A Hybrid Genetic Algorithm with Tabu Search for Optimization of the Traveling Thief Problem

Saad T Alharbi
Computer Science Department
Taibah University
Medina, Saudi Arabia

***Abstract*—Until now, several approaches such as evolutionary computing and heuristic methods have been presented to optimize the traveling thief problem (TTP). However, most of these approaches consider the TTP components independently, usually considering the traveling salesman problem (TSP) and then tackling the knapsack problem (KP), despite their interdependent nature. In this paper, we investigate the use of a hybrid genetic algorithm (GA) and tabu search (TS) for the TTP. Therefore, a novel hybrid genetic approach called GATS is proposed and compared with the state-of-the-art approaches. The key aspect of GATS is that TTP solutions are considered by firmly taking into account the interdependent nature of the TTP subcomponents, where all its operators are simultaneously implemented on TSP and KP solutions. A comprehensive set of TTP benchmark datasets was adopted to investigate the effectiveness of GATS. We selected 540 instances for our investigation, which comprised five different groups of cities (51, 52, 76, 100 and 150 cities) and different groupings of items, from 50 to 745 items. All types of knapsack (uncorrelated, uncorrelated with similar weights and bonded strongly correlated) with all different knapsack capacities were also taken into consideration. Different initialization methods were empirically investigated as well. The results of the computational experiments demonstrated that GATS is capable of surpassing the state-of-the-art results for various instances.**

Keywords*—Combinatorial; hybrid approaches; genetic algorithm; optimization; tabu search; TTP

I. Introduction

The traveling thief problem (TTP) is a benchmark problem recently introduced by [1]. It is an abstraction of real-world problems that consist of multiple components, such as vehicle routing problems and supply chain management. TTP has recently drawn researchers' attention, as its definition represents various aspects of real-world complexities. TTP combines two well-known problems, the traveling salesman problem (TSP) and the knapsack problem (KP). The underlying definition of the problem is that a thief has to visit a set of cities and pick some items from these cities and pack them in a knapsack, where each item has its own weight and value [1]. The challenging aspect of the problem is that the knapsack has a certain capacity, and the total weights of the picked items must not exceed this capacity; the thief also must pay rent for using the knapsack, the rent depending primarily on the total traveling time. Because the two problems (TSP and KP) are interdependent, the speed of the thief decreases when the knapsack gets heavier, which results in an increased total travel time and requires the paying of a higher rent. Therefore, the main objective of TTP is to maximize the total profit of the thief, comprising the total value of the picked items minus the rent of the knapsack.

Despite the TTP only recently being introduced, it has been rigorously considered. Various approaches have been introduced into the literature, adopting different types of techniques and algorithms. For instance, heuristics strategies are among the widely adopted methods for solving TTP, as seen in [2-6]. Searching for the best solutions in such approaches typically involves using classical greedy routines where an initial solution is generated and is iteratively improved. However, adopting heuristics to solve TTP can be computationally complex, especially with a large number of instances [7]. Evolutionary approaches such as genetic algorithms (GAs) and genetic programming have also been adopted for roughly solving the TTP, as in [6, 8-14]. The majority of these approaches try to improve TTP solutions by considering each subproblem (i.e., TSP and KP) independently, despite the interdependence between the subcomponents. Evolutionary operators such as crossover and mutation are normally implemented on tours, and then one of the known packing heuristics is implemented to obtain the best packing plan for the best tours. However, the shortest tours do not necessarily guarantee that the optimal TTP solution will be achieved, due to the nonlinear relationship in the solution's objective function [15]. Other approaches that have occasionally been adopted in the literature to solve TTP include swarm intelligence approaches, such as the ant colony or the artificial bees colony [16, 17].

It has not yet been proven which type of approach is most applicable for solving TTP. Several local search algorithms such as those presented in [18, 19] have been introduced. Most importantly, Packing Routine and PACKITERATIVE, presented by [20], have become key strategies in the literature. The former starts by sorting items according to their weight and then picks the most profitable items. The latter approach is considered an enhanced version of the former, working in the same way but in an iterative manner with some exponent values. Bitflip and Insertion are two local search operators that are also regularly adopted for solutions generated by the previous two methods, to achieve optimal solutions.

TABLE I. MATHEMATICAL REPRESENTATION OF TTP

Equation	No	Description
$v_{x_i} = v_{max} - C * w_{x_i}$	(1)	The speed (velocity) of the thief, where C is a constant value calculated using Equation 2, and w_{x_i} is the weight of the knapsack at the current city
$C = \frac{(v_{max} - v_{min})}{W}$	(2)	v_{max} is the maximum velocity , v_{min} minimum velocity, W is the total wieght of the knapsack
$g(z) = \sum_{m} p_m * z_m$	(3)	The total value of the collected items, where P_m is the profit of the item and z_m is a binary value indicating whether the item is available at a particular city (0 denotes that the item is unpicked. while 1 indicates that it is picked)
$f(x,z) = \sum_{i=1}^{n-1} t_{x_i,x_{i+1}} + t_{x_n,x_1}$	(4)	The total travel time from city x_i to x_{i+1}
$G(x,z) = g(z) - R * f(x,z)$	(5)	The objective value (i.e., the total profit gained by the thief)
$t_{x_i,x_{i+1}} = \frac{d_{x_i,x_{i+1}}}{v_{x_i}}$	(6)	The travel time between two cities, where $d_{x_i,x_{i+1}}$ is the distance between city x_i and x_{i+1}

In this paper, we introduce a hybrid GA [21, 22], called GATS, using one of the well-known local search methods, tabu search (TS) [23-25]. GA has been proven powerful in optimizing various types of problems in different domains, such as machine learning [26], network traffic control [27] and industry [28]. Similarly, TS has also been successfully adopted, both alone and hybridized with other approaches, for solving different problems in various fields, as in [29-31]. In fact, to our knowledge, TS has been hybridized with GA for solving various optimization problems, but the hybridization has not been adopted for TTP.

The contribution of this article to the literature is therefore twofold. First, it introduces a novel hybrid approach developed specifically for TTP. The key aspect of this approach is that TTP solutions are considered by firmly taking into account the interdependent nature of the TTP subcomponents. The adopted operators of the proposed approach are simultaneously implemented on tours and packing plans in the process of solution generation; specifically, GA operators, such as crossover and mutation, are implemented to modify tours, while TS is devoted to seeking the best corresponding packing plan. Second, the paper also presents a wide-ranging study taking into account different aspects of TTP, where the performance of the proposed approach was investigated on 540 datasets. These datasets differed in aspects such as size, knapsack capacity and type of knapsack. For instance, the number of cities in these datasets ranged from 51 to 150 cities, and the number of items ranged from 50 to 745 items. All types and capacities of knapsack were tested in this study, and the results were compared with the state-of-the-art approaches.

The rest of the paper is organized as follows. Section II briefly presents the definition of the TTP. In section III, a detailed description of the proposed approach is presented. The experimental design is discussed in section IV, and the proposed approach is tested on various instances and the experimental results are discussed in section VI . The paper concludes in section VII and directions for future work are outlined.

