Comparing ALNS and NSGA-II on Bi-Objective Travelling Thief Problem

Kazım Erdoğdu
Department of Software Engineering
Faculty of Engineering
Yaşar University
İzmir, Turkey
kazim.erdogdu@yasar.edu.tr

Abstract

Real-life optimization problems consist of various conflicting optimization problems. Due to the interdependency of these sub-optimization problems, their solution is more difficult than single optimization problems. For this reason, most of the real-life optimization problems are considered multi-objective optimization problems by nature, and different methods and approaches are applied. In this paper, a recent multi-objective optimization problem, the bi-objective travelling thief problem (BiTTP) was studied. Two state-of-the-art algorithms, ALNS and NSGA-II, were hybridized with certain local search methods and applied on six problem instances of Polyakovskiy's problem benchmark set. As a result, the performances of these two algorithms were evaluated and a new set of Pareto front sets were generated for the related benchmark instances.

Keywords

Travelling Thief Problem, Multi-objective Optimization Problem, Adaptive Large Neighborhood Search, NSGA-II, Evolutionary Algorithms

1. Introduction

Many real-life optimization problems contain multiple combinatorial optimization problems that are usually interdependent to each other. Hence, the solution of these real-life optimization problems gets more difficult to solve due to the interdependency between these sub-optimization problems. The feasibility and quality of one sub-problem affect the feasibility and quality of the other sub-problems. An example of such combinatorial optimization problems is the Travelling Thief Problem (TTP). It contains two well-known combinatorial optimization problems, which are the Travelling Salesman Problem (TSP) and Knapsack Problem (KP). The former is in the NP-Hard problem class while the second one is in the NP-Complete problem class, which together make TTP an NP-Hard problem (Bonyadi et al. 2013). TTP can be considered as either a single or a multi-objective optimization problem due to its nature. However, considering TTP as a multi-objective optimization problem is closer to the real-life cases since real-life problems contain multiple objectives to be solved simultaneously. In this paper, a bi-objective TTP (Bi-TTP) was studied. Bi-TTP has recently been an attraction to the scientist that study combinatorial optimization. It requires multiple approaches and methods to solve its sub-problems. Although many studies are being done for the single-objective version of TTP, the number of studies for the multi-objective version is only a handful. The motivation of this paper is to contribute to Bi-TTP by applying two state-of-the-art heuristics: Adaptive Large Neighborhood Search (ALNS) (Demir et al. 2012) and Fast and Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al. 2002). The reason for picking these two heuristics is that the former one uses a single solution during its search process and the latter uses a solution population in its search process. The results of these two algorithms were compared and a new set of Pareto fronts were obtained by their combination. This is the main contribution of this study for the Bi-TTP in literature: applying and comparing two different type metaheuristics on BiTTP, and obtaining Pareto front sets for certain benchmark TTP sets.

2. Literature Review

Bonyadi et al. (2013) are the first ones to study TTP. They defined the problem the first time in literature and modelled two variations of the problem: TTP_1 and TTP_2. The former is a single-objective model where the objective is to maximize the total benefit, while the latter is a bi-objective model where the objectives are maximizing the total value

of picked items and minimizing the total travel time of the salesperson. In addition, the value of each item drops in time in TTP_2. In this literature review, the TTP studies were categorized according to single versus multi-objective approaches and the solution methods used in the studies.

The majority of the studies on TTP in literature are the single-objective version of TTP, that is TTP_1. Different solution approaches have been proposed and used to solve single-objective TTP: evolutionary algorithm (EA) (Polyakovskiy et al. 2014) (Chand and Wagner 2016) (Wagner 2016) (Faulkner et al. 2015), a co-evolutionary algorithm (Bonyadi et al. 2014) (El Yafrani and Ahiod 2016) (Mei et al. 2016), memetic algorithm (Mei et al. 2014) (Mei et al. 2016), greedy heuristics (Gupta and Prakash 2015), genetic programming (Mei et al. 2015) (El Yafrani et al. 2018), swarm intelligence (Wagner 2016), exact approaches (Wu et al. 2017), only local search approaches (El Yafrani and Ahiod 2017) (Nieto-Fuentes et al. 2018) (Maity and Das 2020), simulated annealing (El Yafrani and Ahiod 2018). Among these single-objective TTP studies, Chand and Wagner studied a version of TTP that includes multiple travelling salesperson (Chand and Wagner 2016). This way their problem has similarities with the vehicle routing problem (VRP). Besides these studies, Wagner et al. (2018) made an analysis study on the performance of 21 different algorithms. They applied these algorithms on a subset of Polyakovskiy benchmark instances (Polyakovskiy et al. 2014). According to their results, although there is not one single algorithm strongly dominating others, yet the co-evolutionary algorithms respectively performed slightly better than the others.

The studies in multi-objective TTPs are much less than the single-objective ones. The solution methods used in these multi-objective TTPs are greedy algorithms (Blank et al. 2017), evolutionary algorithms (Yafrani et al. 2017), a version of NSGA-II (Chagas et al. 2021), and the weighted-sum method (Chagas and Wagner 2022). The authors in these studies used either hypervolume (HV) or empirical attainment function (EAF) performance metrics to measure the quality of their solutions.

