A Genetic Algorithm for the Thief Orienteering Problem

Leonardo M. Faêda
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, MG, Brazil
leonardo.faeda@gmail.com

André G. Santos
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, MG, Brazil
andre@dpi.ufv.br

***Abstract*—This paper approaches the Thief Orienteering Problem, a multi-component problem that combines two combinatorial problems: Orienteering Problem (OP) and Knapsack Problem (KP). In this problem, a person (called thief) has a capacitated knapsack and has a time limit to collect objects distributed in a set of points. The departure and arrival points are fixed. The thief begins his journey with an empty knapsack and travels with speed inversely proportional to the weight of the knapsack. As long as he has time, the thief can go through the points collecting the objects. The objective of the problem is to define which route and which objects the thief must collect to maximize the profit of the knapsack. We developed a heuristic algorithm based on the genetic algorithm (GA) metaheuristic and computational experiments were carried out in order to compare the performance of the developed algorithm with the existing algorithms in the literature. Our results showed that our GA was superior in the majority of the cases.**

***Index Terms*—Orienteering problem, Knapsack problem, Multi-component problems, Thief Orienteering Problem, Genetic Algorithm.**

I. Introduction

A multicomponent problem is characterized by the combination of two or more subproblems that are interdependent, in the sense that the solution for a subproblem influences the quality of the solutions of the other subproblems [1]. This class of problems was proposed due to the need to approach the complexity of real world problems, as scientists like [1] and [2] argue that there is a big distance between the classic problems that are largely studied and the real world problems. The reason is that many studies improve techniques for classic NP-Hard problems to known benchmarks, but these problems lack the main features of reality, such as combination and interdependence, which make problems even more complex [2].

This work approaches the Thief Orienteering Problem (ThOP), a multicomponent problem proposed by Santos and Chagas [3], which combines two combinatorial problems: the Orienteering Problem and the Knapsack Problem. In this problem, the thief carries a capacitated knapsack and has a time limit to travel through a set of cities, collecting the items (KP component), knowing for each item its weight, profit and location. The thief has a fixed place of departure and arrival, and as long as he has time and capacity in his knapsack, he goes around the cities (called checkpoints) collecting items (OP component). However, as items are collected, the knapsack becomes heavier and the thief walks more slowly. Formally, the minimum and maximum speeds (v_{min} e v_{max}) are the thief speeds when the knapsack is full (at maximum capacity W) or empty, respectively. The thief speed (v) for a knapsack for a given weight (w), with $0 \leq w \leq W$, is represented by $v = v_{max} - (v_{max} - v_{min})w/W$. The objective of the problem is to determine a route and the items to be collected, in order to maximize the profit of the knapsack [3].

An example for this problem is depicted in Figure 1, where there are four points: the first (start point) and the last (end point) have no items; the other points have a set of items, each item i with its profit (p_i) and weight (w_i). For example, the item with identifier 31 is at point 3, has a profit of 110 and weight 3. The distances between each pair of points are presented on the edges. In addition to the information contained in the figure, the thief has a minimum speed 0.1, maximum speed 1, a knapsack with maximum capacity 3 and a time limit of 75 to arrive at the end point.

A solution to the ThOP can be represented only by the items collected in an orderly manner, as the route is derived from the location of the items. The collection of items 32 and 23 has a total profit of 45 + 50 = 95. This solution is feasible, as the capacity and the timeout constraints have been respected, as detailed below:

- the thief travels from the start point (1) to the point (3) at maximum speed, as the knapsack is empty: time is the distance divided by the speed, 5 / 1.0 = 5;
- at point (3) item 32 is collected, with weight 2: the speed decreases to 0.4;
- he then travels from point (3) to point (2): travel time is 8/0.4 = 20;
- at point (2) item 23 is collected, with weight 1: the speed drops to 0.1;
- finally, he travels from point (2) to the end point (4): travel time is 5/0.1 = 50. The total time is 5+20+50=75.

If we reverse the order in which the items are collected (23 and then 32), the solution is not feasible, as the time required to collect the items in this sequence is 77.43, which is over the time limit. For this example, the ideal solution is to collect item 21, thus having a profit of 120, spending time 56 and

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

respecting the capacity 3 of the backpack.