II. THE TRAVELING THIEF PROBLEM

The definition of TTP and its mathematical representation have been well introduced in the literature, for example in [1, 5, 20, 32]. Table 1 shows the mathematical representation of TTP, and we briefly summarize its underlying concept as follows:

- A thief has to travel among a set of cities n, visiting each city only once.
- The tour $\prod$ of the thief must start and end at the same city.
- The tour has a length that can be calculated from the distance $D = \{d_{ij}\}$ between cities.
- Each city contains some items m where each item has weight w_k and value p_k .
- The thief holds a knapsack that has a specific capacity W and a rent R.
- The thief is required to pick the most profitable items from cities during his tour, where the total weight of the picked items must not exceed the knapsack capacity.
- The knapsack rent R is based on the time unit.
- The speed of the thief depends on the knapsack weight, where the thief gets slower when the knapsack becomes heavier.
- The total profit gained by the thief is the total value of the picked items minus the rent.
- The ultimate objective is finding a tour $\prod$ and a packing plan z that maximize the total profit gained by the thief

III. DEVELOPMENT OF THE PROPOSED APPROACH

The aim of this paper is to empirically investigate the use of the GA to solve the TTP. Specifically, the hybridization of the GA with one of the well-known metaheuristic search

algorithms, TS, is investigated. Therefore, a hybrid GA with a TS approach, called GATS, is proposed. Figure 1 shows the flow chart of this proposed approach. Its key modules are described in detail in the following sections. For instance, Section A shows the adoption of GA and its operators, while Tabu Search C shows how TS was employed in the proposed approach to the TTP.

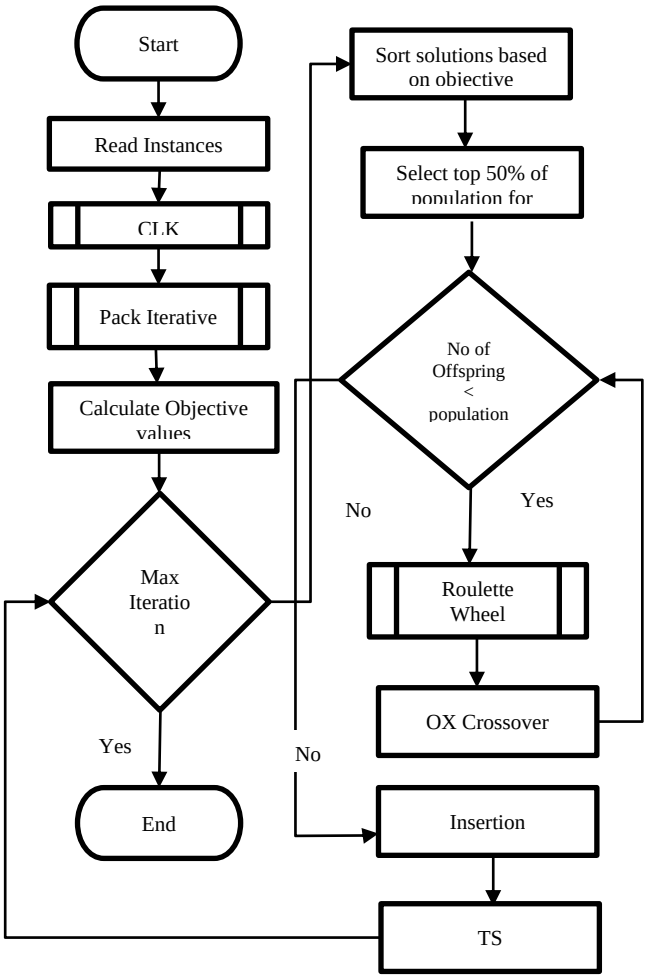

Fig. 1. GATS Flow Chart.

A. Genetic Algorithm for TTP

1) Coding

Tours are encoded directly, with integer numbers indicating city indices in sorted enumeration, starting from a city and ending with the same city. Figure 2 shows an example of a tour consisting of five cities, starting from city 4, traveling to cities 1, 3, 5 and 2 and then returning to the starting city.

n_1	n_2	n_3	n_4	n_5	n_1
4	1	3	5	2	4

Fig. 2. Tour Representation.

Similarly, a packing plan is encoded with an integer enumeration, where the length of the chromosome indicates the number of items, and the integer value in each position represents the number of the city from which the item was picked. Unpicked items are denoted by zero. Figure 3 shows a knapsack solution with 10 items (2 items per city), consisting of three picked items, 2, 3 and 9, at cities 1, 3 and 5, respectively.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
0	1	3	0	0	0	0	0	5	0

Fig. 3. Packing Plan Representation.

$\prod_1$	9	2	3	8	4	5	6	1	7	9
$\prod_2$	4	5	2	1	8	7	6	9	3	4
Step 1										
$\prod c_1$	0	0	0	1	8	7	6	0	0	0
$\prod c_2$	0	0	0	8	4	5	6	0	0	0
Step 2										
{9,2,3,4,5}										
$\prod c_1$	9	2	3	1	8	7	6	4	5	0
{2, 1,7,9,3}										
$\prod c_2$	2	1	7	8	4	5	6	9	3	0
Steps 3 and 4										
$\prod c_1$	9	2	3	1	8	7	6	4	5	9
$\prod c_2$	2	1	7	8	4	5	6	9	3	2
Step 5										

Fig. 4. An Example of Crossover.

2) Initialization

GA starts with sets of routes and their corresponding packing plans, which are considered to be the initial candidate solutions (i.e., population). Various strategies of initialization were adopted and tested for this paper. Tours were either randomly or using the Chained Lin-Kernighan heuristic (CLK) [33]. The impact of each method was tested, and the results will be discussed in Section VI. The knapsack plan, on the other hand, was initialized using of one the well-known heuristics of TTP, which is PACKITERATIVE [20].

3) Crossover

An order crossover (OX) operator [34] was applied on tours to generate new solutions throughout the algorithm iterations. OX is one of the operators successfully used in combinatorial optimization problems, especially with TSP. In OX, a part of one parent is copied to the child. In GATS, two candidate solutions (tours) are selected using the well-known roulette wheel selection [35] method, which depends mainly on fitness values. Then OX is implemented as follows:

1) Two random positions in each tour are selected that would be considered the starting and end of the part that will be copied to the new offspring.

2) The selected part in the first tour $\prod_1$ is copied to the second offspring $\prod c_2$, while the sub-tour in $\prod_2$ is copied to the first offspring $\prod c_1$.

3) Cities in $\prod_1$ that do not exist in $\prod c_1$ are recorded in the same order in which they occur in $\prod_1$.

4) Blank positions in $\prod c_1$ are filled, in order, with the cities recorded from $\prod_1$.

5) The last city of $\prod c_1$ is updated to be the same as the first one.

6) Steps 3–5 are repeated for $\prod c_2$.

Figure 4 shows an example of new offspring generation using the OX crossover. Positions 4 and 7 are selected. Accordingly, then, the sub-tour {8, 4, 5, 6} from $\prod_1$ is copied

to the second offspring $\prod c_2$, while $\prod c_1$ contains the sub-tour {1, 8, 7,6} from $\prod_2$. Blank positions in $\prod c_1$ are filled with cities {9, 2, 3, 4, 5}, in that order, because they occur in that order in $\prod_1$. Finally, the last city in $\prod c_1$ is updated to be the same as the starting city of the tour. Similarly, blank positions in $\prod c_2$ are filled with cities {2, 1, 7, 9, 3}, in that order, as they occur in

Algorithm 1: OX crossover pseudocode

$P1 \leftarrow \Pi_1$ $P2 \leftarrow \Pi_2$;
$Pos1 \leftarrow r1$ $Pos2 \leftarrow r2$;
where $r1, r2$ are random positions in Π_1&Π_2 within [2,n-1] such that $Pos1 \neq Pos2$
Initialize offspring
$child1 \leftarrow \phi$; $child2 \leftarrow \phi$;
$child1(Pos1toPos2) \leftarrow P2(Pos1toPos2)$;
$child2(Pos1toPos2) \leftarrow P1(Pos1toPos2)$;
Remaining cities from P1, $C_1 \leftarrow P1 - child1$;
Remaining cities from P2, $C_2 \leftarrow P2 - child2$;
$child1 \leftarrow \phi$;
$child2 \leftarrow \phi$;
foreach *city in* C_1 **do**
  for $i = 1 \leftarrow n$ **do**
    if $child1 == \phi$ **then**
      $child1 = C_1$
    end
  end
end
foreach $cityinC_2$ **do**
  for $i = 1 \leftarrow n$ **do**
    if $child2 == \phi$ **then**
      $child2 = C_2$
    end
  end
end
Adjusting last city to be the same of the first
$child1(1:n) \leftarrow child1(1:1)$;
$child2(1:n) \leftarrow child1(1:1)$;

Fig. 5. OX Crossover Pseudocode.

