An Evolutionary Approach to the Full Optimization of the Traveling Thief Problem

Nuno Lourenço[1(✉)], Francisco B. Pereira[1,2], and Ernesto Costa[1]

[1] Department of Informatics Engineering, CISUC, University of Coimbra, Polo II - Pinhal de Marrocos, 3030 Coimbra, Portugal
{naml,xico,ernesto}@dei.uc.pt
[2] Polytechnic Institute of Coimbra, Quinta da Nora, 3030-199 Coimbra, Portugal

Abstract. Real-World problems usually consist of several different small sub-problems interacting with each other. These interactions promote a relation of interdependence, where the quality of a solution to one sub-problem influences the quality of another partial solution. The Traveling Thief Problem (TTP) is a recent benchmark that results from the combination of the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP). Thus far, existing approaches solve the TTP by fixing one of the components, usually the TSP, and then tackling the KP. We follow in a different direction and propose an Evolutionary Algorithm that addresses both sub-problems at the same time. Experimental results show that solving the TTP as whole creates conditions for discovering solutions with enhanced quality, and that fixing one of the components might compromise the overall results.

Keywords: Evolutionary algorithms · Combinatorial problems · Traveling thief problem

1 Introduction

Heuristic problem solving, e.g., based on Evolutionary Algorithms (EA), is a successful approach for solving problems for which an analytical solution does not exist, or, when it does, it is computationally intractable. To assess the performance of heuristic-based EAs, researchers usually rely on benchmark problems combined with a sound statistical analysis. Choosing good benchmarks is thus critical, and, over time, discussing which ones should be used has gained relevance in the EA community [13].

Many real world problems comprise a non-linear combination of several sub-problems. The existing interactions between partial solutions impact the quality of the global solution, thus complicating the task of optimization algorithms. Unfortunately, benchmarks adopted by EA researchers tend to ignore this question and the impact of non-linear interactions between problem components is usually outside the discussion of algorithmic effectiveness. The Traveling Thief Problem (TTP) is a recent benchmark [2] that considers the interdependence

F. Chicano et al. (Eds.): EvoCOP 2016, LNCS 9595, pp. 34–45, 2016.
DOI: 10.1007/978-3-319-30698-8_3

between two well known problems: the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP). The underlying idea behind the TTP is to maximize the profit of a thief that is traveling through a certain number cities stealing items. The thief uses a knapsack with a limited capacity and pays a rent for it, that depends on the time needed to visit all the cities. The interdependence of the TTP emerges from the fact that the speed of the thief depends (non-linearly) on the weight of the items picked so far.

There are a few heuristic approaches to tackle the TTP [3,5,11]. However, most of them seek for solutions by fixing one of the components, while solving the other. The typical approach is to initially set the shortest TSP route for the cities comprising the problem and then solve the remaining KP component. By doing this, a bias towards solutions with small tour lengths is clearly created. Also, it is not guaranteed that the best solutions for the TTP can be found, as large portions of the search space are ignored. The interactions between the two sub-problems ensue that there is a tradeoff between the distance the thief travels and the value that he is able to gather. Since the weights of the items affect the speed of the thief, it is likely that the thief should sometimes slightly increase the tour length, providing that is allows him to pick an heavy, but valuable, item near the end of the tour.

In this paper we present an evolutionary unbiased approach for the TTP, that seeks for complete solutions by simultaneously considering the two sub-problems and the existing interdependence between them. In concrete we rely on an EA where each individual has a tour and a packing plan (items that should be picked at each city). The variation operators modify both components, and a packing heuristic helps creating good packing plans for each individual. The performance of the approach is tested in some TTP benchmarks instances proposed in [11]. Experimental results confirm that it is important to simultaneously take into account both components of the problem.

The remainder of the paper is organized as follows: in Sect. 2 we describe the TTP, whereas Sect. 3 reviews some recent approaches to solve the this problem. Section 4 details our approach to the problem. In Sect. 5 we present and detail the experimental results obtained. Finally, in Sect. 6 we gather the main conclusions and point towards future work.