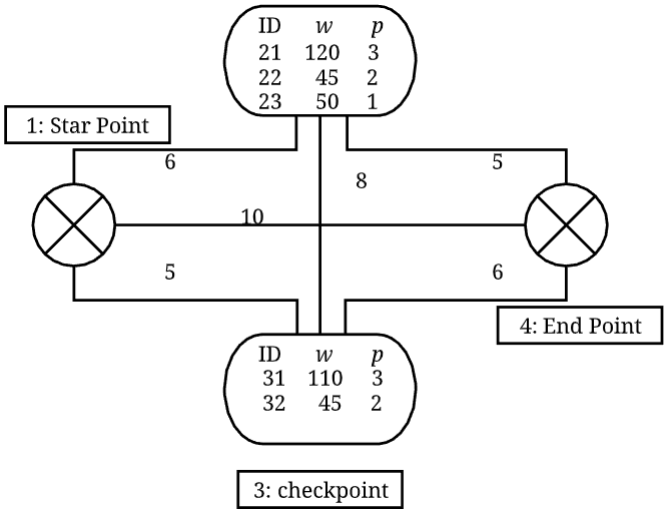

Fig. 1. ThOP example.

An applicability to this problem arises in logistics, assuming that a company has vehicles with limited capacity and the working time of its employees cannot exceed a time limit per day. The company uses vehicles to collect recyclable goods, to carry out the material to reuse process, and each item has a weight and a value. The vehicle's speed decreases as objects are collected, so the travel time between two collection locations increases. Therefore, the objective is to find a route and the items that must be collected, in order to maximize the profit of using the vehicle.

This problem was idealized by Santos and Chagas in [3], where the problem was formally defined, a mathematical model was formulated and two heuristics were proposed. The model could not be used due to its complexity: the number of variables is exponential in the number of items at a given point, due to the number of possible subsets; and the constraint that defines the speed is non-linear, as the distance must be divided by a continuous variable. The heuristics were developed based on the Iterative Local Search (ILS) metaheuristic and the Biased Random Key Genetic Algorithm (BRKGA). The results of the computational experiments showed a superiority of BRKGA when compared to ILS, for larger instances.

In this work we present a Genetic Algorithm and analyze its performance to generate upper bounds in the optimal solution of the Thief Orienteering Problem. Through techniques of experimental statistics we show that the developed algorithm produces results with significant improvements in comparison to the already existing algorithms.

This work is organized as follows: in Section II, we present a short bibliographic review of the problem; in Section III, we describe in detail the proposed heuristic; then, in Section IV, we report the computational experiments and analyze the performance of the proposed heuristic algorithm; finally, in Section V, we present our conclusions and suggest further investigations.

II. Literature Review

The evolution of multi-component problems with interdependence started in 2013 with the Traveling Thief Problem (TTP) [1]. This problem consists of a combination of two well-known classic problems: the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP). In TTP, the thief must visit all cities from a set of cities and during the visit he may collect items present in those cities to fill his knapsack. The speed at which the thief moves from one city to another is inversely proportional to the weight of his knapsack, so the heavier the thief's knapsack, the slower he goes. The knapsack is rented and the price to pay is proportional to the time of use. The thief objective is at the same time maximize the profit from the collected items, respecting the capacity of the knapsack, and minimize the total time of the route, as the total revenue is the profit collected minus the amount payed for the knapsack rental.

Since the presentation of the TTP, a benchmark of 9,720 instances has been made available [4]. Several heuristics have been proposed: Hill Climbing and Simulated Annealing [5], Ant Colony [6], Variable Neighborhood Descent [7], Profit Guided Coordination [8] and Genetic Algorithm [9]. Moreover, an exact algorithm was proposed in [10] to verify the efficiency of the existing heuristic algorithms. The TTP was also used in competitions in the main events of Evolutionary Computing [11] e [12]. A research was developed in [13] that selects the most suitable algorithm, among the twenty one existing, according to the characteristics of the instance.

The ThOP is the most recent problem approaching multi-components with interdependence [3]. It was inspired by TTP, but instead of the Traveling Salesman Problem it is based on the Orienteering Problem. The Orienteering Problem is based on an orientation sports game. In this game, players start at a specific checkpoint (origin) and try to visit as many checkpoints as possible and return to a checkpoint within a time limit. Each control point has a score and the objective is to maximize the total score collected. At ThOP, score is not given by solely visiting the checkpoints, the score is given for collected items, which must be carried by the participants to the finish line. However, since they must carry the items, each one has a knapsack to store the items collected, and the speed on the route, and consequently the time, varies according to the weight of the knapsack, just as in the TTP.

