Solving the Bi-Objective Traveling Thief Problem with Multi-Objective Evolutionary Algorithms

Julian Blank[1], Kalyanmoy Deb[2], and Sanaz Mostaghim[3]

[1] Otto-von-Guericke University, 39106 Magdeburg, Germany,
julian.blank@gmx.de,
WWW home page: http://www.research-blank.de

[2] Michigan State University, East Lansing MI 48824, USA,
kdeb@egr.msu.edu,
WWW home page: http://www.egr.msu.edu/~kdeb/

[3] Otto-von-Guericke University, 39106 Magdeburg, Germany,
sanaz.mostaghim@ovgu.de,
WWW home page: http://www.is.ovgu.de/Team/Sanaz+Mostaghim.html

Abstract. This publication investigates characteristics of and algorithms for the quite new and complex Bi-Objective Traveling Thief Problem, where the well-known Traveling Salesman Problem and Binary Knapsack Problem interact. The interdependence of these two components builds an interwoven system where solving one subproblem separately does not solve the overall problem. The objective space of the Bi-Objective Traveling Thief Problem has through the interaction of two discrete subproblems some interesting properties which are investigated. We propose different kind of algorithms to solve the Bi-Objective Traveling Thief Problem. The first proposed deterministic algorithm picks items on tours calculated by a Traveling Salesman Problem Solver greedily. As an extension, the greedy strategy is substituted by a Knapsack Problem Solver and the resulting Pareto front is locally optimized. These methods serve as a references for the performance of multi-objective evolutionary algorithms. Additional experiments on evolutionary factory and recombination operators are presented. The obtained results provide insights into principles of an exemplary bi-objective interwoven system and new starting points for ongoing research.

Keywords: Traveling Thief Problem, Traveling Salesman Problem, Knapsack Problem, Interwoven Systems, Multi-objective Optimization, Discrete optimization, Combinatoric problems.

1 Introduction

People are facing real-world problems with dependencies and interwovenness every day. Often researchers divide complex problems into various well investigated subproblems to solve them independently. After applying state of the art methods for the subproblems, partial solutions are combined to a solution for the complex problem. However, considering an interwoven problem the optimal solutions for the subproblems do not build an optimal solution for the complex problem because of the interaction. In order to provide an interwoven problem with real-world characteristics, this publication presents the Bi-Objective Traveling Thief Problem (TTP), where the well-known Traveling Salesman Problem (TSP) and Binary Knapsack Problem (KNP) interact.

The focus of this publication is to provide a starting point for doing research on the more complex bi-objective problem by showing characteristics of the objective space and proposing deterministic as well as evolutionary multi-objective algorithms. Research on the Bi-Objective Traveling Thief Problem will give insights into how to handle interwoven problems with multiple objectives and therefore how to solve problems with real-world characteristics.

The outline of this work is as follows: Section 2 explains the TTP itself and its two interweaving components. Furthermore, the interdependencies are described and an example scenario is provided. Section 3 presents related work, while Section 4 describes properties of the input variables and characteristics of the objective space. Moreover, the proposed algorithms are explained in Section 5 and the obtained results are presented in Section 6. Finally, a conclusion and further considerations are provided in Section 7.

2 Problem Definition

2.1 Traveling Thief Problem

The TTP is a combinatorial optimization problem that consists of two interweaving subproblems, namely TSP and KNP [1]. After explaining the components separately, the interaction in the TTP of these two subproblems is shown.

In the TSP [2] a salesman has to visit n cities. The distances are given by a map represented as a distance matrix $D^{n\times n} = (d_{i,j})$ with $i,j \in \{1,..,n\}$ whereby $d_{i,j} = d_{j,i}$. The salesman has to visit each city once and the result is a permutation vector $\pi = (\pi_1, \pi_2, ..., \pi_n) \in \mathbb{P}^n$, where π_i is the i-th city visited by the salesman. Often $\pi_1 = 1$ is required which means the starting city is fixed.

$$\begin{aligned} &min \quad f(\pi) && (1)\\ &f(\pi) = \sum_{i=1}^{n-1} d_{\pi_i,\pi_{i+1}} \; + \; d_{\pi_n,\pi_1} \\ &s.t. \quad \pi \in \mathbb{P}^n \text{ with } \pi_1 = 1 \\ & && (2) \end{aligned}$$

For the KNP [3] a knapsack has to be filled with items by considering the maximal capacity Q. Each item j has a value $b_j \geq 0$ and a weight $w_j \geq 0$ where