In both single and multi-objective TTP studies, the main solution methods were reinforced by local searches. Since multi-objective TTP includes two sub-optimization problems (i.e. TSP and KP), different local searches were separately applied on these subproblems. For the TSP part, the majority of the studies applied Chained Lin-Kernighan Heuristic (CLKH) (Applegate et al. 2003) to construct the initial TSP tour. After this construction of the initial tour, well-known operators such as 2-Opt, insertion, and swap were applied to generate better solutions during the local search phase of the algorithms. For the KP part, the majority of the studies used the PACKITERATIVE method (Wagner 2016) as a constructive heuristic. Others either used a greedy algorithm to construct the initial knapsack permutation or randomly selected items to include in the knapsack. For the former one, the items were sorted by their value and weight ratio and then were included in the knapsack. During the local search on the knapsack, generally, bit-flip and insertion operators were used. The studies that used crossover operators were either applied standard crossover operators such as ordered crossover (OX) or one-point crossover (OPX) or generated their operators such as biased random key crossover (BRK) (Chagas et al. 2021). The studies that used mutation operators applied the standard mutations like bit-flip, and swap.

The majority of the multi-objective TTP studies, used the benchmark instances generated by Polyakovskiy et al. (2014), while some of them generated their instances. Since there are 9720 Polyakovskiy problem instances, each study picked a subset of these instances for their experimental studies. Most of the multi-objective TTP studies ran their algorithms on each instance for a maximum of 600 seconds and each instance was run 30 times. The best results out of these runs were recorded as the solution of the algorithm.

3. Definition of the Problem

In this study, a multi-objective TTP with two objectives (i.e. BiTTP) was studied. BiTTP contains two components, that are, TSP and KP. Depending on these components, it can be defined as follows. There are n cities $\{1,2,\dots,n\}$ and m items $\{1,2,\dots,m\}$ distributed in those cities. There is a thief who needs to travel all those cities and pick up (or steal) items from the cities during his or her tour. The thief has to begin with the first city, visit each city exactly once and return to the first city. During the tour, he or she picks up any item he or she wants. In other words, not all items have to be picked up. Each item has a profit value p_i and weight w_i. The thief aims to find a pick-up plan $z = \{z_1, z_2, \dots, z_m\}$ for KP component to maximize the total profit of the collected items (Eq. 1) and find a TSP tour $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ to minimize the total travelling time along with the packing plan (Eq. 2). Here, z_i indicates the i-th item and π_j indicates the j-th city, where $i \in \{1,2,\dots,m\}$ and $j, \pi_j \in \{1,2,\dots,n\}$. If the i-th item is picked then $z_i = 1$ otherwise

$z_i = 0$. Since, the thief always begins and ends his or her tour with the first city, $\pi_1 = 1$. Depending on these definitions, the objective functions and the constraints of BiTTP are described as follows.

$$\max \ \ g(z) = \sum_{i=1}^{m} p_i . z_i \tag{1}$$

$$\min \ \ h(\pi, z) = \sum_{i=1}^{n-1} \frac{d(\pi_i, \pi_{i+1})}{v(W)} + \frac{d(\pi_n, \pi_1)}{v(W)} \tag{2}$$

Where

$$W = W(i, \pi, z) = \sum_{k=1}^{i} \sum_{j=1}^{m} w_j . z_j . a_j(\pi_k) \tag{3}$$

$$a_j(\pi_k) = \begin{cases} 1, & \text{if item } j \text{ is picked up at city } \pi_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$v(W) = \begin{cases} v_{max} - \frac{W}{Q} . (v_{max} - v_{min}), & \text{if } \ \mathrm{W} \leq \mathrm{Q} \\ v_{min}, & \text{otherwise} \end{cases} \tag{5}$$

Here, $d(\pi_i, \pi_j)$ is the distance between the cities of π_i and π_j, W is the total weight of the knapsack while the thief is travelling to the city π_k, Q is the capacity of the knapsack, v is the travelling speed of the thief during his or her travel from the city π_i to the city π_{i+1}, and v_{min} and v_{max} are the lower and upper bounds of the speeds that the thief can travel at all times. It is seen that these two objective functions (Eq. 1) and (Eq. 2) are interdependent. Which items to include in the knapsack affect the values of both functions. While the value of an item makes a positive contribution to the first objective function, its weight reduces the speed of the thief, which in turn, increases the total travelling time. Furthermore, if the item's price and weight are directly proportional, which is the case most of the time, then these two objective functions become conflicting objectives. As a result, TTP can easily be considered a multi-objective optimization problem.

