

A HYBRID APPROACH TO SOLVE THE TEAM ORIENTEERING PROBLEM

Eduardo A. A. G. Macedo

Instituto Tecnológico de Aeronáutica (ITA)
Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brasil
macedo.eduardo@unifesp.br

Edson Luiz França Senne

Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brasil
edson.senne@unifesp.br

ABSTRACT

This paper addresses the resolution of the Team Orienteering Problem (TOP) through the integration of Iterated Local Search (ILS) and Fix-and-Optimize (F&O) metaheuristics. The TOP is a challenging combinatorial optimization problem classified as an NP-hard, whose objective is to maximize the total prize collected by a fleet of vehicles by determining an optimal sequence and selection of locations to visit for each vehicle, while adhering to the maximum route duration constraint. To evaluate the effectiveness of the proposed method, computational experiments were carried out using 82 instances from the existing literature, with the number of nodes ranging from 100 to 401. The algorithm was run 20 times for every instance and achieved identical results to the best known solutions in 21 instances, with an average relative deviation of 1.02% from the best known solutions, considering the best result reached for each instance. The results have shown that the proposed hybrid approach has a performance comparable to other algorithms documented in the literature.

KEYWORDS. Iterated Local Search. Fix-and-Optimize. Mixed Integer Programming.

Paper topics (Team Orienteering Problem. Iterated Local Search. Fix-and-Optimize. Mixed Integer Programming. Hybrid algorithm. Metaheuristics.)

1. Introduction

The Team Orienteering Problem (TOP) is a variant of the Profit Vehicle Routing Problems whose objective is to maximize the total collected prize while adhering to a maximum distance or route duration constraint. It has received considerable attention due to its practical relevance in various scenarios, including tourist trip planning [Vansteenwegen et al., 2009b], field technician assistance [Tang and Miller-Hooks, 2005], emergency response [Halat and Ozkan, 2021], military reconnaissance [Murray and Karwan, 2010], among others.

The study of routing problems with the goal of maximizing overall benefits initially gained attention through the work of Tsiligrirides [1984] in addressing the Orienteering Problem. However, it was after the contributions of Butt and Cavalier [1994] and Chao et al. [1996] that this area of research gained significant importance.

In Butt and Cavalier [1994] work, the focus was on the Multiple Tour Maximum Collection Problem (MTMCP). The objective was to determine K routes that visit a subset of weighted nodes in a graph, aiming to maximize the total prize collected from that subset of nodes. Chao et al. [1996] introduced the term Team Orienteering Problem (TOP) and proposed a large set of instances.

The existing literature reveals numerous proposed methods for solving the TOP, each exhibiting varying levels of performance. Table 1 summarizes main papers regarding this topic covering heuristic and hybrid solution approaches. The first column presents the authors, the second column shows the approaches used to solve the problem and the third column has a brief summary of performance achieved for these methods.

Golden et al. [1987] demonstrated that the Orienteering Problem (OP) is an NP-hard problem and, therefore, no polynomial-time deterministic algorithm is known to solve any OP instance optimally. Consequently, the TOP is also NP-hard as considering the use of a single vehicle, it can be reduced to the OP. Therefore, the prevailing solution approaches for this problem involve exact methods based on mathematical programming and heuristic methods.

Given the NP-hard nature of this combinatorial problem, efficient and effective solution methods remain a vital research focus, given its practical significance. In light of this, this paper contributes by proposing one hybrid algorithm that combines the metaheuristic Iterated Local Search (ILS) and the framework Fix-and-Optimize (F&O).

The remainder of this paper is organized as follows. Section 2 presents a MIP mathematical formulation. In Section 3, the solution approach is described, whereas Section 4 presents the results obtained by ILS + F&O and compare it to Best Known Solutions (BKS) in the literature. In Section 5, the main achievements and possible future works are summarized.

2. Mathematical formulation

Bianchessi et al. [2018] presents a two-index variable formulation based on a commodity flow containing a polynomial number of variables and constraints. The problem is considered to be defined on a directed graph $G = (V, A)$, where $V = \{0, \dots, n+1\}$ and $A = \{(i, j) \mid i \neq n+1; j \neq 0; i \neq j; i, j \in V\}$. The node set contains the starting point 0 and the ending point $n+1$ and the set $N = 1, \dots, n$ represents the customers. For each set $S \subset N$, let $\delta^+(S) = \{(i, j) \in A \mid i \in S, j \notin S\}$ e $\delta^-(S) = \{(i, j) \in A \mid i \notin S, j \in S\}$ be the set of arcs leaving and entering the set S , with $\delta^+(i) = \delta^+(\{i\})$ e $\delta^-(i) = \delta^-(\{i\})$.