$j \in \{1, .., m\}$. The binary decision vector $z = (z_1, .., z_m) \in \mathbb{B}^m$ defines, if an item is picked or not. The aim is to maximize the profit $g(z)$:

$$\begin{aligned} &max \quad g(z) \\ &g(z) = \sum_{j=1}^{m} z_j\, b_j \\ &\text{s.t.} \quad \sum_{j=1}^{m} z_j\, w_j \leq Q \end{aligned} \tag{3}$$

The TTP combines the above defined subproblems and let them interact together. The interdependency relation between the components of the Bi-Objective TTP is visualized in Figure 2.1. The tour π is only part of the TSP component. Therefore, the profit $g(z)$ remains the same when π is modified. However, the packing plan z is part of both components and both objectives are influenced by z. Since both components need the packing plan z as a parameter, it is hard to solve the problem. In fact, such problems are called interwoven systems as the solution of one subproblem highly depends on the solution of the other subproblems.

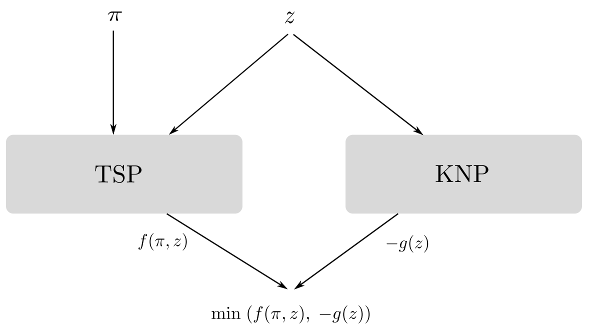


Fig. 1. Interdependence of the Bi-Objective Traveling Thief Problem

Items are assigned to each city on the map through the assignment matrix $A^{m \times n} = (a_{i,j})$. Each item is exactly assigned to one city, but one city can have multiple items. The thief can collect items from each city he is visiting. The items are stored in a knapsack with the maximum capacity Q carried by him. As interaction between the subproblems the velocity $v(q)$ depends on the current knapsack weight $q = w(\pi_i)$. When the thief picks an item, the weight of the knapsack increases and the velocity of the thief decreases. The velocity is always in a specific range $v = [v_{min}, v_{max}]$ and could not be negative for a feasible solution. Whenever the knapsack is heavier than the maximum weight Q, the capacity constraint is violated. The variables π and z as well as the item information w and b are defined analogously to TSP and KNP. The two objective functions $f(\pi, z)$ and $-g(z)$ have to be minimized. The whole problem is defined as follows:

Symbol	Description
$\pi \in \mathbb{P}^n$	Tour as permutation vector with length n. π_i represents the i-th city visited by the thief.
$z \in \mathbb{B}^m$	Picking plan as binary vector with length m. $z_i = 1$ means the i-th item is picked up.
$D^{n \times n} = (d_{i,j})$	Distance matrix: $d_{i,j}$ represents the distance between city i and city j whereby $d_{i,j} = d_{j,i}$.
$w \in \mathbb{Z}^m$	Item's weight vector: w_i is the weight of item i.
$b \in \mathbb{Z}^m$	Item's value vector: b_i is the value of item i.
$v_{min} \in \mathbb{R}^+$	Minimum velocity
$v_{max} \in \mathbb{R}^+$	Maximum velocity
$Q \in \mathbb{Z}^+$	Maximum knapsack capacity
$A^{m \times n} = (a_{i,j})$	Assignment of items to cities where $a_{i,j} \in \{0, 1\}$ and $\forall j \in \{1, .., m\} : \sum_{i=1}^{n} a_{j,i} = 1$. (each item is assigned to exactly one city)

$$
\begin{aligned}
&\min_{\pi, z} \; F\,(\pi, z) && (4)\\
F(\pi, z) &= (f(\pi, z), -g(z))\\
f(\pi, z) &= \sum_{i=1}^{n-1} \frac{d_{\pi_i, \pi_{i+1}}}{v(w(\pi_i))} \; + \; \frac{d_{\pi_n, \pi_1}}{v(w(\pi_n))}\\
g(z) &= \sum_{j=1}^{m} z_j \, b_j\\
v(q) &= v_{max} - \frac{q}{Q} \cdot (v_{max} - v_{min})\\
w(\pi_i) &= \sum_{k=1}^{i} \sum_{j=1}^{m} z_j \, w_j \, a_{j, \pi_i}\\
\text{s.t.} \quad & \sum_{j=1}^{m} z_j \, w_j \leq Q\\
& \pi \in \mathbb{P}^n \text{ with } \pi_1 = 1
\end{aligned}
$$

Note, that the item value drop effect mentioned in [4] is neglected in our considerations. An example scenario is provided in Section 2.2. Given a small map, a tour π and a packing plan z the traveling time and the profit for the thief are calculated and the interdependency effect is shown. Also, the Pareto front for the given example is presented tabularly.