For ThOP, to the best of our knowledge, only two heuristics have already been proposed to find good solutions in reasonable time, a BRKGA and an ILS [3]. The performance of the algorithm based on the BRKGA was superior to that of ILS. The use of other metaheuristics for ThOP may improve the upper bounds on the optimal solution. A Genetic Algorithm was used in TTP and had good performance, finding for some instances the best known solutions. Due to these good results, we developed a heuristic for ThOP based on the Genetic Algorithm.

III. Heuristic Approach

In this section we describe the Genetic Algorithm developed for the problem (thopGA). An overview of the algorithm is presented in the flowchart of Figure 2 . In the following, we detail each procedure.

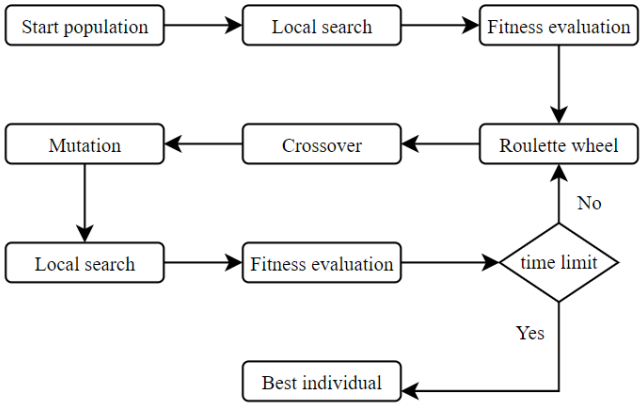

Fig. 2. Flowchart of the thopGA algorithm.

A. Solution representation and evaluation procedure

The solution is represented by two sequences: the first (R) contains the points to be visited in the order in which the journey will be carried out (as the initial and final location is fixed, it is not necessary to store this information); the second (C) contains the items to be collected (in any order, since the first sequence already informs the collection order). The example of solution given for Figure 1 would be represented by R = {3, 2} and C = {32, 23} (or C = {23, 32}).

The evaluation procedure starts with the route R and the collection C filled and the backpack K empty. For each location in R, following the order defined by the permutation, we check the items in C that are related to the R location, so all items in a given location is collected in a single visit. For each item, its weight is accumulated and we check if the capacity of the backpack has been respected and if the remaining time is sufficient for the thief to arrive at the final location. If any of these constraints is violated, this solution is considered invalid, otherwise the profit obtained is returned.

B. Local Search

Algorithm 1 presents the local search technique used as a subroutine in the main algorithm here proposed. It explores the neighborhood for a current solution to search for a better quality neighbor. If a better solution is detected, that solution becomes the current solution and the process is repeated until a stopping criterion is reached it_max, thus returning an improved solution.

This local search process has a simple structure that consists of finding a better route to collect the items in the solution, thus reducing the time spent to collect them. Subsequently, new items are inserted in the route, thus increasing the profit of the backpack. These two procedures respect the maximum capacity of the backpack and the route time limit, and are executed until there is no improvement for several iterations.

The process to find a better route is the 2-Opt neighborhood (Algorithm 2). On each iteration, two cities in the route are randomly selected and the visit order of the cities between them are reversed. For example, consider the route {B, C, D, E, F, G, H, I, J} with start point A and end point K. If the selected cities are D and G, the new route is built in three steps: the cities up to the first selected remain the same {**B, C, D**}; the route from there until the second city selected is reversed {B, C, D, **G, F, E**}; finally, the remaining cities remain in the same order {B, C, D, G, F, E, **H, I, J**}.

The process for inserting new items is performed to include items not yet collected, in order to improve the profit. The set of items is randomly shuffled and items are inserted one-by-one in the solution, if the item does not belong to the solution. For each item included, the new solution is checked if the time and capacity constraints are still satisfied. Otherwise, the item is removed. Algorithm 3 show the steps of this neighborhood.