Table 1: Heuristics and hybrid algorithms used for TOP solution

Reference	Algorithms	Performance
Tang and Miller-Hooks [2005]	Tabu Search	At least 43 new BKS
Archetti et al. [2007]	Variable Neighborhood Search	128 new BKS
Ke et al. [2008]	Ant Colony Optimization	12 new BKS
Vansteenwegen et al. [2009b]	Skewed Variable Neighborhood Search	No improvement
Vansteenwegen et al. [2009a]	Guided Local Search	No improvement
Souffriau et al. [2010]	GRASP with Path Relinking	5 new BKS
Bouly et al. [2010]	Hybrid Genetic Algorithm	5 new BKS
Lin [2013]	Multi-start Simulated Annealing	5 new BKS
Dang et al. [2013]	Particle Swarm Optimization inspired Algorithm (PSOiA)	1 new BKS
Kim et al. [2013]	Large Neighborhood Search	5 new BKS
Ferreira et al. [2014]	Genetic Algorithm	No improvement
Vidal et al. [2015]	Hybrid Genetic Search and Integrated Local Search	No improvement
Ke et al. [2016]	Pareto Mimic Algorithm (PMA)	1 new BKS
Tsakirakis et al. [2019]	Similarity Hybrid Harmony Search	No improvement
Hammami et al. [2020]	Hybrid Adaptive Large Neighborhood Search (HALNS)	1 new BKS
Misir et al. [2022]	ALORS	No improvement

The decision variables are defined as follows: binary variables x_{ij} are equal to 1 if the arc (i, j) is traversed; and 0, otherwise. Binary variables y_i are equal to 1 if node i is visited; and 0, otherwise. Continuous variables z_{ij} represent the arrival time at vertex j of a vehicle coming from vertex i . The parameters B_i , K , T_{max} and t_{ij} represent the prize collected at vertex i , the number of routes or vehicles, the maximum duration of the route and the travel time to go from vertex i to j , respectively.

$$\text{Max } \sum_{i \in N} B_i y_i \quad (1)$$

$$\sum_{j \in N} x_{0j} = \sum_{i \in N} x_{i,n+1} = K \quad (2)$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} = \sum_{(i,j) \in \delta^+(i)} x_{ij} = y_i \quad i \in N \quad (3)$$

$$z_{0j} = t_{0j} x_{0j} \quad j \in N \quad (4)$$

$$\sum_{(i,j) \in \delta^+(i)} z_{ij} - \sum_{(j,i) \in \delta^-(i)} z_{ji} = \sum_{(i,j) \in \delta^+(i)} t_{ij} x_{ij} \quad i \in N \quad (5)$$

$$z_{ij} \leq (T_{\max} - t_{j,n+1}) x_{ij} \quad (i, j) \in A \setminus \{(0, n+1)\} \quad (6)$$

$$z_{ij} \geq (t_{0i} + t_{ij}) x_{ij} \quad (i, j) \in A \setminus \{(0, n+1)\} \quad (7)$$

$$y_i \in \{0, 1\} \quad i \in N \quad (8)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in A \setminus \{(0, n+1)\} \quad (9)$$

$$t_{0,0} = t_{n+1,n+1} = 0 \quad (10)$$

The objective function maximizes the total collected benefit; constraints 2 ensure that K vehicles depart from the initial point and arrive at the final point; 3 impose that a vehicle arrives at and departs from a specific node if it is visited; constraints 4 set a bound on the flow originating from the depot (initial point), while 5 represent flow conservation

constraints. Constraints 6 limit the duration of each route, and constraints 7 impose lower bounds on the values of the variables z in order to restrict the number of feasible values these variables could take in fractional solutions. Constraints 8-9 refer to the domain of the variables and constraint 10 trivially defines the travel time for starting and ending points. This model was used in the Fix-and-Optimize framework as shown in Section 3.4.