These objective functions are subject to certain constraints mentioned below (Chagas and Wagner 2022):

$$\sum_{i=1}^{m} z_i . w_i \leq Q \tag{6}$$

$$z_i \in \{0,1\} \tag{7}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \qquad 1 \leq j \leq n \tag{8}$$

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1, \qquad 1 \leq i \leq n \tag{9}$$

$$u_i - u_j + n . x_{ij} \leq n - 1, \qquad 2 \leq i \neq j \leq n \tag{10}$$

$$1 \leq u_i \leq n-1, \qquad 2 \leq i \leq n \tag{11}$$

$$u_i \in \mathbb{Z}, \qquad 2 \leq i \leq n \tag{12}$$

$$\pi_1 = 1 \tag{13}$$

$$x_{ij} \in \{0,1\}, \qquad 1 \leq i \leq n \tag{14}$$

Here, constraints (6) and (7) are for the KP component and (8) – (14) are for the TSP component of the BiTTP. Constraint (6) ensures that the knapsack capacity is not exceeded. (7) indicates that any item can be picked or disregarded. Constraints (8) and (9) ensure that each city is visited exactly once. Constraints (10) – (12) eliminate the sub tours for the TSP. Constraint (13) guarantees that the thief begins his or her tour at the first city and (14) is the decision variable for the TSP component.

4. Methods

In this paper, two main methods, ALNS and NSGA-II, were hybridized with local searches and applied on the selected benchmark instances in the experimental studies. All these methods and algorithms are briefly explained below.

ALNS (Demir et al. 2012) is one of the two main algorithms used in this study. It begins with an empty non-dominated solution population called archive. The initial solutions are obtained by two constructive heuristics: the TSP tour is constructed by Chained-Lin Kernighan Heuristic (CLKH) (Applegate et al. 2003) and the knapsack is constructed by a greedy heuristic (GH). Then these solutions are non-dominantly added to the archive. Then the search process begins. ALNS uses Simulated Annealing (SA) heuristic for exploration in the search space. At each iteration of the SA, a new solution is found via local searches and this new solution is non-dominantly added to the archive. Depending on the performance of each TSP and KP local search moves, the current and best solutions for SA are updated. Since in a bi-objective optimization problem, there is no best solution but a set of Pareto front solutions, the best solution in the SA is determined by a cost function (Eq. 15). This cost function is the objective function in single objective TTPs that calculates the overall profit of the thief by subtracting the rental price from the total profit of the picked items. R in the function is the hourly rent price of the vehicle that the thief uses and it is provided in the problem instance data. The pseudocode for ALNS used in this study is given in Algorithm 1.

$$f(\pi, z) = g(z) - R.h(\pi, z) \tag{15}$$

The second main solution method used in this study is NSGA-II (Deb et al. 2002). It is a genetic algorithm that contains the concept of non-dominant sorting of the solution population by ranking each solution according to a dominance criterion. In this study, NSGA-II begins with an initial population. The first two solutions of the population are obtained via CLKH and GH heuristics. The rest of the population is randomly generated. The population size is set to 100. After this initial process, the iterations of NSGA-II begin. At each iteration, two solutions are selected by binary tournament selection. Then two new child solutions are generated by a crossover process. For the TSP tour, ordered crossover (OX), and for the knapsack single-point crossover (SPX) are applied. These newly generated offspring solutions are added to the population. The same local searches are applied to these offspring and they are also added to the population. Unlike ALNS, the applied local searches are picked randomly at each iteration. At the end of an iteration, the population is non-dominantly sorted and truncated before the beginning of the next iteration. Once all the iterations are completed, the solutions with rank = 0 in the last population consists of the Pareto front result for NSGA-II. Algorithm 2 gives the pseudocode for NSGA-II used in this study.

Algorithm 1: ALNS

input : a BiTTP problem instance
output: Pareto front

$archive \leftarrow \emptyset$
$\pi \leftarrow$ construct initial tour by CLKH
$archive$.add(solution$(\pi, \emptyset)$)
$z \leftarrow$ construct initial knapsack by GH
$archive$.add(solution(π, z))
$\pi_{best} \leftarrow \pi$
$z_{best} \leftarrow z$
Initialize the weights and scores of TSP and KP operators
$T \leftarrow 100$ // Initial temperature for SA
while running_time $<$ 600 seconds **do**
  $\pi_{new} \leftarrow$ TSP_local_search(π)
  $z_{new} \leftarrow$ KP_local_search(π, z)
  $archive$.add(solution(π_{new}, z_{new}))
  if $f(\pi_{new}, z_{new}) < f(\pi, z)$ **then**
    $\pi \leftarrow \pi_{new}$
    $z \leftarrow z_{new}$
    if $f(\pi_{new}, z_{new}) < f(\pi_{best}, z_{best})$ **then**
      $\pi_{best} \leftarrow \pi_{new}$
      $z_{best} \leftarrow z_{new}$
    end
  else
    if $Random() < e^{-\frac{f(\pi_{new}, z_{new}) - f(\pi, z)}{T}}$ **then**
      $\pi \leftarrow \pi_{new}$
      $z \leftarrow z_{new}$
    end
  end
  Update TSP and KP local searh operator weights and scores
  $T \leftarrow T * 0.99$
end
return $archive$