Algorithm 1: Local search procedure

```
procedure LOCAL-SEARCH ( s, it_max )
k ← 0
while k < it_max do
    s' ← 2-OPT (s, it_max)
    s' ← ITEM-INSERTION (s')
    if f(s') > f(s) then
        s ← s'
        k ← 0
    else
        k ← k + 1
    end
end
return s
```

Algorithm 2: Neighborhood 2-OPT procedure

```
procedure 2-OPT ( s, it_max )
k ← 0
while k < it_max do
    index1 ← RANDOM (1, s.number of points)
    index2 ← RANDOM (1, s.number of points)
    if index1 > index2 then
        SWAP (index1, index2)
    end
    s' ← s.route [1 .. index1] +
         s.route [index2 .. index1 + 1] +
         s.route [index2 + 1 .. s.numberofpoints]
    if f(s').feasible then
        s ← s'
        k ← 0
    else
        k ← k + 1
    end
end
return s
```

Algorithm 3: Neighborhood Item-Insertion procedure

```
procedure ITEM-INSERTION ( s )
i ← number of items
t ← SHUFFLE (items)
k ← 0
while k < i do
    if IN-SOLUTION (t[k].idItem, s) == False then
        INSERT-ITEM (s, t[k].idItem)
        belongs ← IN-SOLUTION (t[k].idPoint)
        if belongs == False then
            INSERT-ROUTE (s, t[k].idPoint)
        end
    end
    if f(s).feasible == False then
        REMOVE-ITEM (s, t[k].idItem)
        if belongs == False then
            REMOVE-ROUTE (s, t[k].idPoint)
        end
    end
    k ← k + 1
end
return s
```

C. Genetic Algorithm

The Genetic Algorithm (GA) is an evolutionary metaheuristic [14] inspired by the Darwinian principle of the evolution of species [15]. Their methods use the evolution of a population of individuals, where each individual has a solution to the problem studied. Over the course of evolution, the characteristics of individuals with greater aptitude tend to survive, thus leading the algorithm to more promising solutions.

In the thopGA proposed here, each individual is represented by two vectors, one containing the route and the other the items that should be collected. The population has p_k individuals, that are generated initially at random, respecting the constraints of the problem: the backpack capacity and time limit. While items fit and the thief has time to arrive at the end point. The vector containing the route is filled in the order that the items are in the collection vector. After the generation of the initial population, all individuals in the population undergo an improvement through the local search explained above.

The thopGA proceeds to the next stage, creating new individuals by recombination and mutation. The process continues until a maximum number of generations is reached, controlled by a parameter $Tmax$. To generate a new child, two parents from the initial population are drawn using the roulette technique, where the probability of choosing a parent is proportional to the quality of his solution. Suppose the parent 1 and 2 collects i and j items respectively. Then, the crossover processes are carried out: the first $i/2$ items from parent 1 and the last $j/2$ items from parent 2, without repeating items, are inserted in the child and the route is generated from the selected items respecting this sequence, without repeating cities. After, the mutation process is carried out, where an item is selected randomly in the child and exchanged for another random item that is not present in the child. The crossover and mutation processes are carried out with a given probability.

The child generated from the crossover and processes undergoes an improvement through local search. The thopGA has then a component like Memetic Algorithms. The solution value of the child is compared to the value of the worst individual in the population. If the child's value is higher, the worst individual is removed from the population and the child is inserted. At the end of the entire process, the one with the most valuable solution is chosen from the population of individuals. Algorithm 4 summarizes the previously mentioned steps of thopGA.

Algorithm 4: Genetic Algorithm procedure

```
procedure thopGA ( T_max, p_c, p_m, it_max )
Population ← START-POPULATION ()
for s in Population do
    LOCAL-SEARCH (s, it_max)
end
while time limit T_max do
    i ← RANDOM (0, 1)
    if i < p_c then
        parent1 ← ROULETTE-WHEEL ()
        parent2 ← ROULETTE-WHEEL ()
        child ← CROSSOVER (parent1, parent2)
        i ← RANDOM (0, 1)
        if i < p_m then
            child ← MUTATION (child)
        end
        LOCAL-SEARCH (child, it_max)
        if f(child) > f(WORST-INDIVIDUAL
         (Population)) then
            Population.REMOVE
             (WORST-INDIVIDUAL (Population))
            Population.INSERT (child)
        end
    end
end
return BEST-INDIVIDUAL (Population)
```

IV. Computational Experiments

The proposed heuristic algorithm was implemented in the C++ language and compiled in Cygwin 3.1.2. The experiments were performed on a notebook with an Intel Core i7 processor with 2.50 GHz clock, 8 GB of memory and Windows 10 64-bit operating system.