3. Solution approach

The solution representation used considers a vector of vectors, corresponding to the paths taken by each vehicle. Each route is represented by a permutation of integers which correspond to the locations or vertices of the graph, indicating which locations and in what sequence they are visited. An illustration of this representation can be seen in Fig. 1 for a problem with 3 vehicles, only one depot and 100 nodes.

0	7	29	4	73	82	23		
0	3	27	13	59	93	99	8	5
0	18	45	65	11				

Figure 1: Solution Representation

The value of the total prize collected in each route is stored in a vector with K elements, and the objective function is given by the sum of the elements of this vector. It should be noted that during the execution of the code, the objective function is only updated, and the sum is performed only at the end.

The cost is stored, represented and updated similarly to the collected premium, allowing handling infeasibility during the execution of the algorithm. In addition to the perturbation mechanism, which aims to reach different local optima, another diversification mechanism is adopted when the algorithm cannot escape from local optima, reaching a certain number of iterations without improvement. When this occurs, a new solution is constructed and the perturbation and local search procedures are restarted. Thus, three solution construction methods were adopted, which are described below.

3.1. ILS Initial Solution and Restart

The initial step involves a pre-processing procedure that eliminates unfeasible points, which cannot be reached by departing from and returning to the warehouse [Chao et al., 1996]. Subsequently, K routes (corresponding to the number of vehicles) are initialized by including the depot and another point, randomly chosen from a shortlist of candidates known as the List of Restricted Candidates (LRC).

The LRC is constructed as follows: initially, the points are sorted based on their distance from the warehouse, and then the elements lying between two intermediate positions are extracted from this sorted list. This approach aims to strike a balance between pure greediness and pure randomness, with the intention of generating diverse solutions at each initialization and facilitating exploration of different regions within the solution space.

Next, K elements from the LRC are selected and assigned as the first points to be visited in each of the routes. Finally, the final destination is included as the last point in every route. Once the initialization is complete, one of the following three methods is employed to continue building the routes: cheapest insertion; cost-effective insertion; or cheaper insertion with bonus.

The first method inserts a given element/vertex w , considering the best insertion economy z_1 between any two points among all routes, calculated by: $z_1 = t_{i,w} + t_{w,j} - t_{i,j}$.

The second method inserts a given element/vertex w , considering the best cost benefit of insertion (z_2) between any two points among all routes, calculated by:

$$z_2 = \frac{B_w \times \sqrt{B_w}}{z_1}, \quad (11)$$

where B_w represents the benefit associated with the considered point. The third method inserts a certain element/vertex w , considering the best economy z_1 of insertion between any two points among all the routes, subtracted by the round trip distance from the considered point to the depot [Subramanian, 2008], calculated by:

$$z_3 = z_1 - \gamma(t_{0,w} + t_{w,0}), \quad \gamma \in [0, 1] \quad (12)$$

This bonus portion aims to allow points distant from the depot to be included in the initial solution, aiming at greater diversity in the construction of the solution. The insertion procedures continue until the time limit (cost) of each route is not exceeded, i.e. the unfeasible solutions are disregarded. After a certain number of iterations without improvement (R_{max}), a new solution can be constructed using one of these methods.

3.2. Perturbation

The perturbation procedure consists of randomly removing elements from the solution. The magnitude of the disturbance is randomly defined in the interval $[\beta_{min}, \beta_{max}]$, as a percentage of the total number of locations visited. To perform each movement (removal), a route is chosen and then the element to be removed is selected. At the end, the procedure 2-OPT, detailed in Section 3.3, is applied to optimize the paths of the resulting solution.

3.3. Local Search Procedures and Acceptance Criteria

The Variable Neighborhood Descent (VND) was used to aggregate the local search heuristics, using INSERT, REPLACE, SWAP and 2-OPT procedures. It is noteworthy that the first two procedures make it possible to achieve better objective function values, while the other ones, despite not increasing the total benefit collected, can reduce the cost of routes allowing new points to be inserted later.

The insertion procedure (INSERT) and the replacement of a point visited by one that was not included in the solution (REPLACE) verify, among all the possibilities of nodes not included in the solution, the one that provides the best cost benefit, according to Eq. 12, that is, the *Best Improvement* strategy was used.