Algorithm 2: NSGA-II

input : a BiTTP problem instance
output: Pareto front

$S \leftarrow \emptyset$
$S \leftarrow$ construct initial tour by CLKH
$S \leftarrow S \cup \{$solution$(\pi, \emptyset)\}$
$z \leftarrow$ construct initial knapsack by GH
$S \leftarrow S \cup \{$solution$(\pi, z)\}$
for $i \leftarrow 2$ **to** $population\_size$ **do**
  $\pi \leftarrow$ construct a random tour
  $z \leftarrow$ construct a random knapsack
  $S \leftarrow S \cup \{$solution$(\pi, z)\}$
end
for $i \leftarrow 1$ **to** $population\_size$ **do**
  $NP \leftarrow$ LocalSearch$(S[i])$
  $S \leftarrow S \cup NP$
end
Non-dominated_Sorting(S)
Truncate(S)
while running_time $<$ 600 seconds **do**
  while $|S| < 2*population\_size$ **do**
    $p_1, p_2 \leftarrow$ Binary tournament selection
    $c_1, c_2 \leftarrow$ Crossover(p_1, p_2)
    $S \leftarrow S \cup \{c_1, c_2\}$
    $NP \leftarrow$ LocalSearch(c_1, c_2)
    $S \leftarrow S \cup NP$
  end
  Non-dominated_Sorting(S)
  Truncate(S)
end
$Pareto\_front \leftarrow \emptyset$
for $i \leftarrow 1$ **to** $|S|$ **do**
  if rank$(S[i]) = 0$ **then**
    $Pareto\_front \leftarrow Pareto\_front \cup S[i]$
  end
end
return $Pareto\_front$

These two methods are hybridized with four local searches for TSP and four local searches for KP. TSP local searches consist of 2-Opt first improvement rule, 2-Opt best improvement rule, swap and insert search methods. KP local searches consist of constructing_KP_random, constructing_KP_by_score_values, bit-flip, and swap search methods. 2-Opt search method (Croes, 1958) aims to improve the current TSP tour by selecting two cities in the tour one by one and inverting the order between these two cities. If the new permutation provides a lesser tour distance then it updates the tour, otherwise, it continues to search without any change. Both 2-Opt methods used in the local searches are the same except for their termination criteria. Hence the name, 2-Opt first improvement rule returns its better solution as soon as it improves the TSP tour the first time, while 2-Opt best improvement rule returns the best solution after trying every possible move on the given TSP tour. The swap method exchanges two cities randomly selected in the TSP tour while the insert method randomly picks one city in the tour, removes it from its current location, and replaces it in a random location. In swap and insert search moves, the tour is updated only if the search move improves the TSP tour distance with regards to the total travelling distance. The constructing_KP_random method randomly adds items in the knapsack without exceeding the knapsack's capacity. The constructing_KP_by_score_values method uses a score function to add items into the knapsack. We implemented the score function proposed by (Maity and Das, 2020). For simplicity, the arbitrary exponential variables in their score function were omitted in this study. Bit-flip operator randomly selects an item in the items list and changes the status of that item. If the item is already in the knapsack then it removes the item from the knapsack or vice versa. The swap operator randomly selects two items in the knapsack array and swaps them.

During the TSP and KP local search processes, all the searches were done in the feasible region and the found infeasible solutions were disregarded. Swap, insert cities, and bit-flip moves were applied to a solution multiple times.

This number was determined randomly between 0 and $\boldsymbol{n / 10}$ for TSP local search moves and between 0 and $\boldsymbol{m / 10}$ for KP local search moves. The rest of the local search moves were applied only once per solution. At each local search phase in ALNS and NSGA-II, only one of the TSP local search method and one KP local search method was applied. In ALNS, the decision on which local search method to be selected was made according to the previous success of the local searches. In ALNS, each local search's probability of selection is adaptively updated depending on their previous success in the search. In NSGA-II, however, there is no prior record of success of each local search move and they are randomly selected at each iteration.

5. Performance Metrics

There are various performance metrics for evaluating the performance of solution algorithms for multi-objective optimization problems. An intensive list of these state-of-the-art performance metrics can be found in the study of (Li and Yao 2019). In our paper, we used two of these performance metrics to evaluate the performance of ALNS and NSGA-II for solving the BiTTP being studied: hypervolume (HV) and C-Metric.

HV indicates the volume (or, area in this study) of the enclosed region in the solution space by the algorithm's Pareto front and the nadir point (Figure 1). In this study, the nadir points for each problem instance were selected as the $(\boldsymbol{h(\pi, z)_{max}, g(z)_{min}})$ coordinates in the solution space after combining the ALNS and NSGA-II Pareto fronts. The higher values of HV indicate a better quality of the solution. HV measures the convergence and spread of the given Pareto front. Convergence refers to the closeness of the algorithm's Pareto front to the true or best-known Pareto front. Spread refers to the distribution of the solutions in the algorithm's Pareto front. The higher value of HV indicates better convergence and spread of the solution.