$\prod_1$, *and then the last city is updated to be the same as the starting city of the tour.* Figure 5 *shows the pseudocode of the implemented OX.*

4) Mutation

Opt-Mutation or insertion [34] was applied on the tours in GATS. Here, a predefined mutation probability determines the number of candidate solutions (tours) that will undergo the mutation process. Let us say, for example, that the mutation probability was set at 0.1, and the population (number of candidate solutions) is 100. Then insertion will only be performed on 10 tours, selected randomly. A randomly selected city is inserted into a randomly selected position in the tour. Figure 7 shows an example of insertion mutation on $\prod_2$. Notice that city 8 ($i = 5$) was selected to be moved to be the third city ($k = 3$), and subsequent cities are shifted. Consequently, $\prod_m$ is obtained. Figure 6 shows the insertion pseudocode.

B. Item Packing

In each iteration, after the crossover and mutation operators have performed on tours, the well-known PACKITERATIVE routine [20] is performed on each tour to generate its corresponding packing plan. PACKITERATIVE is well documented in [20], demonstrating good performance results. The main idea behind it is to pack the most profitable items into the pack by sorting them based on a score calculated based on the items' weights. Items are sorted based on their scores, and the algorithm sequentially checks whether adding an item increases the total profit obtained by the thief. After the generation of packing plans for all tours, the proposed GA calculates the objective value of each candidate solution using equation 5.

Algorithm 2: Insertion pseudocode

$P1 \leftarrow \Pi$;
$Pos1 \leftarrow r1$ $Pos2 \leftarrow r2$;
where $r1, r2$ are random positions in Π within [2,n-1] such that $Pos1 \neq Pos2$
Initialize offspring
$TempOfSpring \leftarrow \Pi$;
$TempOfSpring(Pos1) \leftarrow P1(Pos2)$;
foreach *city bewteen Pos1 and Pos2 in TempOfSpring* **do**
  shift city one postition forward;
end
$P \leftarrow TempOfSpring$;

Fig. 6. Insertion Pseudocode.

$\prod_2$	4	5	2	1	8	7	6	9	3	4
$\prod_m$	4	5	8	2	1	7	6	9	3	4

Fig. 7. An Example of Insertion.

C. Tabu Search

Tabu search is one of the known metaheuristic algorithms that has been efficiently employed in solving various optimization problems, especially combinatorial ones. For instance, it has been extensively used to solve classical job-shop scheduling, as in [36] [37], as well as various environmental problems, such as power system planning [38] and transportation [39]. It has also been adopted in optimizing different aspects of recent technology trends such as big data [40]. The main idea of TS is to generate new solutions from the neighborhood of a current solution. Similar to other metaheuristic algorithms, a certain number of iterations are performed to generate new solutions. However, TS selects the best of these. The most important feature of TS is the tabu list, which is used to store subsets of solutions that are not allowed to be visited again, as they would bring the search to areas that have already been visited. This feature helps the algorithm to avoid cycling and getting trapped by local optima. In our proposed GA, TS was adopted on packing plans for each tour to ensure that the best items were picked for the tour. In each iteration of the GA, after performing the crossover and mutation operators, the TS method is called for each tour. The current tour and the initial packing plan are passed to this method. The length of the tabu list is randomly initialized based on the number of items, and the method terminates when it reaches the maximum number of iterations value where the best packing plan (i.e., with highest objective value) is returned. The Bitflip routine introduced by [20] was adopted in the proposed algorithm as a method for searching the

neighborhood (new packing plans). However, it was modified to adapt to the proposed algorithm, where all items in the pack are checked in order, and the status of each item is flipped where picked items become unpicked and vice versa. After flipping each item, the resulting packing plan is returned to the TS, and the objective value is calculated. Figure 8 shows the algorithm of TS.

Algorithm 4: Tabu Search pseudocode

```
Sbest ← ϕ;
TSList ← ϕ;
S − Tour ← Π;
S − Pack ← P;
while i < Maxiteration do
    foreach city item in S-Pack do
        NewPack = Bitflip(S − Pack, curentItem);
    end
    if NewPack Not in TSList AND Obj(NewPack) > Obj(S − Pack) then
        Sbest ← NewPack;
    end
    TSList.ADD(Sbest);
end
```

Fig. 8. TS Pseudocode.

TABLE II. INSTANCE CHARACTERISTICS

Parameters	Description
Number of cities	Ranged from 51 to 85900
Type of knapsack problem	- Uncorrelated (U) - Uncorrelated with similar weights (USW) - Bounded strongly correlated (BSC)
Items per city (F)	1, 3, 5 and 10
Knapsack Capacity (C)	Ranged from 1 to 10

TABLE III. BEST OBJECTIVE VALUES OBTAINED USING THE RANDOM AND CLK INITIALIZATIONS

	U		USW		BSW	
C	Random	CLK	Random	CLK	Random	CLK
1	4195	2251	-2770	1323	25033	3714
2	7718	1342	-3674	3410	22479	4639
3	5432	9111	-7162	3587	13055	6999
4	7143	1052	-5433	2260	10199	8866
5	19972	3892	-21274	884	-2714	5448
6	-19612	3737	-3674	1409	-28883	7678
7	-21893	4764	-3674	1478	-27672	9147
8	-19245	6019	-7162	2105	-33525	8607
9	-19611	6922	-21274	3609	-38895	8336
10	-21833.6	3965	-23798	5451	-35153	12386

IV. EXPERIMENTAL DESIGN

In order to perform an investigation on a TTP, the set of instances defined by [4] should be considered for performance evaluation. These instances were developed in such a way that the two sub-problems (i.e., TSP and KP) were considered. The total number of instances in this set is 9720, and the instances

have different characteristics based on different parameters, for example, the numbers of cities and items. Most of these characteristics are derived from the TSP library dataset [41]. Table 2 highlights the characteristics of the TTP instances. Because of the complexity of performing an experiment on the entire set of instances, a collection of 540 instances was selected for our investigation. These instances consisted of five different groups of cities, 51, 52, 76, 100 and 150 in number, as well as three different numbers of items per city (F), 1, 3, and 5. The selected instances also consisted of the three types of KPs (uncorrelated [U], uncorrelated with similar weights [USW] and bounded strongly correlated [BSC]) and all varieties of knapsack capacity (C). Our implementation was conducted using MATLAB R2014a, and all computations were performed on machines using an Intel Core i7-4790S 3.20 GHz processor and 12 GB RAM, running Windows 8.