The SWAP procedure performs an exchange of two locations between two different routes. The objective is to find the movement that allows the greatest reduction in execution time. Thus, the exchange can be performed if there is an economy in both routes or if the economy in one route exceeds the addition of time in the other route. The 2-OPT procedure reverses the visitation order of a route segment between two selected nodes. It also does not increase the value of the benefit, but it can optimize the route, saving its execution time, considering the selected points to be visited. In both procedures, only feasible solutions are considered. The acceptance criterion used is that of improvement in all situations, that is, priority was given to intensification after the Local Search procedures.

3.4. Hybridization with Fix-and-Optimize

Fix-and-Optimize is a refinement matheuristic that iteratively solves Mixed Integer Programming (MIP) subproblems to improve the current solution. The subproblems are obtained through the decomposition of the set of integer variables Q into P subsets

$Q_1, Q_2, \dots, Q_p$, such that $Q_1 \cup Q_2 \cup \dots \cup Q_p \subseteq Q$. It should be noted that the P subsets do not need to be disjoint. In the p -th subproblem MIP^p , the integer variables not belonging to the set Q_p are fixed, and the rest are re-optimized by being left free to take on new integer values.

In order to integrate the solution obtained by ILS with F&O, two procedures are necessary: the conversion of the solution representation of the ILS (vectors with permutation of integers) into values for the variables of the MIP; and the nodes and associated variables subsets definition, thus allowing the sequential resolution of subproblems.

The definition of the subset variables is done by adjacent nodes [Oliveira and Scarpin, 2020], considering the vertices not included in the current solution. As the problem is defined over a complete graph, each node is adjacent or neighboring the others and the following steps are performed: 1) identification of the set of nodes not belonging to the current solution (NR); 2) ordering the distances of the elements of NR in relation to the elements of the solution; and 3) subset assembling with inclusion of elements, considering the number of subproblems (N_{SP}) to be solved and the size of each subset (n_p).

The first item is carried out by checking the values of the variable y_i which takes value 1 if a given point belongs to the solution and value 0, otherwise. In the second, the distances are organized for each element in the set NR to the elements assigned in the current solution. After that, the distances are sorted using the values contained in the matrix t_{ij} . The objective is to assemble the subsets taking as reference the points closest to the routes already formed, increasing the chances of occasional improvements in the solution.

Then, having the ordered list of points, the subsets are assembled taking these points as a reference and adding their nearest neighbors regardless of whether they belong to the solution or not. The larger the subset size, the more chance of improving the incumbent solution, although it takes longer for the subproblem to be solved. Furthermore, the greater the number of unvisited locations, the more subproblems can be considered, which does not necessarily guarantee a greater number of improvements. Thus, the balance in the definition of these parameters (N_{SP}, n_p) must be considered in order to allow the improvement of the solution, avoiding that it takes a long time to finish the execution of the algorithm.

To exemplify the distribution of points in subsets, we consider the data of the instance *eil101b_m3* that contains the coordinate values and benefits associated with each point. The green triangle represents the depot and the points to be visited are represented by blue circles, whose size is proportional to the associated benefit. A graphical representation of two subsets can be seen in Fig. 2, where the yellow diamonds with black edges represent the subset of points considered in each subset and the green diamond, the reference point.

As the graph is completely connected, the neighborhood $N(i)$ of a node i is given by $V \setminus \{i\}$. Thus, the variables x_{ij} associated with the subset Q_p are those whose indices represent some combination of the element i belonging to Q_p with its respective neighborhood. At each iteration, the variables x_{ij} associated with the subset Q_p are kept integer and are released to assume new values, that is, they no longer have their values fixed. Then, MIP^p is solved and, if the solution found is equal to or better than the current best solution, the solution S is updated and the new values of the variables associated with the considered subset are fixed. The procedure continues until there is no further improvement after solving the subproblems concerning all subsets. The integration of F&O to metaheuristics is detailed in Algorithm 1.