2.2 Example Scenario

The thief travels on a map represented as a graph shown in Figure 2. He starts at city 1 and has to visit city $2, 3, 4$ exactly once and return to city 1. In this example each city provides one item and the thief could decide to steal item I_j or not.

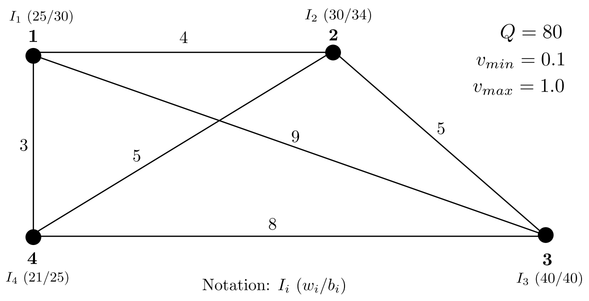


Fig. 2. An Example Scenario for the Traveling Thief Problem

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

In order to understand how the objectives are calculated, an example hand calculation for the tour $\pi = [1, 3, 2, 4]$ and the packing plan $z = [0, 1, 0, 1]$ is done as follows: The thief starts with the maximum velocity, because the knapsack is empty. He begins his tour at city *1* and picks no item there.

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

Table 1. Hand calculations on the Example Scenario for the tour $\pi = [1, 3, 2, 4]$ and picking plan $z = [0, 1, 0, 1]$

For an empty knapsack $w(1) = 0$ the velocity is $v(0) = v_{max} = 1.0$. The distance from city 1 to city 3 is 9.0 and the thief needs 9.0 time units. At city 3 the thief will not pick an item and continue to travel to city 2 with $w(3) = 0$ and therefore with v_{max} in additional 5.0 time units. Here he picks item I_2 with $w_2 = 30$ and the current weight becomes $w(2) = 30$, which means the velocity will be reduced to $v(30) = 1.0 - (\frac{30}{80}) \cdot 0.9 = 0.6625$. For traveling from city 2 to city 4 the thief needs the distance divided by the current velocity $\frac{5.0}{0.6625} \approx 7.5472$. At city 4 he picks I_4 with $w_4 = 21$ and the current knapsack weight increases to $w(4) = 30+21 = 51$. For this reason the velocity decreases to $v(51) = 1.0-(\frac{51}{80}) \cdot 0.9 = 0.42625$. For returning to city 1 the thief needs according to this current speed $\frac{3.0}{0.42625} \approx 7.0381$ time units. Finally, we sum up the time for traveling from each city to the next $\sum_{k=1}^{n} t_{\pi_k,\pi_{k+1}} = 9+5+7.5472+7.0381 = 28.5853$ to calculate the whole traveling time.

The final profit is calculated by summing up the values of all items which is $34 + 25 = 59$. So the variable (π, z) with $\pi = [1, 3, 2, 4]$ and $z = [0, 1, 0, 1]$ is mapped to the point $(28.5853, -59.0)$ in the objective space.

The Pareto front contains 10 solutions and our hand calculation is printed bold in Table 2.

π	z	$f(\pi, z)$	-$g(z)$
$[1, 2, 3, 4]$	$[0, 0, 0, 0]$	20.0	0.0
$[1, 4, 3, 2]$	$[0, 0, 0, 0]$	20.0	0.0
$[1, 2, 3, 4]$	$[0, 0, 0, 1]$	20.93	-25.0
$[1, 4, 3, 2]$	$[0, 1, 0, 0]$	22.04	-34.0
$[1, 4, 3, 2]$	$[0, 0, 1, 0]$	27.36	-40.0
$\mathbf{[1, 3, 2, 4]}$	$\mathbf{[0, 1, 0, 1]}$	**28.59**	**-59.0**
$[1, 4, 3, 2]$	$[1, 1, 0, 0]$	32.75	-64.0
$[1, 2, 3, 4]$	$[0, 0, 1, 1]$	33.11	-65.0
$[1, 4, 3, 2]$	$[0, 1, 1, 0]$	38.91	-74.0
$[1, 3, 2, 4]$	$[1, 1, 0, 1]$	53.28	-89.0

Table 2. Pareto front of the Example Scenario.