The heuristic algorithm was evaluated using different instances, from the ThOP benchmark proposed by [3]. The characteristics of each set vary according to the number and location of cities, the number of items in each city, the weight-profit relationship type of the items, the capacity of the backpack and the maximum travel time. The following characteristics are considered in this benchmark:

- TSP based instance groups: `eil51`, `pr107`, `a280` and `dsj1000`;

- number of items per city: `01`, `03`, `05`, and `10`;
- weight-profit relation type: bounded-strongly-correlated (`bsc`), uncorrelated (`unc`), uncorrelated-similar-weights (`usw`);
- knapsack capacity class: `01`, `05`, `10`;
- maximum travel time class: `01`, `02`, `03`.

Regarding the parameters of the algorithm, we defined as the stopping criterion the execution time equal to $m/10$ seconds, the same used by [3], given in terms of the number of items m of each instance. In addition to the stop parameter, it has four more parameters: the local search stop condition it_{max}, the number of individuals in the population p_k, which is the number of individuals that are stored as the best found, the probability of crossover p_c and the probability of mutation p_m. All parameters were calibrated empirically and in the final experiment they were performed with it_max = 50, p_k = 100, p_c = 0.90, p_m = 0.85.

As the algorithm has random components, it was executed 10 times for each instance with different random seeds. The average value of the objective function and the best found in these 10 runs is used to compare the algorithm with those from the literature. Therefore we have 10 results for 432 instances, a total of 4320 results, from which we extract two values per instance, the best and the average. The complete results can be found in a supplementary material in the links https://bit.ly/3ayvWoQ and in http://www.dpi.ufv.br/~andre/thop. In this paper, we present and analyze the results grouped by instance and types, in Tables I, II, III and IV and the graphs on Figures 3, 4, 5 and 6.

The metric χ used in the work where ThOP was proposed [3] to establish a measure of the quality of convergence of the algorithm was adopted in this paper to verify the quality of the proposed algorithm. The equations (1) and (2) show how the metric χ is calculated for the thopGA proposed in this work and for the BRKGA proposed by Santos and Chagas [3]. The $thopGA^{avg}$ and $BRKGA^{avg}$ values refer to the average of the results found by the thopGA and BRKGA algorithms, respectively, for the corresponding group, and $thopGA^{best}$ and $BRKGA^{best}$ are the best solutions for each algorithm. The metric then measures how close the algorithm approaches, on average, to the best known solution. Note that the higher the χ metric, the greater the convergence of the algorithm to the best known solution.

$$\chi^{thopGA} = \frac{thopGA^{avg}}{max(thopGA^{best}, BRKGA^{best})}.100\% \quad (1)$$

$$\chi^{BRKGA} = \frac{BRKGA^{avg}}{max(BRKGA^{best}, thopGA^{best})}.100\% \quad (2)$$

TABLE I
QUALITY OF CONVERGENCE GROUPED BY NUMBER OF ITEMS PER CITY.

Group	χ^{BRKGA}				χ^{thopGA}			
	01	03	05	10	01	03	05	10
ei151	**93**	83	74	64	91	**95**	**93**	**95**
pr107	**96**	71	63	56	77	**91**	**92**	**93**
a208	**90**	54	47	35	85	**96**	**97**	**98**
dsj1000	83	44	56	74	**87**	**99**	**98**	**97**

TABLE II
QUALITY OF CONVERGENCE GROUPED BY ITEM RELATION TYPE.

Group	χ^{BRKGA}			χ^{thopGA}		
	bsc	unc	usw	bsc	unc	usw
ei151	80	79	78	**97**	**90**	**94**
pr107	73	70	71	**93**	**85**	**89**
a208	55	55	57	**96**	**92**	**94**
dsj1000	69	62	62	**95**	**95**	**96**

TABLE III
QUALITY OF CONVERGENCE GROUPED BY MAXIMUM TRAVEL TIME CLASS.

Group	χ^{BRKGA}			χ^{thopGA}		
	01	02	03	01	02	03
ei151	77	79	81	**94**	**93**	**93**
pr107	73	71	71	**88**	**89**	**89**
a208	55	56	58	**93**	**94**	**95**
dsj1000	61	65	67	**95**	**95**	**96**

TABLE IV
QUALITY OF CONVERGENCE GROUPED BY KNAPSACK CAPACITY CLASS.