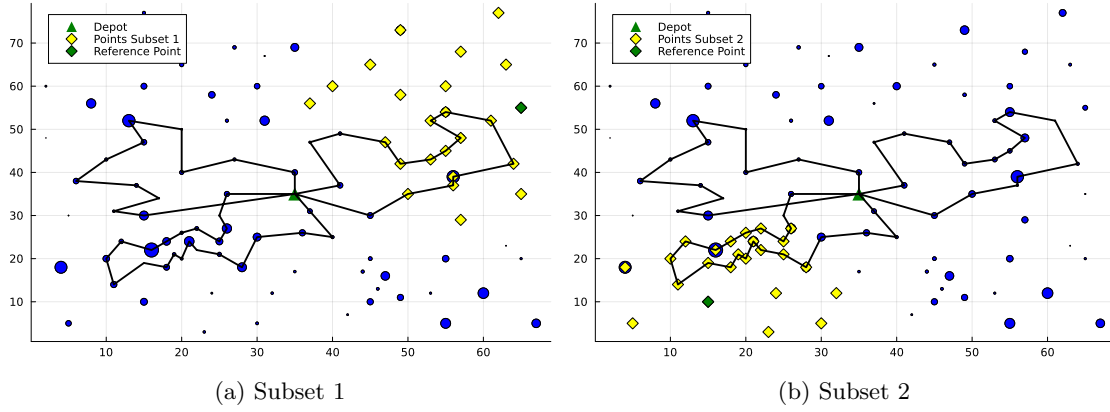


Figure 2: Subsets definition example

Algorithm 1: ILS + Fix-and-Optimize

Data: Instance
Result: Solution S

- 1 $S \leftarrow$ Solution given by ILS;
- 2 Let $S.value$ be the value of the solution S ;
- 3 Fix values of variables x_{ij} based on the obtained solution;
- 4 Execute ‘Subset definition’ procedure Q_p ;
- 5 **while** *improved* **do**
- 6 $improved \leftarrow$ false;
- 7 **for** $p = 1, \dots, N_{SP}$ **do**
- 8 Release the values of the variables of the set Q_p , keeping them integers;
- 9 $S' \leftarrow$ solution MIP^p ;
- 10 **if** $S'.value \geq S.value$ **then**
- 11 $S \leftarrow S'$;
- 12 $improved \leftarrow$ true;
- 13 **end**
- 14 **end**
- 15 **end**
- 16 Return S ;

4. Computational Results

The algorithms were programmed in Julia language, using the JuMP optimization package and *Gurobi.jl* package to interface with the Gurobi solver. The tests were performed on a computer with an Intel(R) Core(TM) i7-4770 CPU @ 3.40 GHz, 12.0 GB RAM memory.

The experiments were performed using instances available in the literature and for each one of them, the algorithm was run 20 times. The IRACE package [López-Ibáñez et al., 2016] was used to calibrate the parameters of ILS with the following possible values: $\beta_{min} \in \{5\%, 7\%, 10\%, 12\%\}$; $\beta_{max} \in \{20\%, 25\%, 30\%, 35\%\}$; and $R_{max} \in \{300, 500, 700, 900\}$. The parameter values obtained were: $\beta_{min} = 7\%$, $\beta_{max} = 35\%$ and $R_{max} = 700$. Moreover, the maximum total number of 10000 iterations was used as stopping criteria. The γ parameter of the construction method for the cheapest insertion solution with a bonus was defined

as 0.6, based on the work of Subramanian [2008] and on the observation of preliminary experiments carried out. The Restricted List of Candidates for initializing the routes, that is, for including the first node to be visited in each route, was defined by the elements between the 30th and 100th most distant elements from the depot.

Regarding the F&O parameters, the following values were used: solver solution time of 10 seconds; subset size of 25 elements (n_p); and number of subproblems (N_{SP}) of 25, 50 or $|NR|$. The size of subsets was defined in order to quickly reach good solutions and the number of subproblems varied according to the size (s_{ni}) of the set of nodes not belonging to the solution. When $s_{ni} < 25$, $N_{SP} = |NR|$; if $25 \leq s_{ni} \leq 50$, then $N_{SP} = 25$; and when $s_{ni} > 50$, $N_{SP} = 50$.

4.1. Instances

Several works in the literature used instances of Chao et al. [1996], composed of 7 sets, defined by the number of nodes ranging from 21 to 102 and totaling 387 instances. Dang et al. [2013] introduced a set of other 82 instances based on two instance classes of Fischetti et al. [1998] for the *Orienteering Problem* for which it used the same transformation used by Chao et al. [1996], dividing the original *Orienteering Problem* T_{max} by the number of vehicles (K), obtaining a new value for the maximum duration of the routes for the TOP instances.