3 Related Work

The TTP was introduced by Bonyadi [4]. He postulated the necessity of research on interwoven problems and provided the mathematical formulation of the TTP. Also, the single and bi-objective version of the problem was introduced. Bonyadi laid the foundation for further research. Afterwards, a new large-scale benchmark was proposed [5]. The benchmark is based on the TSPLIB [6] instances which are available for TSP. Since the TTP also needs the KNP part, items with weight and profit attributes are added to the cities. Three different weight-value correlations and ten different capacity categories are considered. Instances with

1, 3, 5 and 10 items per city are provided. All this together builds a benchmark set with 9,720 different instances.

Later, most of the publications aimed to solve these single-objective benchmark problems by using different types of algorithms: Heuristic Based [5] [7] [8] [9] [10], Local Search [5], Coevolution [8] [11], Evolutionary Algorithm [12] [11] [13], Ant Colony Optimization [9].

However, so far less research is done on the Bi-Objective TTP where the solution space has more than one dimension and pareto-optimal solutions have to be considered.

4 Characteristics of the Bi-Objective Traveling Thief Problem

In the following characteristics of the TTP are investigated. Since our example scenario considers only a few cities and items, the optimal Pareto front can be determined by solving the problem exhaustively. Therefore, inferences about problem specific characteristics can be drawn and finally generalized. The objective space of our example scenario colored by tours and annotated with characteristics is shown in Figure 3.

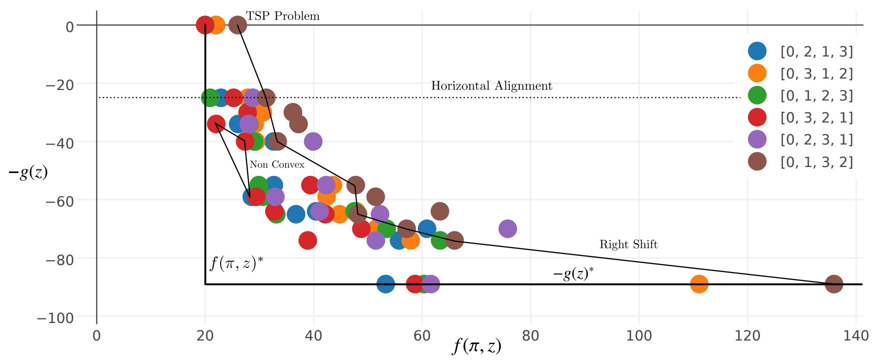


Fig. 3. Objective Space of the Example Scenario colored by Tour

Symmetry of π

For the TTP it is considered that the thief travels on a symmetric map. Whenever the knapsack is empty $z = z^{empty} = \sum_{i=1}^{m} z_i = 0$, the symmetry of π holds. Let us assume we have the tour π and the symmetric tour $sym(\pi)$, then

$$f(\pi, z^{empty}) = f(sym(\pi), z^{empty}) \tag{5}$$

is guaranteed. Whenever the thief starts to pick up an item this fact does not hold anymore. Whenever TSP Solvers are used this fact has to be kept in mind.

Horizontal Alignments in the Objective Space

The objective space has many horizontal alignments. Solutions on the same horizontal line have the same profit $-g(z)$. Considering the TTP as a problem consisting of two subproblems, every horizontal line represents an embedded TSP. The topmost horizontal line where $g(z) = 0$ represents the TSP without any interaction with the KNP component. In the objective space of the example scenario (c.f. Figure 3) six different solutions are on each horizontal line. Because of the symmetry of π when $z = z^{empty}$ only three different points exist in the objective space on the topmost TSP line.

Right Shift

For solutions where $g(z) = 0$ the thief does not make any profit and $z = z^{empty}$. For these solutions the velocity during the tour is consistently v_{max}. If the thief starts to pick an item, he gets slower because the weight increase leads to velocity decrease. All solutions on the x-axis with $-g(z) = 0$ are shifted to the right when items are picked (c.f. Figure 3). These right shifts occur when z is optimized for a fixed π.