2 The Traveling Thief Problem

The TTP is a recent benchmark that was created to mimic the interdependence between problems that occur in real-world applications [2]. It is defined as follows: consider a set of cities $N = 1, ..., n$ and a set of items $M = 1, ..., m$, which are distributed among the cities. The distance d_{ij}, with $i, j \in N$ is known. Each city i, except the first one, has a subset of items $M_i = 1, ..., m_i$. Each item k placed in the city i is described by its profit p_{ik} and weight w_{ik}. The thief departs from the first city, visits each city exactly once, and returns to the starting point. Any item may be collected at any city, as long as the total weight of the items in the knapsack do not exceed its capacity W. Additionally the thief has to pay

a rent R for the use of the knapsack for each time unit. When the thief does not have any item in the knapsack it can travel at maximum speed, υ_{max}, whilst when the knapsack is full it travels at a minimum speed υ_{min}. The goal is to find a tour and a packing plan that results in the maximum profit for the thief. Let y_{ik} be a binary variable that is 1 if the item k is picked at city i. The objective function for a given tour $\Pi = (x_1, ...x_n, x_1), x_i \in N$ and a given packing plan $P = (y_{21}, ..., y_{nm_i})$ is:

$$\begin{aligned} Z(\Pi, P) &= \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{ik} y_{ik} - R * \left(\frac{d_{x_n x_1}}{\upsilon_{max} - \nu W_{x_n}} + \sum_{i=1}^{n-1} \frac{d_{x_i x_{i+1}}}{\upsilon_{max} - \nu W_{x_i}}\right) \\ \text{s.t.} \sum_{i=1}^{n} \sum_{k=1}^{m_i} w_{ik} y_{ik} &\leq W \end{aligned} \tag{1}$$

where $\nu = \frac{\upsilon_{max} - \upsilon_{min}}{W}$, and W_{xi} is the total weight of collected items that the thief has at city i. The first term of the equation represents the total profit of all picked items, whilst the second term is the total cost of the thief's trip.

Consider the example with 4 cities and corresponding distances depicted in Fig. 1, which was adapted from [11]. Every city, with the exception of the first one, has a set of available items that the thief might choose to pack. As an example, city 2 was two items: $I_{21} : \{profit_{21} = 20, weight_{21} = 2\}$ and $I_{22} : \{profit_{22} = 30, weight_{22} = 3\}$. Specifying that the renting rate $R = 1$, $\upsilon_{max} = 1$, $\upsilon_{min} = 0.1$, and that the maximum capacity of the of the sack is $W = 3$ completely defines the instance.

A possible solution for this instances defines a tour $\Pi : (1, 2, 4, 3, 1)$ and a packing plan $P : (I_{21} = 0, I_{22} = 0, I_{31} = 0, I_{32} = 1, I_{33} = 1, I_{41} = 0)$. In this solution, the thief starts in city 1 and moves to cities 2 and 4 without any items. It then moves to city 3, and, at this moment, the cost of the solution is 15 $(5+6+4)$. In city 3, the thief picks up the items I_{32} and I_{33}, which gives a total profit of 80. When returning to city 1 to complete the tour, the knapsack has a weight of 2, thus reducing the velocity of the thief, corresponding to a traveling cost of 15. All in all, the final fitness value is $Z(\Pi, P) = 80 - 15 - 15 = 50$.

3 Review of the Literature

The TTP problem is a benchmark to study the interdependence between different sub-problems and it was proposed by Bonyadi *et al.* in [2]. In concrete, the TTP results from the combination of the TSP and KP. The authors propose a method to create instances of the TTP, so that researchers can compare the results of different approaches. Additionally, the work shows a simple experiment, which studies how the two problems are connected. In concrete they created a simple instance of the TTP, and separately solved the TSP and KP parts to optimality. Then the best solutions found for each sub-problem are combined, and it is shown that this combination does not correspond to the best solution for the TTP.