Before performing the experiment, several parameters were determined. For instance, the maximum running time for an instance was 10 minutes. In addition, due to the randomization of evolutionary approaches, each instance would be tested 10 independent times. The results obtained from our experiment were compared with the best objective values obtained in the literature. At the beginning of the experiment, several variations on the proposed approach were empirically tested; for example, we tested the effect of the random initialization of tours and investigated the performance of GA with CLK initialization. The aim of these investigations was to adopt the best obtained methods in our approach. Then the proposed approach was tested, and the results were compared with some of the state-of-the-art approaches.

V. INITIALIZATION METHOD

In order to investigate the effect of the tour initialization method on the quality of the obtained solutions, two methods were tested, the basic random tour initialization and the CLK heuristic. These methods were implemented within a classic GA for solving TTP. The performance of the two methods was compared on 30 instances with relatively small numbers of cities and items. Surprisingly, the basic random method obtained higher objective values in some instances when compared with CLK, especially with a small knapsack capacity ranging from 1 to 3 and particularly with U and BSC knapsacks (see Table 3). In order to obtain an accurate result, a normalized objective value was calculated for each method in all instances, taking into consideration the values obtained from each run. This normalized value was calculated by taking the ratio between the best objective value for an instance and the average of the objective values for all runs for this instance. Figure 9 shows the results of the comparisons for the three types of knapsack (U, USW and BSC). It is clear that the CLK used for tour initialization is significantly better than the basic random method. Figure 10 also shows an example of a tour generated for an optimal solution obtained by each method; the obtained tour using CLK was significantly (more than 50%) shorter than that obtained by random means.

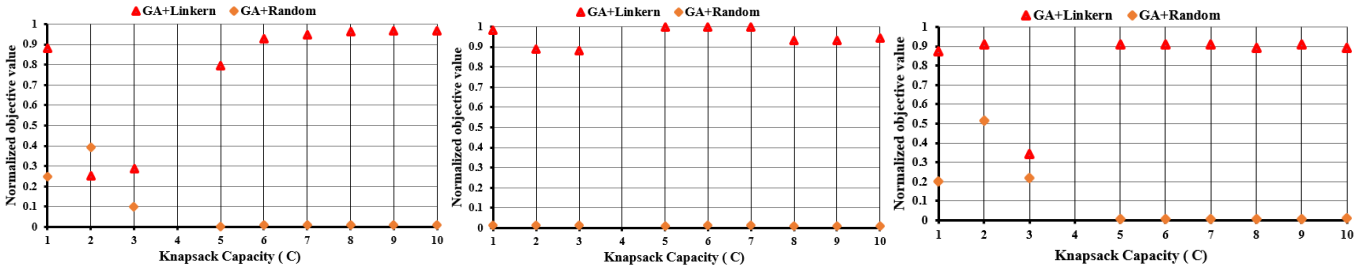

Fig. 9. Normalized objective values for random and CLK initializations in a) U, b) USW and c) BSW.

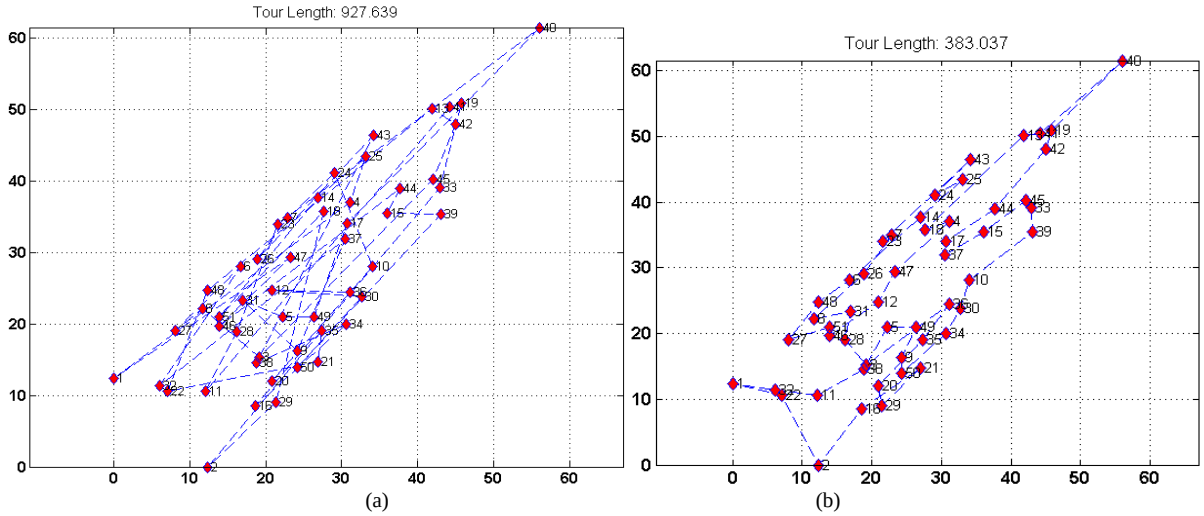

Fig. 10. Tours of an optimal TTP solution obtained by a) random and b) CLK initializations.

TABLE IV. BEST OBTAINED OBJECTIVE VALUES FOR U KNAPSACK AND F=1

Instance	*n*	m	C									
				1	2	3	4	5	6	7	8	9	10
eil51	51	50	GATS	**2251**	**13420**	**9111**	**1596**	**3892**	**3737**	**3965**	**4764**	**6019**	**6922**
			EA	1720	3983	4162	1444	2997	3313	3201	3929	5297	6130
			RLS	1533	3983	4162	1444	2963	3313	3201	3929	5297	6130
berlin52	52	51	GATS	1354	2261	4335	3934	4265	5582	6280	6944	7752	7545
			EA	**2330**	**3220**	**4748**	**3963**	**4141**	**5372**	**6541**	**7532**	**8060**	**7979**
			RLS	2003	**3220**	**4748**	**3963**	**4141**	**5356**	**6541**	**7532**	**8060**	**7979**
eil76	76	75	GATS	**4528**	**6101**	**4467**	**4663**	**6355**	**7425**	**7161**	**7134**	**9013**	**9457**
			EA	3727	5173	4176	3958	5292	6175	5968	6078	7768	8825
			RLS	3412	5012	4114	3958	5292	6150	5968	6061	7768	8825
kroA100	100	99	GATS	**2790**	**6180**	**6246**	**10318**	**11362**	**12001**	**12508**	**14712**	**15662**	**16725**
			EA	1410	4437	5359	8104	8255	9069	9560	12084	13072	14141
			RLS	1193	4409	5357	8104	8255	9069	9560	12084	13072	14141
pr124	124	123	GATS	**1953**	**7410**	11031	14092	13299	**16996**	**16329**	**17096**	**18470**	**19378**
			EA	2180	7214	**11406**	**15324**	**14092**	16766	15436	16316	17659	18544
			RLS	2180	7175	**11406**	15315	**14092**	16766	15406	16316	17659	18546
ch150	150	149	GATS	3007	8645	11335	10207	12107	14304	15926	**17621**	**18860**	**20595**
			EA	**5033**	**10805**	**12964**	**10972**	**12402**	**14767**	**16182**	17475	18271	20171
			RLS	4982	**10805**	12890	**10972**	12393	**14767**	**16182**	17475	18271	20171