5 Algorithms

In the following different algorithms for the Bi-Objective TTP are proposed. The first two algorithms are deterministic and explained by pseudo code. Furthermore, the principles of the multi-objective evolutionary algorithm NSGA-II are described. Different factory and recombination operators are directly evaluated in Section 6.

Greedy Algorithm This algorithm represents a naive deterministic approach. The procedure is shown in Algorithm 1. First, an existing TSP Solver - in our implementation LKH with the Lin-Kernighan heuristic [14] - is used to calculate a tour π^*. A set which will contain only non dominated solutions in the further optimization procedure is initialized with $F(\pi^*, z^{empty})$ and $F(sym(\pi^*), z^{empty})$. As long as the maximum capacity constraint is not violated items are picked. The *getNextItemGreedily* method returns the next item which is not picked so far and provides the best $\frac{b_i}{w_i}$ rate. Then z combined with π^* and $sym(\pi^*)$ is added to the set. Whereby the *add* method needs to ensure to keep a set of non dominated solutions. Finally the resulting Pareto front is returned.

The *Greedy Algorithm* serves as a reference for the other algorithms and does maximally evaluate $O(2m) = O(m)$ times the $F(\pi, z)$ function. There is no

Algorithm 1. Greedy Algorithm

```
Require: $P \leftarrow$ Thief Problem
  $\pi^* \leftarrow$ tspSolver($P$)
  $z \leftarrow z^{empty}$
  front $\leftarrow \{F(\pi^*, z), F(sym(\pi^*), z)\}$
  while $\sum_{j=1}^{m} w_j z_j \leq Q$ do
      $i \leftarrow$ getNextItemGreedily($P$, $z$)
      $z_i \leftarrow= 1$
      add(front,$F(\pi^*, z)$)
      add(front,$F(sym(\pi^*), z)$)
  end while
  return front
```

maximum number for non dominated solutions in the final front which means for problem instances with many items and a high maximum capacity Q many solutions will be added to the front.

Independent Subproblem Algorithm Since a lot of research has been done on the knapsack problem, the approach above can be extended by using a KNP Solver instead of solving it greedily. The pseudo code for the algorithm is shown in Figure 2. After calculating π^*, the maximum weight capacity is divided equally into T intervals. For instance, if $Q = 12$ and $T = 4$ the corresponding maximum capacities are $I = \{0, 4, 8, 12\}$. For each of these maximum capacities the KNP Solver is used to calculate a packing plan z^*. Then - analogous to the Greedy Algorithm - the resulting packing plan is combined with π^* and $sym(\pi^*)$.

Algorithm 2. Independent Subproblem Algorithm

```
Require: $P \leftarrow$ Thief Problem
Require: $T \leftarrow$ Maximum of Solutions
  $\pi^* \leftarrow$ tspSolver($P$)
  $I \leftarrow$ getLinearMaxWeight($P$, $T$)
  front $\leftarrow \{\}$
  for all $i \in I$ do
      $z^* \leftarrow$ knpSolver($P$, $i$)
      add(front,$F(\pi^*, z^*)$)
      add(front,$F(sym(\pi^*), z^*)$)
  end for
  if localOptimize then
      while hasEvaluations() do
          add(front, localOptimize(getRandom(front)))
      end while
  end if
  return front
```

Additionally, a further local optimizing procedure can be applied by selecting randomly one solution of the Pareto front and optimizing it. In our implementation we perform with a probability of $\frac{1}{3}$ a swap tour mutation, of $\frac{1}{3}$ a bitlfip pack mutation and of $\frac{1}{3}$ both mutations. If the new solution is not dominated by any other solution, it is added to the front. The local optimization is done until no function evaluations are left. The standard version is labeled with ISA and with ISA-LOCAL if the front is local optimized.

NSGA-II As optimization framework the evolutionary multi-objective **N**on-dominated **S**orting **G**enetic **A**lgorithm-II (NSGA-II) [15] is used. As usual for evolutionary algorithms all individuals are factored and added to the population in the beginning. Then each individual gets a rank based on its level of domination and a crowding distance which is used as a density estimation in the objective space. The binary tournament selection compares rank and crowding distance of randomly selected individuals in order to return individuals for the recombination. After executing crossover and mutation operators the offspring is added to the population. In the end of each generation the population is truncated after sorting it by domination rank and crowding distance. All details of the algorithm can be found in [15].