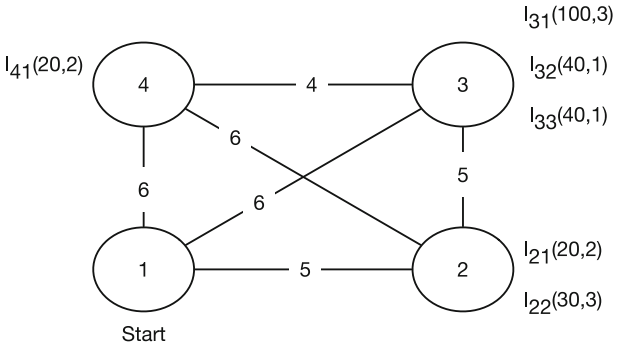

Fig. 1. Example of a TTP instance with 4 cities and 6 items. Adapted from [11]

Later, Polyakovskiy *et al.* in [11] proposes an ensemble of benchmark instances and heuristic methods to tackle the TTP problem. Regarding the instances, they considered the TSPLIB[1] [12] dataset as a starting point. For the KP part of the TTP they created several sets of items following the recommendations of [8]. To generate the TTP instances they considered different combinations of the TSPLIB instances with the KP dataset[2]. Although the creation of a comprehensive dataset was the main goal of the work, it also suggests some simple heuristics to solve the TTP problem. Although the heuristics consider the two sub-problems independently. Firstly they solve the TSP problem to obtain a good tour, disregarding the KP part of the problem. Once a good tour is found, it is kept fixed for the remaining part of the optimization. Then a local search algorithm is used to create a packing plan that achieves a good objective value for the TTP. This work was extended by Faulkner *et al.* [5] and a set of new heuristics were proposed. However the underlying idea is the same: find a good solution for one of the components and then fix it. Although, fixing one of the components of the TTP and neglecting the dependence that exist between the two sub-problems, might prevent the discovery of best solution for the problem.

The work of Mei *et al.* [9] focus on solving large TTP instances with low resource consuming heuristics. They analyze the computational complexity of several different algorithms, and propose a new two-stage local search procedure to create packing plans. The main idea of the algorithm is to first prioritize the insertion of items. Then, as more items are added they check how the addition of a item worsen the thief's speed. They compute some thresholds to decide whether it is worth to add item or not. They incorporate the proposed heuristic into a memetic algorithm, where several initial tour solutions are generated and optimized by the Lin-Kerninghan heuristic (LK) [1]. The algorithm iteratively combines the different solutions to create new packing plans. The approach was

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
[2] The TTP instances are available at: http://cs.adelaide.edu.au/~optlog/CEC2014COMP_InstancesNew/.

applied to different instances of the TTP, and they show that it was able to outperform previous heuristics when used to solve large instances.

The work presented in [3] puts forward two methods to solve the TTP problem. The first is called Density-based Heuristic (DH). The DH starts by using the LK heuristic to create a tour as short as possible for the TSP. Then it computes the profit that the thief would get if only one item was picked, after completing the tour. They do this for all available items. After it iteratively adds the items to the packing plan. An item is only packed if it does not worsen the overall profit of the TTP.

The second method attempts to solve the TTP by decomposing it in two sub-problems and trying two solve them in parallel. Each component communicates with each other, and from time to time the algorithm tries to combine the solutions of the sub-problems to create an overall approximated solution for the TTP.

4 Evolutionary Approach

The approaches presented in the previous section are the state-of-art for the TTP problem. Despite their relative success there is margin for progress, mostly because they fix one of the TTP's sub-problems. Our approach tries to overcome this by tackling both sub-problems at the same time. For each new tour generated by the optimization algorithm, the packing plan is rearranged.

Our proposal relies on an Evolutionary Algorithm (EA) to search for good solutions for the TTP. The underlying idea behind EA is the simulation of evolution by natural selection of a population of artificial individuals via application of selection, variation operators, and reproduction. These components are guided by a fitness function that evaluates each individual, measuring the quality of the solution it represents. In our approach each individual is composed by a tour and by a packing plan. The tour is a permutation of integers that represents the order in which each city should be visited by the thief. Note that each tour starts and ends in the first city of the instance (city 0). The packing plan is a binary string, that indicates if an item should be picked at a certain city. Fig. 2 shows a simple example of a solution for a TTP instance with 5 cities and 2 items per city.

