similar weights, bounded strongly correlated;
- two different sizes of knapsacks (capacities): 3 and 7 times the size of the smallest knapsack.

We run all algorithms for a maximum of 10 minutes per instance. Due to their randomised 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: `http://cs.adelaide.edu.au/~optlog/research/ttp.php`.

We assess the quality of the algorithms using the following approach. For each instance, we consider the best solution found to be a lower bound on the achievable objective value. Then, we take the average of the 30 results produced

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

by an algorithm and then compute the ratio between that average and the best objective value found, which gives us the approximation ratio. This ratio allows us to compare the performances across the chosen set of instances, since the objective values vary across several orders of magnitude.

4.2 MMAS configurations

The ACOTSPjava package allows us to set a large number of different parameters. One of them is the choice of the actual ant colony optimisation approach. To prevent pheromones from dropping to arbitrarily small values, we use the MAX-MIN ant system by Stützle and Hoos [12], which restricts all pheromones to a bounded interval. The MMAS parameters that we employ are the default ones in ACOTSPjava: $\rho = 0.5$, $\alpha = 1$, $\beta = 2$, ants=25, max_tours=100, tries=1, elitist_ants=100, ras_ranks=6.

In preliminary experiments, we noticed that the use of TSP-specific local search (see line 4 of Algorithm 1) was crucial for achieving good TSP tours, which is a commonly made observation (see for example [5, 12]). In our study, we employ the following two variants of local search: *ls3* runs 3-opt on a tour generated by ants, and *ls4* randomly picks for each tour either 2-opt, 2-h-opt, or 3-opt. With the latter, we allow for slightly more varied exploitations of tours.

For the boosting that we perform, we use the operators described in [6]. If boosting is performed, then we first run (1+1)-EA on the packing plan for 10,000 iterations as a hill-climber. Then we perform one pass of INSERTION, which means that for each city we attempt once to relocate it to each position in the travel sequence. Lastly, we perform one pass of BITFLIP, where for each item we check once whether changing its packing status increases the objective score. The overall computational complexity of this boosting is quadratic in the number of cities and linear in the number of item.

In summary, we investigate the following four MMAS configurations, depending on the chosen local search and depending on whether or not boosting is activated: MMASls3, MMASls3boost, MMASls4, MMASls4boost.

In our opinion, our MMAS approaches are natural successors of the approaches S5 and C3–C6 from [6]. S5 resamples new routes independently, whereas our approach resamples new routes based on the previous ones. With boosting activated, our MMAS approaches are somewhat similar to the heuristics C3–C6, which employ hill-climbers on top of single tours as well. In contrast to C3–C6, our algorithms search with distributions of tours. With this change in focus we expect performance gains, as longer tours are investigated systematically.

4.3 Comparison with state-of-the-art

We compare our MMAS-based approaches with recent ones from the literature. In particular, these are S1/S5 and C3/C4/C5/C6 [6] and MATLS [8].

In Figure 1 we show a summary of the over 1100 average approximation ratios as trend lines. We can make the following observations.

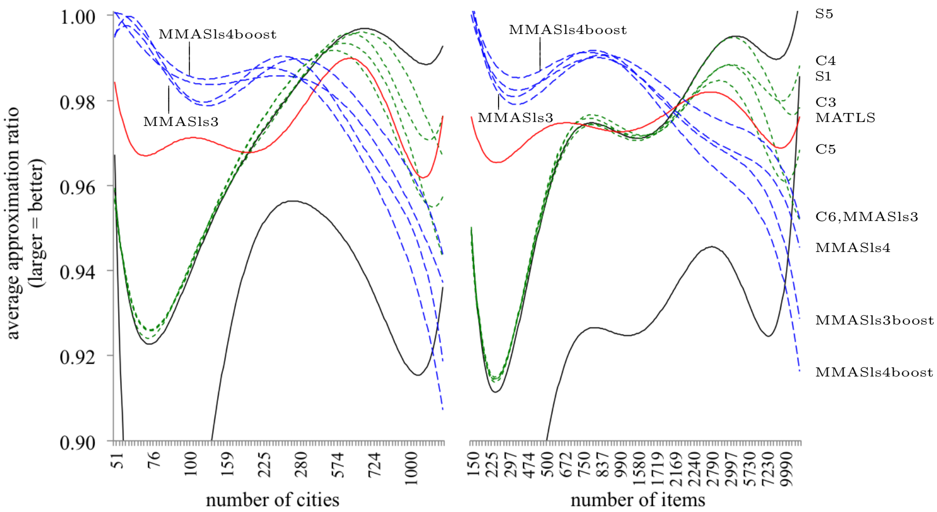

Fig. 1. Summary of results shown as trend lines. The curves are polynomials of degree six. Similar approaches are coloured identically to allow us to focus on the different types of the approaches: S1/S5 are solid black lines, C3–C6 are green short dashes, MATLS is a red solid line, and the MMAS-approaches are blue long dashes. Our MMAS-based approaches are the best performing ones for TTP instances with up to 250 cities and 2000 items, on which previously MATLS and C3–C6 performed best.

1. The baseline approach S1, where PackIterative is run only once on top of a single ChainedLinKernighan tour, is clearly outperformed by all others.
2. Our MMAS-approaches (blue) are the best performing ones for TTP instances with up to 250 cities and 2000 items. Previously, the more holistic approach MATLS (red) performed best one these.
3. Our boosting of solutions and also the variation in the TSP local search prove helpful for instances with up to 200–250 cities and 800 items. For larger instances, MMASls3 is the best performing swarm intelligence approach.
4. For instances with 250–500 cities, the complex approaches C3–C6 with their local search routines achieve the top ranks. On even larger instances, the simple resampling heuristic S5 (black) dominates, as already observed in [6].