Also, meta-algorithms provide a great possibility to use the same method to solve many problems, different domain specific operators have to be defined. NSGA-II needs to know how to create and recombine individuals. For the experiment we used a population size of 100 individuals and performed the mutation operations in 30% of cases. Moreover, an additional method to remove duplicates each generation is executed in order to prevent diversity loss. The evaluation of different operators is presented in Section 6

6 Results

In the following the proposed algorithms are evaluated and the results are shown. After describing the problem instances, different factory and recombination operators for NSGA-II are compared. Afterwards, the final evaluation takes the best found operator combination for NSGA-II and the other proposed algorithms into account.

All algorithms were executed 30 times on the same problem instance with maximum 100000 function evaluations each run. The median Pareto fronts of these runs were calculated by using the **E**mpirical first-order **A**ttainment **F**unction (EAF) [16].

Problem instances

We created different problem instances, that vary in the city location generator, number of cities, number of items per city and the knapsack capacity. The number of cities are 10, 20, 50 and 100 with 1, 5 and 10 items per city. The maximum capacity is divided into three different categories, namely small $c = 0.2$, medium

$c = 0.5$ and large $c = 0.8$ where $Q = c \sum_{i=1}^{m} w_i$. The velocity range is set to $v_{min} = 0.1$ and $v_{max} = 1$.

The name pattern for problem instances with the random city generator looks as follows:

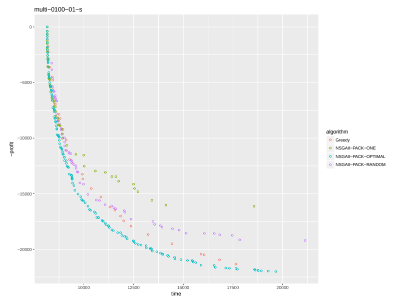
$$\text{multi} - \mathit{numOfCities} - \mathit{itemsPerCity} - \mathit{knapsackCapacity}$$

For all *multi-cluster-XX* problem instances 100 cities are assigned to XX different clusters. Only one item is assigned to each city and the knapsack has a medium capacity. This imitates a map with cities where many locations occur in densely populated areas.

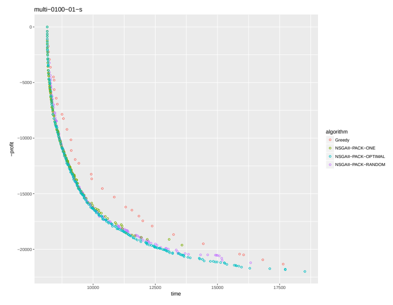
NSGAII - Knapsack Component

The factory is an important part of an evolutionary algorithm. Therefore, we made up an experiment in order to test three different packing plan factories. Note, that this experiment showed the same results whatever tour factory and recombination operators we combined it with. Two packing plans are combined by using a single point crossover and mutated by a bitflip mutation.

- PACK-ONE: Produces knapsacks with only one randomly picked item.
- PACK-RANDOM: Completely random knapsacks without violating the maximum capacity constraint.
- PACK-OPTIMAL: Define randomly $0 \leq Q' \leq Q$ and return the result of a KNP Solver [17] with maximum capacity Q'



(a) Generation 30



(b) Generation 100

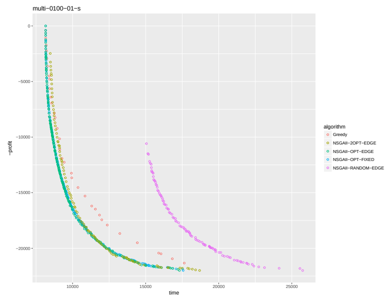
Fig. 4. Objective Space after 30 and 100 Generations on the Problem Instance *multi-0100-1-s* with different Knapsack Factories.

The obtained results for the knapsack factories for one problem instance are shown exemplarily in Figure 4. The factory PACK-OPTIMAL produces better solutions than the naive *Greedy* approach. PACK-ONE needs more time to converge with solutions with higher profit than PACK-RANDOM. All NSGA-II algorithms outperform *Greedy* after 100 generations. The same effect has been observed on all other problem instances.