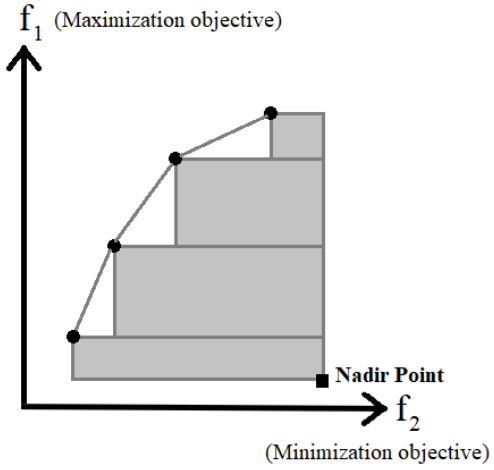

Figure 1. HV in BiTTP

C-Metric, on the other hand, measures the dominance relation between two Pareto fronts. C-Metric calculates the relative quality of two solutions based on the number of solutions they dominate each other. The higher values of C-Metric indicate better performance of the solution. C-Metric is calculated as follows:

$$\text{C-Metric}(A, B) = \frac{|\{b \in B \mid \exists a \in A \; such \; that \; a \preccurlyeq b\}|}{|B|} \tag{16}$$

In Equation 16, A and B represent two Pareto front sets of two different algorithms, $\boldsymbol{a}$ and $\boldsymbol{b}$ are individual solutions in related Pareto fronts, $|\{\ldots\}|$ is the cardinality of the related set and $\boldsymbol{a \preccurlyeq b}$ indicates that solution $\boldsymbol{a}$ dominates solution $\boldsymbol{b}$. This way, $\text{C-Me}\mathbf{tric}(\boldsymbol{A, B})$ calculates the ratio of the number of solutions in B that are dominated by at least one element in A over the cardinality of B. This value shows the coverage of A over B.

6. Benchmark Instances

In this paper, a subset of the Polyakovskiy benchmark instances were used in the experimental studies. Polyakovskiy et al. (2014) generated 9720 instances based on the conventional 81 TSP instances with the number of cities varying from 14 to 85900. They converted these TSP instances to TTP instances by generating knapsack items with weights and profits. Each TSP instance was attached with a combination of these different options: four different numbers of items (i.e. 1, 3, 5, 10) per city, three different types of knapsacks (i.e. bounded and strongly correlated (bsc), uncorrelated with similar weights (usw), and uncorrelated (unc)), and ten different sizes of knapsacks (i.e. 1, 2, …, 10 times the size of the smallest knapsack). Regarding the type of knapsack, although all of these three options were generated by uniform distribution, the correlation between the weight and profit of each item gets stronger in this order: unc, usw, and bsc. As the type of knapsack gets more strongly correlated, the feasible solution space gets expanded which, in turn, complicates the searching process for the algorithms.

In this study, only a small sample subset of Polyakovskiy instances was selected: berlin52_n51_bsc_01, berlin52_n255_usw_05, berlin52_n510_uc_10, a280_n279_bsc_01, a280_n1395_usw_05, and a280_n2790_uc_10. In these instance names, each part between the underscores gives information about the TTP instance. The first part gives the name of the TSP instance and the number of cities in the instance, the second part gives the number of knapsack items, the third part indicates the type of the knapsack and the last part shows the coefficient of the size of the knapsack. These instances can be found at https://cs.adelaide.edu.au/~optlog/CEC2014COMP_InstancesNew/.

7. Results

In this paper, the proposed methods, i.e. ALNS and NSGA-II were applied to the aforementioned benchmark instances. Each algorithm was run 30 times on each instance and each run was limited to 600 seconds. It is because most of the studies in the literature used the same number of runs and termination criteria. The Pareto fronts of each algorithm were obtained by non-dominantly combining each run's results. In addition, the Pareto fronts of each algorithm were also combined to obtain the combined Pareto fronts for each related instance. Due to the space limitation in this paper, all the detailed results were uploaded to the website https://kerdogdu.yasar.edu.tr/benchmark-instances-results/bittp-results-ieom-2022-conference/. In the subsections below, the performances and sample graphical results were given.

Table 1 shows the HV and C-Metric results of ALNS and NSGA-II in this study. In the table, n and m indicate the number of cities and number of items in the related instance, respectively. As is seen in the performance metric results, NSGA-II outperforms ALNS in all instances in this experimental study. The higher values of HV for NSGA-II indicate that the Pareto front found by NSGA-II covers a greater area in the solution space than the one covered by ALNS, which means that NSGA-II's Pareto fronts are much closer to the true Pareto front than the ALNS's Pareto fronts for the related instance. C-Metric, on the other hand, tells more about the cardinality of dominance between the results of two algorithms. According to the last two columns of Table 1, it is obvious that the solutions in the NSGA-II Pareto fronts mostly dominate the solutions in the ALNS's Pareto fronts. Except for "a280_n2790_uc_10" instance, the solutions in the ALNS's Pareto fronts do not dominate any solution in NSGA-II Pareto fronts for the related instance. According to the results of both HV and C-Metric of both algorithms, it can easily be said that NSGA-II produced more spread, covering, and dominating solutions than ALNS for the BiTTP studied in this experimental study.