Group	χ^{BRKGA}			χ^{thopGA}		
	01	05	10	01	05	10
ei151	**94**	78	65	91	**98**	**94**
pr107	**96**	65	54	90	**92**	**86**
a208	83	46	41	**92**	**96**	**96**
dsj1000	74	62	57	**89**	**99**	**99**

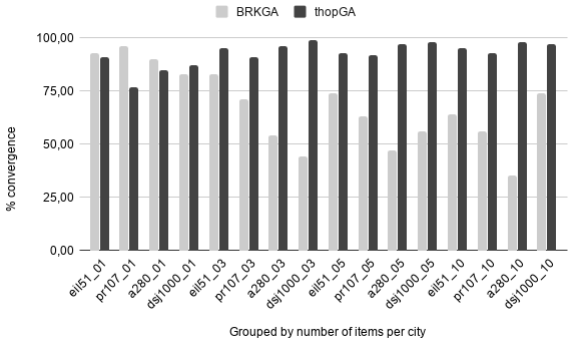

Fig. 3. Quality of convergence per number of items per city.

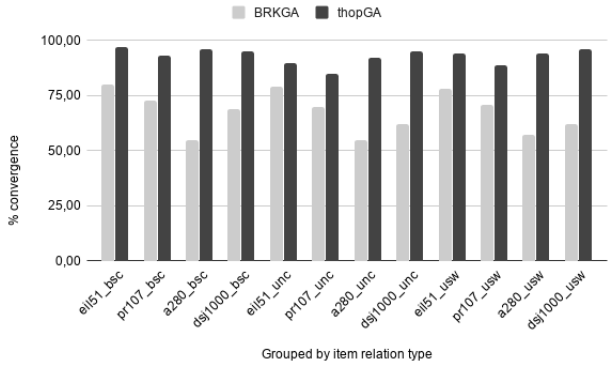

Fig. 4. Quality of convergence per weight-profit relation type.

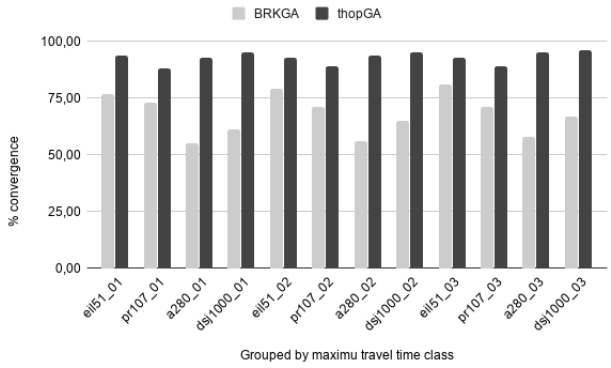

Fig. 5. Quality of convergence per maximum travel time class.

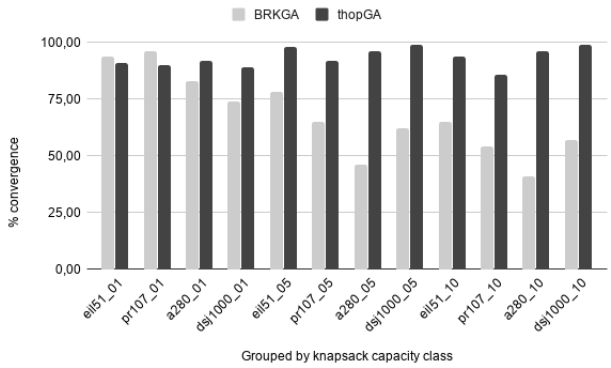

Fig. 6. Quality of convergence per knapsack capacity class.

One can see that the thopGA algorithm remains stable for most groups (convergence around 85%), while the convergence of the BRKGA algorithm decreases as the number of checkpoint increases and the number of items per checkpoint increases for the same number of checkpoint, from 93% to approximately 44%. The BRKGA has a better convergence only for the group with only one item per checkpoint (except for class `dsj1000`) and for the group with the lowest capacity of the backpack (except in classes `a208` and `dsj1000`). We believe that the machines used influenced the worse performance of thopGA, because the fewer items available, the tight is the execution time limit, hence the solution for these instances are more sensitive to the performance of the machine used. The processor of the machine used to carry out the experiments of the BRKGA algorithm is superior to that used for the experiments of the thopGA algorithm.