Bearing in mind that the instances proposed by Dang et al. [2013] are larger and still little explored in the literature because only Dang et al. [2013], Ke et al. [2016] and Hammami et al. [2020] used them, it was decided to use this set of instances to run the computational experiments.

4.2. Performance analysis

The results can be seen in Table 2 along with the best results known in the literature. The percentage deviation from the BKS (RPD) was calculated as follows, where ‘ins’ denotes a certain instance:

$$RPD = \frac{BKS_{ins} - S_{ins}}{BKS_{ins}} \times 100\% \quad (13)$$

The first column of Table 2 contains the names of the instances; ‘**HALNS**’ corresponds to the values of the solutions and execution time reached by the Hybrid Adaptive Large Neighborhood Search algorithm proposed by Hammami et al. [2020] and which coincide with the Best Known Solutions in the literature. The other columns present the ‘**Best**’ values found by the ILS + F&O and the respective relative deviation in percentage (‘**RPD**’), as well as the mean values of the objective function (‘**Average**’) and the respective relative deviation (‘**ARPD**’). The ‘**Average Time**’ is reported in the last column.

The solutions using the ILS + F&O method reached identical solutions to the best known solutions in 21 out of 82 instances with an average of the best percentage deviation of 1.02%. Considering the 20 executions for each instance, the mean deviation (ARPD) was 2.34%, demonstrating a slight drop in performance in some executions.

The comparison of computational time obtained in the execution of different algorithms can be influenced by several factors such as the CPU clock frequency, system architecture, programming language used, data structures, compilers, among others. However, such a comparison may suggest some insights. Table 3 presents characteristics of the computational environment used by the authors who solved the TOP in the same set of instances, the number of best known solutions obtained by each method, as well as the average computational time, considering all the executions in the several instances.