Table 1. HV and C-Metric values

Instance	n	m	HV (ALNS)	HV (NSGA-II)	C-Metric (ALNS, NSGA-II)	C-Metric (NSGA-II, ALNS)
berlin52_n51_bsc_01	52	51	0.96	0.99	0.00%	72.73%
berlin52_n255_usw_05	52	255	0.39	0.70	0.00%	25.00%
berlin52_n510_uc_10	52	510	0.56	0.72	0.00%	76.47%
a280_n279_bsc_01	280	279	0.73	0.88	0.00%	77.78%
a280_n1395_usw_05	280	1395	0.40	0.51	0.00%	85.71%
a280_n2790_uc_10	280	2790	0.52	0.54	0.54%	62.50%

Figures 2 and 3 are sample graphical results for ALNS and NSGA-II algorithms on "berlin52_n51_bsc_01" and "a280_n279_bsc_01" instances, respectively. It is seen in both graphics that NSGA-II contains more non-dominated

solutions in its Pareto fronts than the solutions in ALNS's Pareto fronts. Furthermore, most of the NSGA-II Pareto front solutions dominate the ALNS Pareto front solutions. This fact explains the performance metric values of both algorithms mentioned in section 5.1. Since the BiTTP aims to maximize the profit objective (Eq. 1) and to minimize the travelling time objective (Eq. 2), lesser values on the horizontal axis and greater values on the vertical axis are preferable. Therefore, the Pareto fronts obtained by NSGA-II in both samples provide better solutions than ALNS.

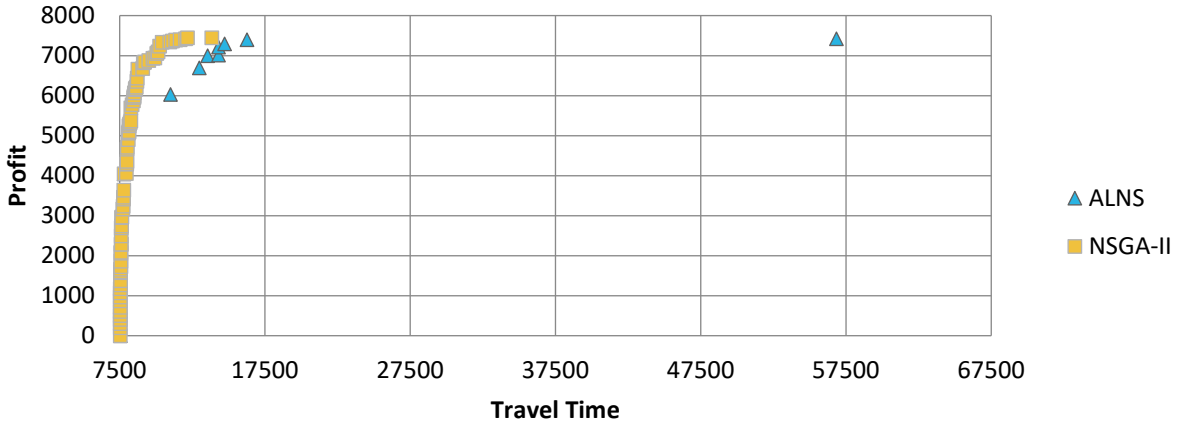

Figure 2. ALNS and NSGA-II Pareto fronts for berlin52_n51_bsc_01 instance

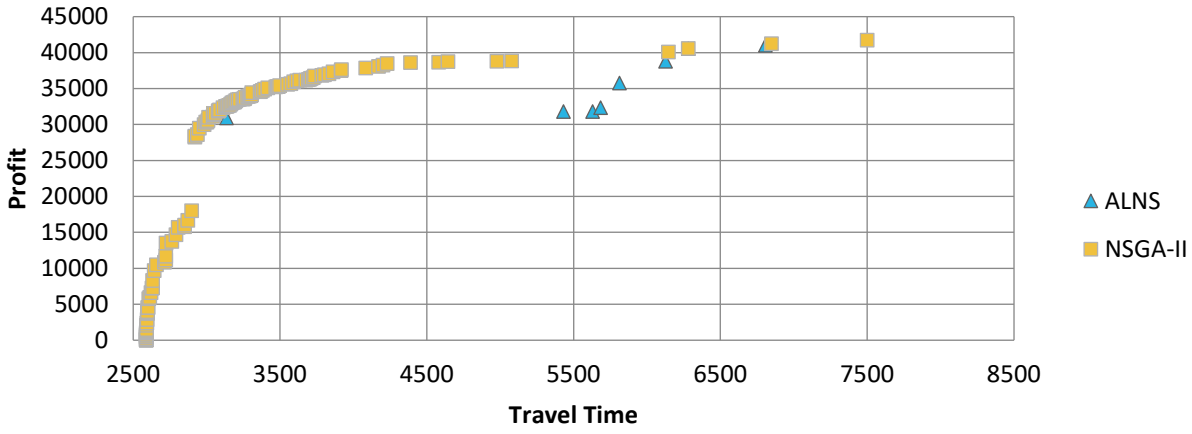

Figure 3. ALNS and NSGA-II Pareto fronts for a280_n279_bsc_01 instance

8. Conclusion

In this paper, one of the state-of-the-art problems, BiTTP, was studied. Although there are many studies for single objective TTP, the bi-objective version of the TTP has recently been an attraction to researchers. BiTTP is closer to real-life problems but it is more difficult to solve due to its multi-objective nature. In this paper, these two objectives for BiTTP were taken into consideration: maximizing the total profit of the picked items and minimizing the total travel time of the thief. Two state-of-the-art algorithms, ALNS and NSGA-II, were hybridized with certain local search methods and applied on six problem instances of Polyakovskiy's problem benchmark set.