The behavior of the algorithms changes for different weight-profit relationship, as can be seen in Table II. Both algorithms perform better for the `bsc` type, with the thopGA algorithm being superior in all types compared to the BRKGA algorithm. The worst performance is for the `unc` type. This indicates that the algorithms still depend on a good relationship between the weight and the value of the items. Regarding the maximum travel time, the algorithms tend to have a better convergence as this parameter increases, as reported in Table III, except in the BRKGA algorithm in the cases of `pr107`, where the behavior is the opposite. And for the convergence of the algorithms regarding the backpack capacity class, the BRKGA algorithm has a better performance for group `01` (in classes `eil51` and `pr107`) and worse for the others, and the thopGA algorithm was better in groups `01` (in classes classes `a208` and `dsj1000`), `05` and `10`. These results indicate that the algorithms behave better when time and capacity limits are not highly constrained. Instances with tighter limits are more difficult.

Besides the convergence, we also analyze how often a method find an overall best solution, compared to the others. The results are presented in Tables V and VI, grouped by the type of instance. The values for each method and group of instances inform in how many of the 108 instances of the group the best (or average) solution found by the respective algorithm has the best value among the methods. The column "equals" informs in how many there is a tie, i.e., more than one method found the best value. The thopGA was superior to the other methods in all groups, in both cases, the best solution and the average solution in 10 runs. Regarding the best solution, thopGA found the best one in 304 out of 432 instances, which corresponds to more then 70%.

TABLE V
NUMBER OF BEST RESULTS FOUND FOR EACH GROUP

Instances	Best Solutions			
	ILS	BRKGA	thopGA	equals
eil51	12	26	64	6
pr107	17	25	64	2
a208	20	4	84	0
dsj1000	1	15	92	0

TABLE VI
NUMBER OF BEST AVERAGE RESULTS FOR EACH GROUP

Instances	Average Solutions			
	ILS	BRKGA	thopGA	equals
eil51	11	30	64	3
pr107	14	29	65	0
a208	19	2	87	0
dsj1000	0	13	95	0

In order to verify whether the results found by the algorithms already developed for the ThOP problem, including the one proposed in this work, have a significant difference among them, the Friedman test was applied. The Friedman test was chosen because we cannot say that the samples used in the tests are independent [16]. The hypothesis observed in this test is that the algorithms are the same; if this statement is denied, we can say that there is at least one different algorithm. Using the p-value = 0.05, the result found was less than 0.05, thus concluding that there is at least one different algorithm with 95% confidence.

As a way to observe the results, Figure 7 has a boxplot graph that illustrates the results of the experiments. On the x-axis we have the BRKGA (best results currently in the literature) and the thopGA algorithm proposed in this work, and on the y-axis the percentage of solution convergence for each instance. The normalization used to verify the percentage of improvement in the solution of each instance was the χ metric presented in equations (1) and (2). The thopGA algorithm has a higher average result than BRKGA and we can also observe that the thopGA algorithm has greater convergence than BRKGA.

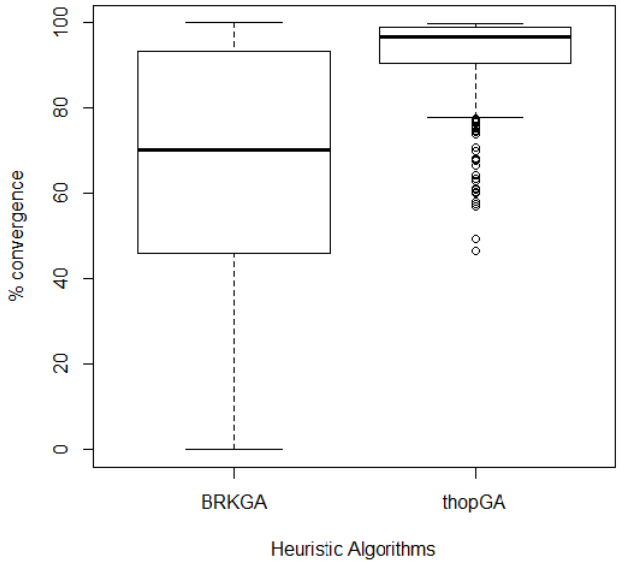

Fig. 7. Comparative analysis of the convergence of BRKGA algorithm and thopGA for all instances.