Table 2: Results of ILS + F&O

Instance	HALNS [Hammami et al., 2020]		ILS + F&O				
	Solution	CPU (s)	Best	RPD (%)	Average	RPD (%)	Average Time (s)
eil101b.m3	916	69.44	916	0.00	915.60	0.04	20.14
eil101c.m2	1305	153.85	1305	0.00	1302.20	0.21	13.58
eil101c.m3	1251	116.15	1251	0.00	1247.00	0.32	20.85
cmt101c.m3	1300	23.10	1300	0.00	1294.00	0.46	14.60
bier127_gen1.m2	106	207.20	105	0.94	104.20	1.70	14.57
bier127_gen1.m3	103	209.55	103	0.00	102.00	0.97	15.98
bier127_gen2.m2	5464	301.33	5458	0.11	5438.00	0.47	26.57
bier127_gen2.m3	5393	227.51	5391	0.04	5319.20	1.37	33.64
bier127_gen2.m4	5123	355.99	5120	0.06	5113.80	0.18	21.76
bier127_gen3.m2	2885	184.99	2878	0.24	2802.80	2.85	21.50
bier127_gen3.m3	2706	69.36	2704	0.07	2658.40	1.76	25.75
bier127_gen3.m4	2402	222.40	2402	0.00	2365.20	1.53	16.71
pr136_gen1.m2	63	70.72	63	0.00	61.60	2.22	44.67
pr136_gen2.m2	3646	95.62	3641	0.14	3621.80	0.67	64.17
kroA150_gen2.m2	4335	392.30	4333	0.05	4327.60	0.17	102.99
kroA150_gen3.m3	2726	168.81	2726	0.00	2690.40	1.31	58.32
cmt151b.m3	1385	164.65	1382	0.22	1367.20	1.28	105.17
cmt151c.m2	1964	131.96	1962	0.10	1955.60	0.43	27.80
cmt151c.m3	1916	355.61	1911	0.26	1902.60	0.70	33.51
cmt151c.m4	1880	107.89	1880	0.00	1868.20	0.63	31.89
rat195_gen2.m2	5148	335.53	5148	0.00	5130.20	0.35	167.74
rat195_gen3.m3	2574	276.84	2555	0.74	2536.80	1.45	30.84
cmt200b.m2	2096	183.28	2074	1.05	2037.60	2.79	260.20
cmt200b.m3	2019	386.33	1971	2.38	1957.80	3.03	265.26
cmt200b.m4	1894	324.31	1894	0.00	1861.00	1.74	200.68
cmt200c.m2	2818	368.42	2808	0.35	2801.00	0.60	51.61
cmt200c.m3	2766	273.01	2756	0.36	2743.20	0.82	65.33
cmt200c.m4	2712	395.91	2700	0.44	2676.40	1.31	79.83
kroA200_gen1.m4	81	114.61	81	0.00	79.40	1.97	55.43
kroB200_gen1.m2	111	295.30	109	1.80	104.40	5.95	246.15
kroB200_gen2.m2	6185	438.32	6180	0.08	6104.40	1.30	236.85
kroB200_gen2.m4	4944	360.44	4930	0.28	4931.20	0.26	45.29
kroB200_gen3.m2	4765	470.10	4765	0.00	4685.00	1.66	248.14
kroB200_gen3.m3	3028	547.22	2984	1.45	2894.00	4.43	118.51
ts225_gen2.m2	5859	686.54	5854	0.09	5702.60	2.67	168.31
gr229_gen1.m4	223	344.28	220	1.35	220.00	1.35	15.90
gr229_gen2.m3	11566	441.80	11529	0.32	11529.60	0.32	2.77
gr229_gen2.m4	11355	377.28	11120	2.07	11120.00	2.07	17.81
gr229_gen3.m3	8056	938.75	8031	0.31	7983.40	0.90	2.58
gr229_gen3.m4	7651	828.68	7621	0.39	7539.80	1.45	16.63
gil262a.m2	4078	451.76	4078	0.00	4059.40	0.46	381.69
gil262a.m4	3175	133.79	3175	0.00	3141.20	1.06	69.96
gil262b.m2	8081	545.17	7902	2.22	7721.20	4.45	342.60
gil262b.m3	7585	463.19	7527	0.76	7367.40	2.87	338.19
gil262b.m4	6781	329.10	6751	0.44	6593.80	2.76	270.44
gil262c.m2	11030	731.25	10999	0.28	10908.60	1.10	227.14
gil262c.m3	10757	650.87	10634	1.14	10551.20	1.92	277.46
gil262c.m4	10281	516.50	10216	0.63	10070.80	2.05	266.73
gil262_gen1.m3	101	709.66	101	0.00	98.20	2.77	136.91
gil262_gen1.m4	78	76.98	77	1.28	77.00	1.28	15.40
gil262_gen2.m2	7498	387.00	7471	0.36	7259.00	3.19	383.78
gil262_gen2.m3	5615	352.39	5598	0.30	5535.20	1.42	188.74
gil262_gen3.m2	7183	532.11	6938	3.41	6707.60	6.62	336.43
gil262_gen3.m4	2507	227.15	2507	0.00	2482.00	1.00	37.62
pr264_gen1.m4	107	195.83	107	0.00	106.00	0.93	8.02
pr264_gen2.m2	6635	642.98	6628	0.11	6617.40	0.27	3.73
pr264_gen2.m3	6420	527.27	6420	0.00	6406.00	0.22	58.85
pr264_gen2.m4	5584	294.70	5567	0.30	5561.00	0.41	53.12
pr264_gen3.m3	2772	1490.82	2768	0.14	2756.80	0.55	41.28
pr299_gen1.m2	139	355.26	137	1.44	135.60	2.45	208.18
pr299_gen1.m3	111	413.78	110	0.90	108.60	2.16	61.78
pr299_gen1.m4	84	191.29	84	0.00	82.60	1.67	10.60
pr299_gen2.m3	6018	690.41	5964	0.90	5883.40	2.23	95.99
pr299_gen2.m4	4457	257.75	4456	0.02	4449.40	0.17	25.42
pr299_gen3.m2	5729	692.12	5568	2.81	5279.40	7.85	211.11
pr299_gen3.m3	3655	644.38	3625	0.82	3522.40	3.63	89.41
pr299_gen3.m4	2268	271.94	2268	0.00	2237.80	1.33	19.03
lin318_gen1.m2	180	2702.83	166	7.78	164.20	8.78	412.55
lin318_gen1.m3	149	203.51	147	1.34	146.60	1.61	190.02
lin318_gen2.m2	9544	1962.14	9157	4.05	9018.00	5.51	451.52
lin318_gen2.m3	7807	1600.58	7761	0.59	7690.20	1.50	259.99
lin318_gen3.m2	7936	581.27	7585	4.42	7475.80	5.80	437.42
lin318_gen3.m4	3797	128.84	3794	0.08	3791.00	0.16	49.12
rd400_gen2.m2	13045	4035.08	12429	4.72	12270.20	5.94	528.51
rd400_gen2.m3	12646	2814.58	12341	2.41	11878.20	6.07	529.12
rd400_gen2.m4	12032	4070.44	11728	2.53	11270.00	6.33	520.62
rd400_gen1.m2	232	3925.58	219	5.60	215.20	7.24	517.28
rd400_gen1.m3	224	4178.00	213	4.91	208.20	7.05	523.19
rd400_gen1.m4	213	3000.90	207	2.82	200.00	6.10	511.64
rd400_gen3.m2	12431	2814.66	12201	1.85	11676.60	6.07	503.33
rd400_gen3.m3	11639	3811.24	11275	3.13	10952.00	5.90	515.02
rd400_gen3.m4	10436	3615.46	10067	3.54	9855.60	5.56	518.19
Average	4436.10	783.36	4379.60	1.02	4312.51	2.34	166.31