The EA starts by randomly generating a population of tours for the TTP. Then, for each generated tour, we rely on the packing heuristic described in [11] to create a valid packing plan for the specific tour of the individual. The

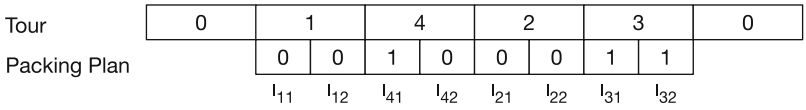

Fig. 2. Example of an EA individual for a TTP instance with 5 cities and 2 items per city

heuristic calculates a score that estimates how profitable an item is, according to a certain tour. Consider that an item I_{x_ik} can be picked at a city x_i, and that d_{x_i} and t_{x_ik} are the total traveling distance and the total traveling time with I_{x_ik} being carried until the end of the tour, respectively. The score of each item is computed has follows:

$$score_{x_ik} = p_{x_ik} - R * t_{x_ik} \tag{2}$$

where

$$t_{x_ik} = \frac{d_{x_i}}{\upsilon_{max} - \nu w_{x_ik}} \tag{3}$$

The $score_{x_ik}$ is the total profit that the thief would obtain if only the item I_{x_ik} is picked during the whole tour. Using this we iteratively select the items that have the highest score. We stop when there are no more items to add, or the knapsack is full. Then each solution is evaluated using the objective function defined by in Eq. 1.

After being evaluated, each solution is improved by a straightforward local search procedure (Algorithm 1), which tries to refine the packing plan over a limited number of iterations. In this procedure we randomly select an item in the current plan and flip its status: if the item is in the knapsack it is removed, else it is added. If this flip results in a better solution to the problem, we keep the new packing plan, else, it is discard. We repeat this until an improvement is found or the maximum number of flips is reached.

Tournament selection chooses the individuals that undergo reproduction. We rely on two variation operators to create new solutions, by modifying existing ones. The first is an adaptation of the Partially Mapped Crossover (PMX) [6]. The second is the Inversion mutation operator for permutations [4]. The recombination operator creates offspring by exchanging the information about the tours as well as the items that the thief picked at each city. When switching cities in the tour, the operators also move the items that were picked at that city (Fig. 3).

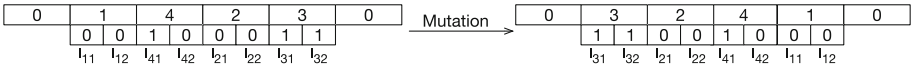

Fig. 3. Example of the application of the mutation operation

It is known that the PMX operator alone is not particularly effective, partially because it does not transmit common edges between the parents to the offspring. However using it together with inversion mutation yielded better results than using, for instance, edge recombination. This is in accordance with the observations made by [7]: using inversion mutation (2-opt) in solutions that have been already improved by other methods can result in poorer overall results than using inversion mutation on worst solutions.

Having a new tour, the packing plan is corrected/improved to take into account the new routes that the thief has to travel. We rely again on the packing

heuristic described above to select the most promising items and/or to remove the items that are in excess. Subsequently the new solutions are evaluated, and the packing plan undergoes a new iteration of the local search. Algorithm 2 outlines the proposal described.