TABLE V. BEST OBTAINED OBJECTIVE VALUES FOR USW KNAPSACK AND F=1

Instance	*n*	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	50	GATS	**1323**	**3490**	3864	2354	1281	1726	**2183**	**3060**	**4114**	**5909**
			EA	1238	3318	**3890**	**2531**	**1373**	**1933**	2133	2977	3894	5420
			RLS	473	2035	3182	2486	**1373**	**1933**	2097	2977	3894	5404
berlin52	52	51	GATS	**1356**	**2605**	**3502**	**2890**	3647	3623	4345	5927	5785	6676
			EA	**1350**	2516	3254	2763	**3975**	**3847**	**4817**	**6716**	**6678**	**7231**
			RLS	754	1503	1831	2663	3965	3796	**4817**	**6716**	6674	7226
eil76	76	75	GATS	**1344**	1574	**2212**	**2481**	3556	1468	**3960**	**5496**	**7687**	**8166**
			EA	1227	**1717**	1847	2449	**3560**	**1601**	3722	5276	7360	7807
			RLS	-27.9	1470	1847	2284	3545	**1601**	3722	5282	7360	7807
kroA100	100	99	GATS	**2247**	**5784**	**7538**	**8276**	7821	**10097**	**11778**	**12903**	**14490**	**15607**
			EA	1642	5066	6994	7855	**7904**	9925	11187	11675	13146	1642
			RLS	482	4510	6994	7775	7893	9915	11187	11675	13146	482
pr124	124	123	GATS	**4151**	**6669**	**6594**	8507	**10743**	12619	14325	16519	16814	**17950**
			EA	3833	6474	6224	**8725**	10622	**12852**	**14533**	**16566**	**17010**	17942
			RLS	2717	6174	6152	8661	10593	12845	**14533**	**16566**	**17002**	17942
ch150	150	149	GATS	3187	3893	4680	6380	10005	11523	12555	16806	17504	19475
			EA	2786	4033	5110	6973	10034	12104	12919	16638	17048	18445
			RLS	696	4033	4914	6954	10019	12112	12919	16638	17048	18445

TABLE VI. BEST OBTAINED OBJECTIVE VALUES FOR BSC KNAPSACK AND F=1

Instance	*n*	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	50	GATS	**7046**	**5705**	**6210**	**5633**	**5763**	**7752**	**9209**	**8619**	**8431**	**12604**
			EA	3669	5076	4918	5528	3920	5701	7120	5980	5576	9613
			RLS	2077	3922	4383	5037	3918	5497	7103	5978	5576	9574
berlin52	52	51	GATS	3257	4063	4206	6421	8785	6834	8944	**11481**	**13777**	**10591**
			EA	**3927**	**5634**	**5393**	**7419**	**9884**	**7900**	9706	9048	11131	7875
			RLS	3104	5319	4978	7014	9524	7524	**9325**	9035	11131	7868
eil76	76	75	GATS	2505	5591	9698	**8990**	**6082**	**10586**	**9868**	**12179**	**9773**	**12999**
			EA	**3353**	**6220**	**9750**	8517	5469	9961	9779	11001	9238	10033
			RLS	2965	4891	9216	7855	5410	9678	9757	10973	9009	9920
kroA100	100	99	GATS	**3519**	**8510**	**12545**	**16345**	**21823**	**24768**	**27075**	**27222**	**27545**	**27733**
			EA	2936	7607	8463	10024	13983	14752	15548	14710	14665	14740
			RLS	3185	7233	7632	9824	13636	14728	15285	14456	14631	14762
pr124	124	123	GATS	**5279**	**9868**	**15891**	**22874**	**26369**	**27065**	**31564**	**31509**	**31748**	**28895**
			EA	4918	5977	8981	12518	15501	16410	21540	22435	23838	21770
			RLS	4610	5584	8744	11769	14618	16237	21517	22434	23838	21748
ch150	150	149	GATS	6003	9361	**14416**	**19712**	**21516**	**21474**	**26198**	**30004**	**28441**	**21983**
			EA	**7060**	**9930**	13363	17123	18267	17915	22443	25924	23967	17133
			RLS	5540	7890	13138	16969	18205	17915	22446	25736	23898	17139

TABLE VII. BEST OBTAINED OBJECTIVE VALUES FOR U KNAPSACK AND F=3

Instance			C										
	n	m		1	2	3	4	5	6	7	8	9	10
eil51	51	150	GATS	**12757**	6862	**10408**	**11631**	**10125**	**11404**	**12407**	**14898**	**18978**	**20940**
			EA	5866	8190	9496	6704	7405	9110	10039	12848	15883	18783
			RLS	5644	**8224**	9548	6706	7367	9110	10039	12848	15883	18783
berlin52	52	153	GATS	5653	6243	11573	13346	14434	12841	14244	15958	16055	18265
			EA	**9550**	**9446**	**14467**	**15717**	**15453**	**13783**	**16367**	**18420**	**18479**	**20755**
			RLS	**9550**	**9446**	**14467**	**15717**	**15453**	**13783**	**16367**	**18420**	**18479**	**20755**
eil76	76	225	GATS	9885	10755	9793	12910	**15712**	**18468**	**20719**	**21363**	**26461**	**26235**
			EA	**11215**	**11577**	**10285**	**13518**	14881	18011	19134	21004	25645	25696
			RLS	11153	11567	**10285**	**13518**	14881	18011	19135	21004	25645	25696
kroA100	100	297	GATS	5958	15285	22481	25766	**29504**	**31782**	**34282**	**40355**	**40065**	**45303**
			EA	**10305**	**19318**	**25500**	**26306**	26622	28964	32229	38375	37576	41997
			RLS	10294	**19318**	**25500**	**26306**	26622	28959	32229	38375	37576	41997
pr124	124	369	GATS	9438	15284	25219	30209	36136	39723	42801	47099	51937	57657
			EA	**16303**	**23105**	29438	37560	42888	**42589**	**43700**	**49025**	**52550**	**56691**
			RLS	**16303**	23068	**29462**	**37572**	**42899**	**42589**	**43700**	**49029**	**52550**	**56691**
ch150	150	477	GATS	5086	17325	26264	28929	31472	37347	40510	46622	51966	55714
			EA	16721	28115	35162	34578	37215	41684	44664	49488	54138	57236
			RLS	16721	28136	35162	34572	37217	41684	44683	49488	54138	57236

TABLE VIII. BEST OBTAINED OBJECTIVE VALUES FOR USW KNAPSACK AND F=3

Instance	n	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	150	GATS	**4116**	6622	8109	5967	6789	8302	9943	**12327**	**13402**	**16159**
			EA	3798	**7325**	**9966**	**6807**	**7657**	**8472**	**10397**	11070	12804	14844
			RLS	2545	7208	9879	6774	7653	**8452**	**10397**	11070	12804	14841
berlin52	52	153	GATS	3708	4905	3089	2231	5447	7705	10469	14224	15148	19844
			EA	**5122**	**8317**	8492	**7680**	**11501**	**12556**	**15218**	**18025**	**20323**	**23209**
			RLS	3002	5232	**8505**	**7680**	**11501**	**12556**	**15211**	**18025**	**20323**	**23209**
eil76	76	225	GATS	4860	3644	4897	6716	10454	**8483**	13552	**16530**	**21883**	**24085**
			EA	**5411**	**4969**	6958	**8684**	**10717**	8819	**14067**	16440	20802	23165
			RLS	5112	4898	**6971**	8653	**10717**	8814	**14067**	16440	20802	23165
kroA100	100	297	GATS	**8155**	11858	14149	15844	18600	22103	25725	**31424**	**39119**	**42713**
			EA	7145	**13163**	**15424**	17418	**20282**	**23721**	**27051**	31370	38089	39463
			RLS	6886	13020	15387	**17421**	**20282**	**23721**	**27051**	31370	38089	39463
pr124	124	369	GATS	11919	15691	20015	22816	27333	32262	36064	**43836**	**47830**	**55714**
			EA	**13643**	**20347**	**25838**	**27580**	**32387**	**37483**	**40047**	43151	47270	52732
			RLS	12388	20190	25787	27575	**32387**	37480	**40047**	43150	47272	52732
ch150	150	477	GATS	6144	8269	10327	13749	19747	23968	27928	35052	41202	47371
			EA	**9117**	**14249**	**17855**	**23513**	**30111**	**34003**	**35413**	41297	**45660**	**49657**
			RLS	5998	14221	17838	23497	30094	**34003**	**35413**	**41301**	**45660**	**49657**