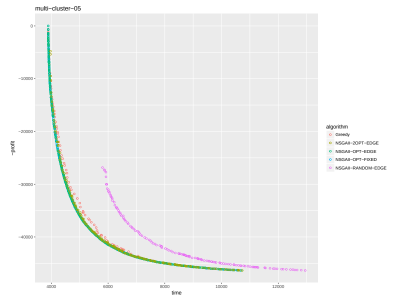
NSGAII - Tour Component

As it seems to be a good choice to factor packing plans z calculated by a KNP Solver, it should also be considered to return a tour π^* calculated by a TSP Solver. However, in this case only π^* and $sym(\pi^*)$ in the starting population exist and no diversity of different tours is given. Moreover, the 2OPT factory returns two-opt optimal [14] or the RANDOM factory completely random tours. The edge recombination crossover [18] (EDGE) is used to combine tours and a swap mutation to mutate. The recombination of packing plans is done as described above and the optimal packing plan factory is used. In order to see the usefulness of EDGE, we tried another factory FIXED as well which does not perform any crossover on the tours. To sum up, this experiment includes the following four combinations:

– NSGAII-RANDOM-EDGE: Random tours with EDGE
– NSGAII-2OPT-EDGE: Two-opt optimal tours with EDGE
– NSGAII-OPT-EDGE: Tours calculated by TSP Solver and EDGE
– NSGAII-OPT-FIXED: Tours calculated by TSP Solver and no crossover



(a) multi-0100-01-s



(b) multi-cluster-05

Fig. 5. Objective Space after 1000 Generations with different Tour Factories.

Figure 5 illustrates the results of the different tour factories. The algorithm NSGAII-RANDOM-EDGE shows the worst results because the evolution is not able to find tours that are fast enough. The OPT factory outperforms the 2OPT factory on the multi-0100-01-s instance. OPT-FIXED and OPT-EDGE showed almost the same results which means the edge recombination crossover did not bring much improvement. In fact, as mentioned above only two tours are in the starting population. A crossover between π^* itself or π^* and $sym(\pi^*)$ is difficult to perform. However, when a mutation brings an improvement, a diversity of tours might exist.

Evaluation

After analyzing the evolutionary components of NSGA-II, a complete evaluation including the other proposed algorithm is presented. The hypervolume of the normalized median Pareto fronts is shown in Table 6. The algorithm denoted by NSGAII is the combination of operators with the best performance found above. It uses for the packing plan the OPT factory, the bitlfip mutation and the single point crossover and for the tour the OPT factory, the swap mutation and the edge recombination crossover. The other algorithms are implemented as described in Section 5.

	Greedy	**ISA**	**ISA-LOCAL**	**NSGA-II**
multi-0010-01-m	0.562	0.852	**0.874**	**0.874**
multi-0010-05-m	0.79	0.861	0.863	**0.881**
multi-0020-01-m	0.616	0.797	0.805	**0.835**
multi-0020-05-m	0.626	0.816	0.816	**0.853**
multi-0050-01-m	0.701	0.847	0.848	**0.872**
multi-0100-01-l	0.803	0.805	0.806	**0.847**
multi-0100-01-m	0.772	0.777	0.781	**0.851**
multi-0100-01-s	0.741	0.752	0.755	**0.85**
multi-0100-05-l	0.832	0.830	0.830	**0.849**
multi-0100-05-m	0.808	0.806	0.806	**0.838**
multi-0100-05-s	0.801	0.800	0.800	**0.851**
multi-0100-10-l	0.843	0.840	0.840	**0.852**
multi-0100-10-m	0.825	0.822	0.822	**0.84**
multi-0100-10-s	0.808	0.806	0.806	**0.846**
multi-cluster-03	0.827	0.846	0.846	**0.867**
multi-cluster-05	0.813	0.867	0.868	**0.884**
multi-cluster-10	0.804	0.866	0.866	**0.881**

Table 3. Hypervolume of Normalized Median Pareto fronts after 100000 Evaluations

Clearly NSGA-II shows the best performance on all problem instances. Since *Greedy* and ISA do not use as many evaluations as the other algorithms, they are more or less a reference that must be outperformed. In fact, using a KNP Solver instead of picking up items greedily improves the overall result. But also note, that ISA-LOCAL is not able to show significantly better results than ISA.

Moreover, further analysis of NSGA-II Pareto fronts showed the following: In average only 3.39 different tours exist in each Pareto front. Also, there were runs where only the two starting tours exist. The maximum diversity was found in *multi-0100-01-s* with 15 different tours. In fact, this shows the challenge to start with fast tours in order to have a good initial population, but to preserve diversity during the evolution. Note, that all Pareto fronts seemed to be highly converged for each type of algorithm. The hypervolume of minimum and maximum Pareto fronts showed almost the same results over all runs.