The algorithm that is apparently the best one, observed in the graph previously, must pass the Wilcoxon test [17]. In this test, it is verified if the visually better algorithm has a significant difference in relation to the other algorithms. Initially, the p-value is adjusted using the Bonferroni correction [18], and the hypothesis observed for the test is that the pairwise compared algorithms are the same. The value found for (thopGA with BRKGA) is less than 0.05, then we can say that the thopGA algorithm is significantly different from BRKGA and has a better performance.

V. Conclusions and Further Investigations

In this paper, we approach the Thief Orienteering Problem (ThOP), a combinatorial problem with interdependence. We proposed the thopGA, a Genetic Algorithm based heuristic. We tested thopGA for the 432 instances available in the literature and compared it with the existing heuristic algorithms (ILS and BRKGA). The results showed a superiority of thopGA when compared to the other algorithms, mainly in large instances. We believe that this superiority is due to the combination of individuals who have good characteristics, and also by the local search used in the generation of the initial population and after the new individuals generated, generating others with better results. As a future work, we would like to develop an exact mathematical programming model to test the quality of the developed heuristics algorithms, at least for the smaller instances, and to analyze the performance of new heuristics based on other metaheuristics.

Acknowledgement

The authors thank Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for the financial support of this project.

Appendix
Supplementary Material

Supplementary material associated with this paper can be found, in the online version, at the following urls: https://bit.ly/3ayvWoQ and in http://www.dpi.ufv.br/~andre/thop

References

[1] M. R. Bonyadi, Z. Michalewicz, L. Barone. The travelling thief problem: the first step in the transition from theoretical problems to realistic problems. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, pp. 1037–1044, 2013.

[2] Z. Michalewicz. Quo vadis, evolutionary computation?. In IEEE World Congress on Computational Intelligence, pp. 98-121, 2012.

[3] A. G. Santos and J. B. Chagas. The Thief Orienteering Problem: Formulation and Heuristic Approaches. In 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-9, 2018.

[4] S. POLYAKOVSKIY, et al. A comprehensive benchmark set and heuristics for the traveling thief problem. In: ACM.Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, p. 477–484, 2014.

[5] M. E. YAFRANI and B AHIOD. Efficiently solving the traveling thief problem using hill climbing and simulated annealing. In: Information Sciences, p. 231–244, 2018.

[6] M. WAGNER. Stealing items more efficiently with ants: a swarm intelligence approach to the travelling thief problem. In: SPRINGER.International Conference on Swarm Intelligence, p. 273–281, 2016.

[7] R. P. ARAUJO, et al. A novel list-constrained randomized vnd approach in gpu for the traveling thief problem. In: Electronic Notes in Discrete Mathematics, Elsevier, p. 183–190, 2018.

[8] M. NAMAZI, et al. A profit guided coordination heuristic for travelling thief problems. In: Twelfth Annual Symposium on Combinatorial Search, 2019.

[9] D. K. Vieira, G. L. Soares, J. A. Vasconcelos and M. H Mendes. A genetic algorithm for multi-component optimization problems: the case of the travelling thief problem. In European Conference on Evolutionary Computation in Combinatorial Optimization pp. 18-29, 2017.

[10] J. WU, et al. Exact approaches for the travelling thief problem. In: SPRINGER.Asia-Pacific Conference on Simulated Evolution and Learning, p.110–121, 2017.
[11] “Optimisation of Problems with Multiple Interdependent Components”. Internet: https://cs.adelaide.edu.au/ optlog/CEC2014Comp Jul. 9, 2014 [Jan. 15, 2018].
[12] “Optimisation of Problems with Multiple Interdependent Components”. Internet: http://gecco-2017.sigevo.org/index.html/Competitions [Jan. 15, 2018].
[13] M. WAGNER, et al. A case study of algorithm selection for the traveling thief problem. In: Journal of Heuristics, Springer, p. 295–320, 2018.
[14] J. H. Holland. Genetic algorithms. In: Scientific american, p. 66-73, (1992).
[15] C. Darwin, and G. De Beer. “The Origin of Species by Means of Natural Selection, Or, The Preservation of Favored Races in the Struggle for Life”, In: Oxford University Press, 1956.
[16] D. W. ZIMMERMAN and B. D. ZUMBO. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. In The Journal of Experimental Education, pp. 75-86, 1993.
[17] E. A. GEHAN. A generalized Wilcoxon test for comparing arbitrarily singly-censored samples. Biometrika, pp. 203-224, 1965.
[18] T. V. PERNEGER. What’s wrong with Bonferroni adjustments. Bmj, pp. 1236-1238, 1998.