Algorithm 1. Local Search

```
function LOCALIMPROVEMENT(ind)
    tempInd ← ind
    i ← 0
    while i < MaximumLocalImprovementIterations do
        posToInvert ← randomInteger(0, length(tempInd.packingPlan))
        if tempInd.packingPlan[posToInvert] = 1 then
            tempInd.packingPlan[posToInvert] ← 0
        else
            tempInd.packingPlan[posToInvert] ← 1
        end if
        evaluate tempInd using Eq. 1
        if tempInd.Fitness > ind.Fitness then
            ind ← tempInd
            return ind
        end if
    end while
    return ind
end function
```

5 Experiments

In this section we report the results attained by our approach, and compare it with an implementation of the DH heuristic described in [3]. We conducted the experimental study using instances with a number of cities ranging from 51 to 100. Each instance is composed by a group of cities, a set of items, the maximum (v_{max}) and minimum (v_{min}) velocity at which the thief can travel, the maximum capacity W of the knapsack, the renting ratio R, and the number of items available per city. In the instances considered every city (except the first) has the same number of items, which are either 1 or 3 items. For this study we selected instances whose items have their profit strongly correlated to their weight. The higher the correlation between weight and profit, the more time consuming is the KP to solve [8,10].

The experimental parameters for the EA are described in Table 1. The algorithm is composed by 100 individuals that are evolved for $2.5 * 10^5$ function evaluations. Contrary to previous approaches to this problem, we use the number of evaluations instead of time as stop criterion, as this simplifies the reproducibility of results. We performed 30 independent runs of the EA on each

Algorithm 2. Evolutionary Algorithm for the TTP

```
create population Pop randomly
evaluations ← 0
for all ind ∈ Pop do
    ind.PackingPlan ← packingHeuristic(ind.Tour)
    evaluate ind using Eq. 1
    ind ← localImprovement(ind)
end for
while evaluations < maximumNumberEvaluations do
    Parents ← parentSelection(Pop)
    Offspring ← AdaptedPMX(RecombinationRate, Parents)
    Offspring ← InversionMutation(MutationRate, Offspring)
    for all ind ∈ Offspring do
        ind.PackingPlan ← packingHeuristic(ind.Tour)
        evaluate ind using Eq. 1
        ind ← localImprovement(ind)
    end for
    Pop ← selectSurvivors(Pop, Offspring)
end while
return best Solution discovered
```

instance considered in this study. We do not present a statistical analysis comparing the results of the EA and the DH heuristic, since for this second method we only have access to the best solutions found.

5.1 Results

A summary of the optimization results is presented in Table 2. The first column identifies the instance, with the number indicating how many cities the thief has to visit (e.g., eil51 has 51 cities). The second column (*Items per City*) shows the

Table 1. Parameter Settings

Parameter	Value
Runs	30
Population Size	100
Evaluations	$2.5 * 10^5$
Parent Selection	Tournament with size 5
Replacement	Generational
Recombination Operator	Partially Mapped Crossover
Mutation Operator	Inversion
Recombination Rate	0.9
Mutation Rate	0.1
Local Search Iterations	#cities * #items

Table 2. Results obtained using the Evolutionary Approach proposed in this work and the DH heuristic

Instance	Items Per City	Algorithms		
		DH	EA	
			Best	MBF
eil51	1	2591.95	**4706.54**	4088.90 (± 309.95)
	3	9163.66	**10115.55**	8247.07 (± 1067.26)
eil76	1	4264.13	**5506.92**	4502.23 (± 737.26)
	3	10583.40	**12040.77**	8871.19 (± 1805.64)
kroA100	1	**7095.97**	6442.41	4392.21 (± 1035.68)
	3	**25923.00**	22325.30	17989.43 (± 2757.97)

number of items that are available in each city. The remaining columns contain the results obtained by the DH heuristics and by the EA proposed in this paper. The outcomes of the EA include the fitness of the best solution found (column *Best*) and the average and standard deviation of the best solutions found in the 30 runs (Mean Best Fitness - column *MBF*).