TABLE IX. BEST OBTAINED OBJECTIVE VALUES FOR BSC KNAPSACK AND F=3

Instance	n	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	150	GATS	**14794**	10763	15472	**21741**	22798	20715	**21456**	**21686**	**25276**	**27136**
			EA	6664	**12796**	**15786**	21558	**22913**	**21012**	21188	20677	23414	24114
			RLS	4487	11673	15037	21337	22366	20957	21154	20647	23382	24124
berlin52	52	153	GATS	5590	10577	12932	21148	20540	26319	25721	**39668**	**35159**	31443
			EA	**9652**	**16485**	**19684**	**28125**	**26488**	31150	**30211**	34029	30659	31700
			RLS	6600	14304	18337	27379	26344	**31352**	29934	34021	30509	**31784**
eil76	76	225	GATS	6393	14296	24487	**26867**	**31103**	**38152**	**40316**	**43023**	**39945**	**45623**
			EA	**7644**	**14692**	22538	22502	25785	31633	32611	35101	30232	34183
			RLS	6806	14687	**22561**	22468	25669	31571	32584	35138	30226	34190
kroA100	100	297	GATS	12447	**29433**	**42359**	**52054**	**62991**	**69103**	**72403**	**68952**	**63789**	**67504**
			EA	**13725**	25080	32678	37156	43668	46954	48207	42549	36035	39067
			RLS	12600	24664	32720	37043	43618	46969	48211	42544	36017	39076
pr124	124	369	GATS	13088	**35206**	**52693**	**74355**	**83716**	**97891**	**102660**	**101580**	**90070**	**98197**
			EA	**16217**	29193	41810	57901	64792	79009	84810	82360	71276	78945
			RLS	15539	28777	40657	57477	64361	78972	84866	82291	71265	78899
ch150	150	477	GATS	16124	35081	**53125**	**60824**	**68400**	**75684**	**76558**	**71987**	**75781**	**69198**
			EA	**19600**	**37596**	52111	53592	57974	62374	61093	54987	55871	49642
			RLS	16039	36403	52106	53533	57939	62419	61021	54922	55891	49642

TABLE X. BEST OBTAINED OBJECTIVE VALUES FOR U KNAPSACK AND F=5

Instance	n	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	250	GATS	6256	13322	13482	15624	**18430**	**20628**	**19378**	**22934**	**26390**	**31520**
			EA	10683	16345	13885	**16367**	16375	19057	17871	21961	25208	29508
			RLS	**10688**	**16380**	**13887**	16366	16375	19057	17871	21963	25208	29508
berlin52	52	255	GATS	9805	13413	19636	22637	26302	27357	31294	30881	29606	35997
			EA	**18049**	**21247**	**26153**	**29547**	31348	**32109**	**36819**	**35942**	**36584**	**39386**
			RLS	18029	21222	**26153**	**29547**	**31354**	**32109**	**36819**	**35942**	**36584**	**39386**
eil76	76	375	GATS	7268	13883	16258	18624	24665	27517	31973	35890	42916	**46714**
			EA	**13023**	18070	**19554**	**22349**	**26986**	28412	**32510**	**36327**	**43008**	44770
			RLS	**13023**	**18080**	**19554**	**22349**	**26986**	**28424**	**32510**	**36327**	**43008**	44770
kroA100	100	495	GATS	4084	18049	31563	41507	50054	**52219**	**55082**	**64904**	**69106**	**78315**
			EA	**13670**	**27789**	**38737**	**45781**	51076	51588	53633	62457	66117	73998
			RLS	13674	**27789**	38726	45764	**51086**	51592	53633	62457	66115	73999
pr124	124	615	GATS	21880	32241	40368	45632	56115	65422	69301	76267	**91335**	**101453**
			EA	28416	**40811**	**45873**	**58503**	66018	70346	**76155**	**78626**	84805	91399
			RLS	**28422**	**40811**	**45873**	**58503**	**66029**	**70351**	**76155**	**78626**	84805	91399
ch150	150	745	GATS	22475	34033	37762	39908	57564	58785	63270	56226	**85023**	**92916**
			EA	33047	50046	**55532**	58683	**58739**	59979	**64561**	72084	79610	87000
			RLS	**33052**	**50049**	**55532**	**58688**	**58739**	**59985**	**64561**	**72085**	79610	87000

TABLE XI. BEST OBTAINED OBJECTIVE VALUES FOR USW KNAPSACK AND F=5

Instance	n	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	250	GATS	4835	7312	8816	9449	10831	14091	17749	17645	21245	28107
			EA	**5285**	9121	**11140**	**10455**	**12878**	**15574**	**18683**	**18184**	**21343**	**26483**
			RLS	3811	**9206**	11093	**10445**	**12878**	**15574**	**18683**	**18184**	**21343**	**26483**
berlin52	52	255	GATS	6097	6775	5847	6338	9443	15392	18098	22447	28650	33515
			EA	**9178**	**13505**	15700	**17132**	20587	**22873**	**26359**	31739	**36054**	**40537**
			RLS	6079	11347	**15714**	17085	**20590**	**22873**	26363	**31741**	**36054**	**40537**
eil76	76	375	GATS	8245	5761	10534	12509	14123	14392	20804	24840	30082	38073
			EA	**9883**	10290	13293	**18532**	19944	18801	**24721**	**27858**	**32213**	**36553**
			RLS	9867	**10368**	**13329**	18508	**19957**	**18806**	**24721**	**27858**	**32213**	**36553**
kroA100	100	495	GATS	7875	13426	18677	23964	29733	36743	43030	48959	**63755**	**70255**
			EA	**10530**	21104	**27042**	**33303**	**38855**	45102	**49565**	**52726**	63657	66751
			RLS	10401	**21145**	26978	**33303**	38846	**45105**	**49565**	**52726**	63657	66751
pr124	124	615	GATS	13735	18155	23084	29777	34953	52432	47900	58064	65419	94974
			EA	19906	26311	33455	37740	44301	52962	60710	65982	74340	85794
			RLS	16085	26311	33444	37740	44299	52961	60710	65982	74340	85795
ch150	150	745	GATS	7397	10199	13604	29700	36421	44233	**62350**	63635	65498	**95766**
			EA	**15609**	**24436**	29324	33000	**40922**	**47562**	55701	**64278**	**74430**	83275
			RLS	10107	24427	**29341**	**33004**	40919	47560	55701	**64278**	**74430**	83275