Let us briefly look into the impact that the MMAS search has on the tours of the final solutions. In Table 1 we show details of different *best* solutions found. While most solutions might look quite similar at first glance, they differ in some fundamental aspects, of which we highlight a few in the following.

For the first instance, which is one of the smallest investigated ones, our MMASls4 was the best performing approach on average, and it also found the best solution. For both S1 and C6, their best solution is >10% worse. The reason appears to be their strong focus on the use of the Chained Lin-Kernighan heuristic, which results in a shorter tour. Even the local search routines in C6 are not sufficient to escape the local optima. In contrast to this, our MMAS-approaches successfully explore parts of the search space with longer tours.

For the second instance, which is one of the largest investigated ones, the best solutions by S1, S5, and C6 are again the ones with the shortest travelled

approach	used knapsack capacity	unused knapsack capacity	total profit of items	travel distance	travel time	objective score	average approx. ratio
			eil51_n150_uncorr_07				
MMASls4	36538	11671	53368	467.00	652.11	11763	0.997
S1/C6	34622	13587	52145	459.00	659.35	10079	0.856/0.857
			dsj1000_n2997_uncorr-similar-weights_03				
MMASls4	758385	62635	590594	19286106	27205708	46480	0.832
MMASls3boost	774523	46497	595519	19290271	26699765	61524	0.871
S1	758408	62612	584276	18705228	26709155	50093	0.622
S5	758364	62656	590515	18750512	26599551	58524	0.931
C6	761602	59418	587164	18750975	26376923	59626	0.876

Table 1. For two instances, we show details of the best solutions (in terms of "objective score") found by different approaches for two instances. For example, MMASls3boost found an outstanding solution for the second instance, however, it is outperformed on average by S5 (0.871 vs 0.931). The shaded cells highlight the best objective scores and best average approximation ratios.

distances. S1's best solution actually has the shortest tour, but the resulting objective score is the second-worst. S1's resampling variant S5 investigates many tours, which can be longer and which can offer different ways of constructing the packing plans. MMASls4 now performs poorly as it lacks hillclimbers to optimise the packing plans. MMASls3boost performs significantly better on average, and even found an oustanding solution once. Interestingly, this solution has the longest travel distance *and* the highest knapsack use among all five shown solutions.

In summary, we can see that exploring longer tours can be very beneficial, if done efficiently. This is exactly what we expect to see in the TTP, as it is the combination of the travelling salesperson problems and the knapsack problem.

5 Concluding remarks

While our MMAS approach is most definitely not "one approach to rule them all", it outperforms existing approaches on instances with up to 250 cities and 2000 items, sometimes by over 10%. It achieves this because it focusses less than existing approaches on good TSP tours, but more on good TTP tours.

We investigated the boosting of solutions in the form of TTP-specific local search. This was effective in general, however, it is too time-consuming on larger instances and thus detrimental to the performance, as it reduces the number of tours the algorithms can consider given the fixed time budget. This brings us back to the general problem. Currently, the TTP's search space seems to be incredibly hard to navigate. We understand that it can be tempting for researchers to focus on large instances using construction heuristics and hill-climbers. However, we suggest to focus on small instances instead, because large performance gains are still possible there as our investigations show. By creating good approximation algorithms that are effective in considering the interaction of the problem components, but that are not necessarily computationally efficient, we should be able to gain additional insights into the actual interaction.

In the future, maybe instance analysis where the influence of the different components is varied may help to understand how the interactions influence algorithm performance. A first step towards this has recently been taken by Nallaperuma et al. [9], who systematically analysed the difficulty of TSP instances for MMAS with different parameter settings.

It is interesting to note that no parameter tuning on the MMAS side of our approaches has been performed. We have made all code and all results publicly available: `http://cs.adelaide.edu.au/~optlog/research/ttp.php`. On this project website, we also have an extended version of this article with additional insights.

Bibliography

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

[2] 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 *Congress on Evolutionary Computation*, pages 1037–1044. IEEE, 2013.

[3] M. R. Bonyadi, Z. Michalewicz, M. R. Przybylek, and A. Wierzbicki. Socially inspired algorithms for the TTP. In *Genetic and Evolutionary Computation Conference*, pages 421–428. ACM, 2014.

[4] S. Chand and M. Wagner. Fast heuristics for the multiple traveling thieves problem. In *Genetic and Evolutionary Computation Conference*. ACM, 2016. Accepted for publication.

[5] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

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

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

[8] Y. Mei, X. Li, and X. Yao. Improving efficiency of heuristics for the large scale traveling thief problem. In *Simulated Evolution and Learning*, volume 8886 of *LNCS*, pages 631–643. Springer, 2014.

[9] 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.

[10] S. Polyakovskiy and F. Neumann. Packing while traveling: Mixed integer programming for a class of nonlinear knapsack problems. In *Integration of AI and OR Techniques in Constraint Programming*, volume 9075 of *LNCS*, pages 330–344. Springer, 2015.

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

[12] T. Stützle and H. H. Hoos. MAX-MIN ant system. *Journal of Future Generation Computer Systems*, 16:889–914, 2000.