The experimental results show the outperforming performance of NSGA-II over ALNS in the BiTTP studied in this paper. The main reason for this conclusion is mostly because of NSGA-II's use of population in the search process. NSGA-II can simultaneously evaluate multiple solutions in its population. Although this is an advantage for NSGA-II over ALNS, handling a population at each iteration extends the iteration time which in turn decreases the number of iterations. This fact can be very disadvantageous for NSGA-II when running on the instances having much higher

n and m values. ALNS, on the other hand, uses only one solution during its search process and keeps updating it according to a non-dominance criterion. Working on only one solution speeds up the search process and increases the number of iteration. That is an advantage of ALNS since the search can do more exploration and exploitation with more iterations. In addition, ALNS allows respectively worse solutions at certain times in the search to prevent the search from being stuck in the local optima for both objectives. The local search in ALNS keeps a record of the success of each search move so that it can give more chances to more successful moves in the next search iterations. This is also an advantage of ALNS over NSGA-II. And yet, despite all these advantages of ALNS, it was still outperformed by NSGA-II in the experimental studies in this paper.

The number of cities (i.e. *n*) and the number of items (i.e. *m*) also affected the search process for both algorithms. As *n* and *m* increase, the performance of the proposed algorithms decreases proportionally. It can be seen in the performance metrics in Table 1. This is an expected result since the increase of *n* and *m* expands the feasible solution space. The type of the knapsack might have affected the performance of the algorithms as well, and yet it is difficult to make a clear comment on that. In a future study, more of the same knapsack type instances can be included in the experimental studies and their effect can be analyzed better. Furthermore, NSGA-II's and ALNS's performances can also be evaluated by applying them to the instances with much higher *n* and *m* values, which can be done in another study.

The main contribution of this study is that applying two state-of-the-art heuristics for multi-objective optimization problems (ALNS and NSGA-II) to BiTTP, comparing their performance, and generating new Pareto front benchmarks. To the best of our knowledge, this is the first study that ALNS was applied on BiTTP. Although its performance was not as good as NSGA-II, it could be improved by implementing other local search moves that will be more effective in exploring and exploiting the search space.

The BiTTP studied in this paper can easily be adapted to real-life cases. It can be modified and applied on the distribution of products, collecting regular, recycling, medical or radioactive wastes, public transportation, transportation of water tanks and printed circuit board design (Bonyadi et al. 2013), etc. Once the objective functions and the problem-specific constraints are modified to these real-life cases, the mathematical model and the proposed methods can easily and successfully be applied to the problem.

In future studies, a new heuristic algorithm can be investigated or the current ones can be improved by new local search heuristics. These algorithms can also be applied to the Polyakovskiy instances that have a higher number of cities and items so that their performances can be evaluated according to the increase in these dimensions. In addition, more of the same type of knapsack (i.e. usc, uc, bsw) instances can be included to see their effect on the solution.

References

Applegate, D., Cook, W. and Rohe, A., Chained Lin-Kernighan for large travelling salesman problems, *INFORMS: Journal on Computing*, vol. 15, no. 1, pp. 82–92, 2003.

Blank, J., Deb, K. and Mostaghim, S., Solving the bi-objective travelling thief problem with multi-objective evolutionary algorithms, *Lecture Notes in Computer Science*, vol. 10173, pp. 46–60, 2017.

Bonyadi, M. R., Michalewicz, Z. and Barone, L., The travelling thief problem: The first step in the transition from theoretical problems to realistic problems, *2013 IEEE Congress on Evolutionary Computation*, pp. 1037–1044, Cancun, Mexico, June 20-23, 2013.

Bonyadi, M. R., Michalewicz, Z., Przybyłek, M. R. and Wierzbicki, A., Socially inspired algorithms for the travelling thief problem, *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, pp. 421–428, Vancouver BC, Canada, July 12-16, 2014.

Chagas, J. B. C., Blank, J., Wagner, M., Souza, M. J. F. and Deb, K., A non-dominated sorting based customized random-key genetic algorithm for the bi-objective travelling thief problem, *Journal of Heuristics*, vol. 27, no. 3, pp. 267–301, 2021.

Chagas, J. B. C. and Wagner, M., A weighted-sum method for solving the bi-objective travelling thief problem, *Computers and Operations Research*, vol. 138, 2022.