TABLE XII. BEST OBTAINED OBJECTIVE VALUES FOR BSC KNAPSACK AND F=5

Instance	n	m	C										
				1	2	3	4	5	6	7	8	9	10
eil51	51	250	GATS	8215	17551	29452	**33230**	**33350**	**40293**	**40203**	**44120**	**45228**	**43007**
			EA	**10397**	**19722**	**30181**	32102	30482	35749	34191	37373	37247	33697
			RLS	9173	18991	29998	31560	30497	35752	34230	37058	37209	33697
berlin52	52	255	GATS	8107	20247	32764	35018	41165	53501	56404	74836	63133	50526
			EA	**14528**	**28835**	**44516**	**48130**	**52655**	**63329**	65358	67873	**65713**	55935
			RLS	11398	24981	40838	48127	52559	63221	**65362**	**67882**	**65713**	**55952**
eil76	76	375	GATS	12387	28397	43974	**53637**	**63576**	**64705**	**72170**	**74670**	**73161**	**72539**
			EA	**14684**	**29705**	**41728**	45733	53648	52825	59137	60762	58755	57441
			RLS	14522	29765	41568	45454	53625	52853	59172	60658	58752	57435
kroA100	100	495	GATS	17871	**42303**	**62852**	**78784**	**97322**	**109370**	**122500**	**111540**	**103920**	**108070**
			EA	**20171**	37407	52409	60664	71889	80653	91242	77821	69296	73190
			RLS	19592	37101	52544	60497	71853	80656	91106	77849	69300	73190
pr124	124	615	GATS	24295	57282	90776	**113290**	**129060**	**142430**	**157270**	**143380**	**143720**	**132110**
			EA	52341	**79028**	**93081**	106169	115370	133307	118764	119662	107580	52341
			RLS	**52492**	78851	92919	106100	115370	133342	118773	119662	107549	52492
ch150	150	745	GATS	26247	61770	88715	**107560**	**121410**	**132560**	**132380**	**121960**	**129690**	**126520**
			EA	**34374**	**69088**	**91584**	99919	108119	114990	111872	98960	104834	100531
			RLS	31445	68036	91362	99873	108125	114974	111876	98962	104836	100525

VI. GATS RESULTS

The best obtained objective values for GATS were recorded and compared with the best obtained by two state-of-the-art approaches, EA and RLS. Tables 4–6 show the results for instances with one item per city and a knapsack capacity ranging from 1 to 10. Table 4 shows that GATS obtained higher objective values than EA and RLS for all knapsack capacities for the instances consisting of 51, 76 and 100 cities with uncorrelated (U) item weights. However, the results showed that GATS was not able to record higher objective values in instances consisting of 52 cities; it achieved better values in only two instances, with medium knapsack capacity

($C = 5\ and\ 6$). Nevertheless, the differences between the highest objective values obtained by the three approaches were not substantial in these instances; for example, EA and RLS achieved objective values, for the knapsack with $C = 4$, that were only 1% higher than GATS. GATS also outperformed RLS and EA in instances with a large number of cities (i.e., 124 and 150) and knapsacks with a high capacity (8, 9 and 10). With USW, Table 5 shows that GATS was able to achieve better objective values than EA and RLS, with the majority of instances having a small knapsack capacity (ranging from 1 to 3). Similarly, it outperformed them for most of the large-capacity instances, ranging from 8 to 10, except instances with 52 cities (i.e., berlin52). On the other hand, EA and RLS outperformed GATS with a medium KP, ranging from 4 to 7 in most instances, although the differences between the objective values obtained by the three approaches were not significant. For example, the best value reached by GATS with the instance consisting of 76 cities (i.e., eil76) and a knapsack capacity of 5 was 3556, whereas it was 3560 in RLS. In BSC, GATS surpassed RLS and EA in almost all instances, with medium- and large-capacity knapsacks ranging from 4 to 10, except in the instances with 52 cities (i.e., berlin52), where EA and RLS were able to achieve better objective values in the majority of the instances (see Table 6). GATS was also able to obtain better objective values with a small-capacity knapsack in various instances, particularly the ones for 51, 100, 124 and 150 cities, while the best objective values were achieved by EA in the rest of the instances for 52 and 76 cities. Tables 7–9 show the best objective values obtained from instances consisting of three items per city. Table 7 shows that GATS achieved better objective values in several uncorrelated instances with different knapsack capacities. For instance, it outperformed EA and RLS in all knapsack capacities in instances consisting of 51 cities (i.e., eil51). In contrast, lower objective values were achieved by GATS in all instances consisting of 52, 124 and 150 cities. Although the values obtained by GATS fluctuated among instances, it performed better with medium- and large-capacity knapsacks ranging from 5 to 10 instances containing 76 and 100 cities. The best objective values were achieved interchangeably with USW by the three approaches. However, GATS was able to surpass EA and RLS with large-capacity knapsacks, specifically when $C = 8, 9\ and\ 10$, in instances containing 51, 76, 100 and 124 cities. But EA and RLS achieved better values in the other two instances with the same knapsack capacities (see Table 8). Table 9 shows that GATS also obtained better objective values with BSC items, especially with large knapsack capacities ($C = 7, 8, 9\ and\ 10$).

As observed in Table 10, with uncorrelated items, GATS was unable to exceed the objective values obtained by EA and RLS when increasing the number of items to five per city. This became apparent with small and medium knapsacks ($C =$ 1 to 4). However, GATS revealed the highest objective values with large-capacity knapsacks in various instances. For instance, GATS outperformed RLS and EA in all instances consisting of large knapsacks, ranging from 6 to 10, for 51 and 100 cities (eil51 and kroA100). It also reached the highest objective value in most uncorrelated item instances with a knapsack capacity equal to 10, except for those that contained 52 cities (berlin52). But GATS was not able to outperform the two state-of-the-art approaches in all instances with USW, except instances with the largest knapsack capacity ($C = 10$). In fact, it recorded significantly higher objective values in four sets of various instances, particularly those composed of 51, 100, 124 and 150 cities (see Table 11). The performance of GATS for a BSC knapsack was notably better than that of the two state-of-the-art approaches in the majority of instances (see Table 12). However, this improvement became most evident with medium- and large-capacity knapsacks ranging from 4 to 10.

Based on the obtained results, the following findings were observed:

- The method used for tour initialization significantly affects the obtained solutions.
- Our proposed approach (GATS) performed better, in terms of objective values achieved, than two of the well-known state-of-the-art approaches (RLS and EA) in the majority of instances.
- GATS performed better, especially with instances of a large-capacity knapsack.
- GATS' performance significantly decreased when increasing the number of items and cities, particularly with a small-capacity knapsack.
- GATS had some issues with one set of instances, for berlin52, where it struggled to achieve better objective values in almost all instances.

VII. Conclusion

This paper investigates one of the recent NP-hard problems called the traveling thief problem, a multicomponent problem consisting of the two well-known problems TSP and KP. The optimization of TTP is challenging because of the interdependence between its components, where finding an optimal solution for one problem independently does not guarantee obtaining an optimal TTP solution. The aim of this paper was to investigate the use of hybrid GAs for the TTP. Therefore, we proposed a hybrid genetic approach with TS, a combination called GATS. The key aspect of GATS is that TTP solutions are considered by taking into account the interdependent nature of the TTP subcomponents. The performance of GATS was analyzed and compared with that of two state-of-the-art approaches, EA and RLS. A comprehensive set of TTP benchmark datasets was adopted in this experimental work, and 540 instances were selected for our investigation. These instances consisted of five different groups of cities, 51, 52, 76, 100 and 150 in number, as well as groups of items ranging in number from 50 to 745. The selected instances also consisted of the three types of KPs (U, USW and BSC) and all varieties of knapsack capacity (C). The obtained results were analyzed based on several factors, such as the type of knapsack, knapsack capacity and number of items per city.

The obtained results revealed that GATS was able to outperform EA and RLS in terms of objective values for several instances. This became more apparent with a large-capacity knapsack. However, some limitations of GATS were

observed, for example, that its performance significantly decreased when the number of items and cities increased, particularly with a small-capacity knapsack. It was also noted that GATS had some issues with one set of the datasets, berlin52, where it struggled to achieve better objective values in almost all instances. Therefore, in the future, further experiments should be conducted to tackle such issues. Larger numbers of instances should also be investigated.