An overview of the results reveals that the EA is able to find promising solutions and is competitive with existing state-of-the-art approaches for the TTP. In concrete, it discovered clearly better solutions than DH in 4 of the selected instances. For the two larger instances (kroA100 with 1 and 3 items), it found solutions whose quality is not worse than 9 % and 13 %, respectively. The difficulties in the larger instances are not unexpected and are probably related to the increasing size of the search space. The DH heuristic breaks the problems in two independent components: it firstly finds an high quality tour and fixes it. Then it tries to create the best packing plan that fits into that tour. By doing this, DH narrows the search space to a region of solutions that are based on (near) optimal tours. Although this simplifies the task of creating a packing plan, there is no guarantee that the best solution for the compete problem is found. On the contrary, the EA is performing its search in the space of all possible TTP solutions. Increasing the number of cities and/or the number of items results in larger search spaces. Since the number of evaluations is kept fixed for all instances considered in the study, it is likely that there is degradation in the performance of the algorithm, since it does not have time to effectively sample the search space and discover the region(s) where the promising solutions are.

The results from the last column of Table 2 show that the number of items available per city impact the variance of the results: for all considered instances, more items lead to a higher standard deviation. This is another piece of evidence that confirms the decreased performance of the EA when dealing with larger instances.

5.2 Solution Analysis

Our approach is based on the argument that solving one of the sub-problems (namely the TSP) and then solving the other provides compromises the quality

of the global solution and prevents a meaningful exploration of the search space. Results presented in the previous section revealed that solving the TTP as a whole is advantageous and creates conditions for the discovery of enhanced solutions. Here we analyse some features of the solutions discovered by the EA to gain a better insight into the optimization behavior of this approach. Specifically, we are interested in comparing the tour that provides the best fitness for the TTP, with the best tour for the corresponding TSP problem. If they are different, it supports the claim that non-linear components of an optimization problem should not be solved independently.

The comparison of the tours is performed as follows. We take the permutation set that represent the best TSP tour for each instance and compute a set G_1 with the number of transitions. A transition is a tuple (x_i, x_{i+1}), that indicates a direct move from city x_i to city x_{i+1}. The same procedure is followed to create the set G_2 with all transitions from the best tour found by the EA. Finally we created a set $G_3 = G_1 \cap G_2$, containing all transitions that are shared between G_1 and G_2. Table 3 present the percentage of transitions that are different between the best TSP tour and the best TTP tour found by the EA. The results show that the TTP tours tend to be substantially different from the TSP tours. The percentage of different transitions ranges from 31.4 % to 71.1 %. Moreover, in all cases but one, the difference exceeds 50 %. These values indicate that the tours are substantially different, and that fixing the tour might compromise the discovery of effective solutions for the TTP.

Table 4 compares the distances of the best TTP tours with the optimal distance of the TSP tour. The column *TSP Optimal Distance* represents the best known solution for the TSP, whereas column *TTP Distance* represents the distance of the tour associated with the best TTP solution. A brief perusal show that the TTP tours are longer, suggesting that sometimes it is worth to delay the pick of a valuable item that is in a particular city. This might result in slightly longer tours, but in lower carrying cost.

The results presented in this section help to justify why it is important to solve the TTP problem as a whole. Solving one of the sub-problems and then solving the other compromises the interdependence between components, leading to poorer global results. The study presented here is a first step towards a better

Table 3. Percentage of transitions that are different between the TSP best known tour and the tour belonging to the best TTP solution found by the EA

Instance	Items Per City	#Different Edges (in %)
eil51	1	58.8
	3	31.4
eil76	1	71.1
	3	51.3
kroA100	1	69.0
	3	52.0

Table 4. Difference between the distance of the TSP tours and the TTP tours.

Instance	TSP Optimal Distance	Items Per City	TTP Distance	Absolute Difference (in %)
eil51	426	1	495	16.2
		3	470	10.3
eil76	538	1	695	29.2
		3	641	19.1
kroA100	21282	1	26691	25.4
		3	24847	16.8

understanding of the non-linear interconnections occurring in the TTP. It still has some limitations, as only a subset of instances were considered and the analysis focuses just on the influence of relying in fixed TSP optimal tours. A logical next step if to study the impact of fixing the set of items, while trying to evolve a tour.

6 Conclusions and Future Work

In this paper we proposed an unbiased approach to the Traveling Thief Problem (TTP). The TTP is a new benchmark that results form the combination of two well-known problems, the Traveling Salesman Problem and the Knapsack Problem. The benchmark is created in such a way that it promotes a relation of interdependence between the two sub-problems. This means that the solution of one sub-problem influences the quality of the solutions for the other sub-problem. Based on this, solving one sub-problem alone, even to optimality, might result in a inferior performance, when considering thew global optimization scenario.