Chand, S. and Wagner, M., Fast heuristics for the multiple travelling thieves problem, *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, pp. 293–300, Denver, Colorado, USA, July 20-24, 2016.

Croes, G. A., A Method for Solving Travelling-Salesman Problems, *Operations Research*, vol. 6, no. 6, pp. 791–

812, 1958.
Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, 182–197, 2002.
Demir, E., Bektaş, T. and Laporte, G., An adaptive large neighborhood search heuristic for the Pollution-Routing Problem, *European Journal of Operational Research*, vol 223, no. 2, pp. 346–359, 2012.
El Yafrani, M. and Ahiod, B., Population-based vs. Single-solution heuristics for the Travelling Thief Problem, *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, pp. 317–324, Denver, Colorado, USA, July 20-24, 2016.
El Yafrani, M. and Ahiod, B., A local search based approach for solving the Travelling Thief Problem: The pros and cons, *Applied Soft Computing*, vol. 52, pp. 795–804, 2017.
El Yafrani, M. and Ahiod, B., Efficiently solving the Travelling Thief Problem using hill climbing and simulated annealing, *Information Sciences*, vol. 432, pp. 231–244, 2018.
El Yafrani, M., Martins, M., Wagner, M., Ahiod, B., Delgado, M. and Lüders, R., A hyperheuristic approach based on low-level heuristics for the travelling thief problem, *Genetic Programming and Evolvable Machines*, vol. 19, no. 1–2, pp. 121–150, 2018.
Faulkner, H., Polyakovskiy, S., Schultz, T. and Wagner, M., Approximate approaches to the travelling thief problem, *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, pp. 385–392, Madrid, Spain, July 11-15, 2015.
Gupta, B. C. and Prakash, V. P., Greedy heuristics for the Travelling Thief Problem, *2015 39th National Systems Conference*, pp. 1–5, Greater Noida, India, December 14-16, 2015. https://doi.org/10.1109/NATSYS.2015.7489116
Li, M. and Yao, X., Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey, *ACM Computing Surveys*, vol. 52, no. 2, pp. 1-38, 2019.
Maity, A. and Das, S., Efficient hybrid local search heuristics for solving the travelling thief problem, *Applied Soft Computing*, vol. 93, 2020.
Mei, Y., Li, X., Salim, F. and Yao, X., Heuristic evolution with Genetic Programming for Travelling Thief Problem, *2015 IEEE Congress on Evolutionary Computation*, pp. 2753–2760, Sendai, Japan, May 25-28, 2015
Mei, Y., Li, X. and Yao, X., Improving efficiency of heuristics for the large scale travelling thief problem, *Lecture Notes in Computer Science*, vol. 8886, pp. 631–643, 2014.
Mei, Y., Li, X. and Yao, X., On investigation of interdependence between sub-problems of the Travelling Thief Problem, *Soft Computing*, vol. 20, no. 1, pp. 157–172, 2016.
Nieto-Fuentes, R., Segura, C. and Valdez, S. I., A Guided Local Search Approach for the Travelling Thief Problem, *2018 IEEE Congress on Evolutionary Computation*, pp. 1-8, Rio de Janeiro, Brazil, July 8-13, 2018.
Polyakovskiy, S., Bonyadi, M. R., Wagner, M., Michalewicz, Z. and Neumann, F., A comprehensive benchmark set and heuristics for the travelling thief problem, *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, pp. 477–484, Vancouver BC, Canada, July 12-16, 2014.
Wagner, M., Stealing items more efficiently with ants: A swarm intelligence approach to the travelling thief problem, *Lecture Notes in Computer Science*, vol. 9882, pp. 273–281, 2016.
Wagner, M., Lindauer, M., Mısır, M., Nallaperuma, S. and Hutter, F., A case study of algorithm selection for the travelling thief problem, *Journal of Heuristics*, vol. 24, no. 3, pp. 295–320, 2018.
Wu, J., Wagner, M., Polyakovskiy, S. and Neumann, F., Exact approaches for the travelling thief problem, *Lecture Notes in Computer Science*, vol. 10593, pp. 110–121, 2017.
Yafrani, M. El, Chand, S., Neumann, A., Ahiod, B. and Wagner, M., Multi-objectiveness in the single-objective travelling thief problem, *GECCO 2017 - Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 107–108, Berlin Germany, July 15-19, 2017.

Biography

Kazım Erdoğdu, Ph.D. currently works as an Asst. Prof. Dr. at Department of Software in Faculty of Engineering at Yaşar University. After completing his B.Sc. degree in the Department of Computer Sciences in Mathematics at Ege University Faculty of Sciences, he accomplished his M.Sc. degree in Department of Mathematics at Ege University Graduate Faculty of Natural and Applied Science. He earned his Ph.D. degree in the Department of Computer Engineering at Yaşar University Graduate Faculty of Natural and Applied Science. He was employed as a full-time lecturer at Yaşar University in 2019 and promoted to Assistant Professor Doctor position at Yaşar University in 2021. He has published articles in leading periodical journals indexed by SCI-Expanded, Scopus, and TR-Dizin.