REFERENCES

[1] Bonyadi, M.R., Z. Michalewicz, and L. Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. in 2013 IEEE Congress on Evolutionary Computation. 2013.

[2] Gupta, B.C. and V.P. Prakash. Greedy heuristics for the Travelling Thief Problem. in 2015 39th National Systems Conference (NSC). 2015.

[3] Martins, M.S.R., et al., HSEDA: a heuristic selection approach based on estimation of distribution algorithm for the travelling thief problem, in Proceedings of the Genetic and Evolutionary Computation Conference. 2017, ACM: Berlin, Germany. p. 361-368.

[4] Polyakovskiy, S., et al., A comprehensive benchmark set and heuristics for the traveling thief problem, in Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. 2014, ACM: Vancouver, BC, Canada. p. 477-484.

[5] Chand, S. and M. Wagner, Fast Heuristics for the Multiple Traveling Thieves Problem, in Proceedings of the Genetic and Evolutionary Computation Conference 2016. 2016, ACM: Denver, Colorado, USA. p. 293-300.

[6] El Yafrani, M., et al., A hyperheuristic approach based on low-level heuristics for the travelling thief problem. Genetic Programming and Evolvable Machines, 2017.

[7] Mei, Y., X. Li, and X. Yao. Improving Efficiency of Heuristics for the Large Scale Traveling Thief Problem. 2014. Cham: Springer International Publishing.

[8] Bonyadi, M.R., et al., Socially inspired algorithms for the travelling thief problem, in Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. 2014, ACM: Vancouver, BC, Canada. p. 421-428.

[9] Karder, J., et al. Solving the Traveling Thief Problem Using Orchestration in Optimization Networks. 2018. Cham: Springer International Publishing.

[10] Mei, Y., et al. Heuristic evolution with genetic programming for traveling thief problem. in 2015 IEEE Congress on Evolutionary Computation (CEC). 2015.

[11] Moeini, M., D. Schermer, and O. Wendt. A Hybrid Evolutionary Approach for Solving the Traveling Thief Problem. 2017. Cham: Springer International Publishing.

[12] Vieira, D.K.S., et al. A Genetic Algorithm for Multi-component Optimization Problems: The Case of the Travelling Thief Problem. 2017. Cham: Springer International Publishing.

[13] Wu, J., et al., Evolutionary Computation plus Dynamic Programming for the Bi-Objective Travelling Thief Problem. arXiv preprint arXiv:1802.02434, 2018.

[14] Lourenço, N., F.B. Pereira, and E. Costa. An Evolutionary Approach to the Full Optimization of the Traveling Thief Problem. 2016. Cham: Springer International Publishing.

[15] Mei, Y., X. Li, and X. Yao, On investigation of interdependence between sub-problems of the Travelling Thief Problem. Soft Computing, 2016. 20(1): p. 157-172.

[16] Alharbi, S.T., The Design and Development of a Modified Artificial Bee Colony Approach for the Traveling Thief Problem. International Journal of Applied Evolutionary Computation (IJAEC), 2018. 9(3): p. 32-47.

[17] Wagner, M. Stealing Items More Efficiently with Ants: A Swarm Intelligence Approach to the Travelling Thief Problem. 2016. Cham: Springer International Publishing.

[18] El Yafrani, M. and B. Ahiod, A local search based approach for solving the Travelling Thief Problem: The pros and cons. Applied Soft Computing, 2017. 52: p. 795-804.

[19] Araujo, R.P., et al., A novel List-Constrained Randomized VND approach in GPU for the Traveling Thief Problem. Electronic Notes in Discrete Mathematics, 2018. 66: p. 183-190.

[20] Faulkner, H., et al., Approximate Approaches to the Traveling Thief Problem, in Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. 2015, ACM: Madrid, Spain. p. 385-392.

[21] Srinivas, M. and L.M. Patnaik, Genetic algorithms: a survey. Computer, 1994. 27(6): p. 17-26.

[22] Davis, L., Handbook of genetic algorithms. 1991.

[23] Glover, F. and M. Laguna, Tabu Search, in Handbook of Combinatorial Optimization: Volume1–3, D.-Z. Du and P.M. Pardalos, Editors. 1999, Springer US: Boston, MA. p. 2093-2229.

[24] Glover, F., Tabu Search—Part I. ORSA Journal on Computing, 1989. 1(3): p. 190-206.

[25] Glover, F., Tabu Search—Part II. ORSA Journal on Computing, 1990. 2(1): p. 4-32.

[26] Shapiro, J., Genetic Algorithms in Machine Learning, in Machine Learning and Its Applications: Advanced Lectures, G. Paliouras, V. Karkaletsis, and C.D. Spyropoulos, Editors. 2001, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 146-168.

[27] Barolli, A., et al. Application of Genetic Algorithms for QoS Routing in Mobile Ad Hoc Networks: A Survey. in 2010 International Conference on Broadband, Wireless Computing, Communication and Applications. 2010.

[28] Sivanandam, S.N. and S.N. Deepa, Genetic Algorithm Optimization Problems, in Introduction to Genetic Algorithms, S.N. Sivanandam and S.N. Deepa, Editors. 2008, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 165-209.

[29] Brusco, M.J. and P. Doreian, Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search. Social Networks, 2019. 56: p. 70-80.

[30] Fauziah, N.F. and Y.H. Putra, Scheduling Regular Classrooms using Heuristic Genetic and Tabu Search Algorithms. IOP Conference Series: Materials Science and Engineering, 2018. 407(1): p. 012116.

[31] Shafahi, A., Z. Wang, and A. Haghani, SpeedRoute: Fast, efficient solutions for school bus routing problems. Transportation Research Part B: Methodological, 2018. 117: p. 473-493.

[32] Wagner, M., et al., A case study of algorithm selection for the traveling thief problem. Journal of Heuristics, 2017.

[33] Applegate, D., W. Cook, and A. Rohe, Chained Lin-Kernighan for Large Traveling Salesman Problems. INFORMS Journal on Computing, 2003. 15(1): p. 82-92.

[34] Simon, D., Evolutionary optimization algorithms. 2013: John Wiley & Sons.

[35] Holland, J. and D. Goldberg, Genetic algorithms in search, optimization and machine learning. Massachusetts: Addison-Wesley, 1989.

[36] Abdul–Razaq, T.S., Solving Composite Multi objective Single Machine Scheduling Problem Using Branch and Bound and Local Search Algorithms. Al-Mustansiriyah Journal of Science, 2017. 28(3): p. 200-208.

[37] Tamssaouet, K., S. Dauzère-Pérès, and C. Yugma, Metaheuristics for the Job-Shop Scheduling Problem with Machine Availability Constraints. Computers & Industrial Engineering, 2018.

[38] Cherukupalli, K., P.R. Chinda, and S. Peddakotla, Security Constrained Optimal Power Flow by Hybrid SATS Algorithm. Journal of Advanced Research in Dynamical and Control Systems, 2018(09-Special Issue).

[39] Wang, J. and Y. Wu, Optimal Design of Urban and Rural Public Transportation Network Based on Spatio - Temporal Constraints. Journal of Applied Science and Engineering, 2018. 21(1): p. 51-58.

[40] Lu, Y., et al., A Tabu search based clustering algorithm and its parallel implementation on Spark. Applied Soft Computing, 2018. 63: p. 97-109.

[41] Reinelt, G., TSPLIB—A traveling salesman problem library. ORSA journal on computing, 1991. 3(4): p. 376-384.