7 Conclusion

After introducing the Bi-Objective Traveling Thief Problem, showing the interaction of its subproblems, and presenting their interdependencies the problem is defined mathematically. Characteristics of the problem itself and the objective space provide basic knowledge in order to propose algorithms for the Bi-Objective Traveling Thief Problem. The evaluation of these algorithms shows the advantages of multi-objective evolutionary methods in comparison to deterministic approaches. However, the diversity of tours in the Pareto front and therefore a better convergence might be improvable with other evolutionary operators. Therefore, operators that consider the interwovenness and are not performing recombinations on tour and packing plan independently might be useful. Furthermore, the performance of these algorithms on problem instances with a large-scale has to be verified. Even though, the Traveling Thief Problem combines two well-known problems, the interdependence is challenging and therefore the complexity is more than just the sum of these two subproblems. It is more, because the interwovenness does not allow applying state-of-the-art algorithms for TSP or KNP. Therefore, one plus one is more than two and further studies should be done on the overflow part.

Acknowledgment

This work was supported by a fellowship within the FITweltweit programme of the German Academic Exchange Service (DAAD).

References

1. H. Ishibushi, K. Klamroth, S. Mostaghim, B. Naujoks, S. Poles, R. Purshouse, G. Rudolph, S. Ruzika, S. Sayin, M. M. Wiecek, and X. Yao, *Multiobjective Optimization for Interwoven Systems.* Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.
2. D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics).* Princeton, NJ, USA: Princeton University Press, 2007.
3. M. G. Lagoudakis, “The 0-1 knapsack problem – an introductory survey,” 1996.
4. 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 Congress on Evolutionary Computation.* IEEE, 2013, pp. 1037–1044.
5. S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann, “A comprehensive benchmark set and heuristics for the traveling thief problem,” in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’14. New York, NY, USA: ACM, 2014, pp. 477–484. [Online]. Available: http://doi.acm.org/10.1145/2576768.2598249
6. G. Reinelt, “TSPLIB - A t.s.p. library,” Universität Augsburg, Institut für Mathematik, Augsburg, Tech. Rep. 250, 1990.

7. H. Faulkner, S. Polyakovskiy, T. Schultz, and M. Wagner, "Approximate approaches to the traveling thief problem," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 385–392. [Online]. Available: http://doi.acm.org/10.1145/2739480.2754716
8. M. R. Bonyadi, Z. Michalewicz, M. R. Przybylek, and A. Wierzbicki, "Socially inspired algorithms for the travelling thief problem," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: ACM, 2014, pp. 421–428. [Online]. Available: http://doi.acm.org/10.1145/2576768.2598367
9. R. Birkedal, "Design, implementation and comparison of randomized search heuristics for the traveling thief problem," Master's thesis, Technical University of Denmark, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, DK-2800 Kgs. Lyngby, Denmark, compute@compute.dtu.dk, 2015. [Online]. Available: http://www.compute.dtu.dk/English.aspx
10. Y. Mei, X. Li, F. Salim, and X. Yao, "Heuristic evolution with genetic programming for traveling thief problem," in *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, 2015, pp. 2753–2760. [Online]. Available: http://dx.doi.org/10.1109/CEC.2015.7257230
11. Y. Mei, X. Li, and X. Yao, "On investigation of interdependence between sub-problems of the travelling thief problem," *Soft Computing*, pp. 1–16, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00500-014-1487-2
12. ——, "Improving efficiency of heuristics for the large scale traveling thief problem," in *Simulated Evolution and Learning - 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings*, 2014, pp. 631–643. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-13563-2_53
13. C. Wachter, "Solving the travelling thief problem with an evolutionary algorithm," Diplomarbeit, Technischen Universitt Wien, 2015.
14. D. Applegate, W. Cook, and A. Rohe, "Chained lin-kernighan for large traveling salesman problems," *INFORMS J. on Computing*, vol. 15, no. 1, pp. 82–92, Jan. 2003. [Online]. Available: http://dx.doi.org/10.1287/ijoc.15.1.82.15157
15. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.
16. V. G. Fonseca and C. M. Fonseca, *The Attainment-Function Approach to Stochastic Multiobjective Optimizer Assessment and Comparison.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 103–130. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02538-9_5
17. S. Martello, D. Pisinger, and P. Toth, "Dynamic programming and strong bounds for the 0-1 knapsack problem," *Manage. Sci.*, vol. 45, no. 3, pp. 414–424, 1999.
18. I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application.* Mahwah, NJ, USA: L. E. Associates, Inc., 1987, pp. 224–230.