Almost all of the existing approaches described in the literature solve the TTP problem by fixing one of its sub-problems, thus creating a bias towards some solutions. These approaches create a reasonably good TSP tour (most of the times they use the optimal solution), fix it, and try to find the set of items the give the maximum profit for that tour.

Our approach follows a different direction. We proposed an Evolutionary Algorithm (EA) that solves the TTP as a whole. The EA evolves a population of individual solutions, where each solution is a tour and a packing plan. During the evolutionary process both components are modified, keeping the synergy that exists between the sub-problems of the TTP. The results obtained confirm the potential of the approach, as the EA was able to find good quality solutions in most of the instances used. We also performed an analysis on the structure of the best solutions found. Specifically, we compared the optimal TSP tours with the tours proposed by the EA to the corresponding TTP problem and verified that, in all but one case, they differ in more than 50 % of the transitions. This is an important result for it shows that fixing one of the sub-problems might compromise the discovery of global effective solutions.

More experiments are still needed to fully comprehend the relation between the two problems. In the near future we will focus our efforts in understanding how the distribution of items through the cities affects the quality of the

results. Also, we intended to address the scalability issues, in order to solve larger instances. Finally, we plan to develop alternative representations that create a more symbiotic relationship between the two sub-problems.

Acknowledgments. This work was partially supported by Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant SFRH/BD/79649/2011.

References

1. Applegate, D., Cook, W., Rohe, A.: Chained lin-kernighan for large traveling salesman problems. INFORMS J. Comput. **15**(1), 82–92 (2003)
2. Bonyadi, M.R., Michalewicz, Z., Barone, L.: The travelling thief problem: the first step in the transition from theoretical problems to realistic problems. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 1037–1044. IEEE (2013)
3. Bonyadi, M.R., Michalewicz, Z., Roman Przybyoek, M., Wierzbicki, A.: Socially inspired algorithms for the travelling thief problem. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation. pp. 421–428. ACM (2014)
4. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
5. Faulkner, H., Polyakovskiy, S., Schultz, T., Wagner, M.: Approximate approaches to the traveling thief problem. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, pp. 385–392. ACM (2015)
6. Goldberg, D.E., Lingle, R.: Alleles, loci, and the traveling salesman problem. In: Proceedings of the first International Conference on Genetic Algorithms and their Applications pp. 154–159 (1985)
7. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: a case study in local optimization. Local Search Comb. Optim. **1**, 215–310 (1997)
8. Martello, S., Pisinger, D., Toth, P.: Dynamic programming and strong bounds for the 0–1 knapsack problem. Manage. Sci. **45**(3), 414–424 (1999)
9. Mei, Y., Li, X., Yao, X.: Improving efficiency of heuristics for the large scale traveling thief problem. In: Dick, G., Browne, W.N., Whigham, P., Zhang, M., Bui, L.T., Ishibuchi, H., Jin, Y., Li, X., Shi, Y., Singh, P., Tan, K.C., Tang, K. (eds.) SEAL 2014. LNCS, vol. 8886, pp. 631–643. Springer, Heidelberg (2014)
10. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
11. Polyakovskiy, S., Bonyadi, M.R., Wagner, M., Michalewicz, Z., Neumann, F.: A comprehensive benchmark set and heuristics for the traveling thief problem. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, pp. 477–484. ACM (2014)
12. Reinelt, G.: Tspliba traveling salesman problem library. ORSA J. Comput. **3**(4), 376–384 (1991)
13. White, D.R., McDermott, J., Castelli, M., Manzoni, L., Goldman, B.W., Kronberger, G., Jaśkowski, W., OReilly, U.M., Luke, S.: Better gp benchmarks: community survey results and proposals. Genet. Program. Evolvable Mach. **14**(1), 3–29 (2013)