The computational time value of the method proposed in this work reported in parentheses refers to a converted value based on the CPU frequency ratio between the computer used by Dang et al. [2013] and the one used in this work, which resulted in an approximate multiplicative factor of 1.31. Although it is not possible to make a direct comparison, the results suggest that the proposed algorithm took less time in executions. Furthermore, although it has not reached the state-of-the-art in terms of solution quality, the relative deviation reported in Table 2 shows that the results are close.

Table 3: Results comparison

Criteria	Methods			
	PSOiA	PMA	HALNS	ILS + F&O
# BKS	71	81	82	21
Processor	AMD Opteron 2.60 GHz	Intel Core i5, 3.2 GHz	Intel Core i7, 2.80 GHz	Intel Core i7, 3.40 GHz
Random Access Memory	-	4 GB	16 GB	12 GB
Programming Language	C ++	C ++	Java	Julia
Compiler	GNU GCC	-	-	REPL
Operational System	Linux	Windows	Windows 10 64-bit	Windows 10 64-bit
Average CPU (s)	11031.04	996.4	676.08	166.31 (217.87)

5. Conclusion

In this paper, IKS + F&O was introduced as solution approach to solve the Team Orienteering Problem and the method reached a good performance, finding 21 BKS among 82 instances which are the larger ones in the literature. Moreover, the average deviation regarding best solutions found was 1.02%, while the average deviation taking account all executions mean was 2.34%.

The F&O contributed to improving the solutions in terms of objective function values, albeit at the expense of increased computational time, which remained lower compared to other documented works in the literature, as can be seen in Dang et al. [2013], Ke et al. [2016] and Hammami et al. [2020], although computational environments used had been different.

It has been observed that the F&O encounters challenges in enhancing the solution when the metaheuristic's output is very close to optimal, that is, when it operates within a solution space region that no longer permits significant improvements. The former scenario can be elucidated as follows: for the inclusion of a new location to visit or the substitution of a lower-priced one with a higher-priced alternative, sufficient time must be available for executing the new route. Occasionally, this can only be achieved by freeing up or conserving time in other solution space regions that are presently unconsidered in the ongoing F&O iteration.

Thus, future work may focus on implementing new types of subset / subproblem definition for F&O based on distances to the center of gravity of routes or on clustering of locations. Furthermore, it would be interesting to evaluate different sizes of subproblems, number of variables in each subproblem, as well as the time given to the solver.

The size of the subproblems cannot be too small so that they do not allow modifications and improvements to the routes, but neither can they be too large so as not to compromise efficiency in terms of computational time. The worst case would be to solve the problem entirely by the exact method. Research in this direction may show the best combination in terms of solution improvement and computational time.

References

- Archetti, C., Hertz, A., and Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13:49–76. ISSN 13811231.
- Bianchessi, N., Mansini, R., and Speranza, M. G. (2018). A branch-and-cut algorithm for the team orienteering problem. *International Transactions in Operational Research*, 25: 627–635. ISSN 14753995.
- Bouly, H., Dang, D. C., and Moukrim, A. (2010). A memetic algorithm for the team orienteering problem. *JOR*, 8:49–70. ISSN 16142411.
- Butt, S. E. and Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21:101–111. ISSN 0305-0548.
- Chao, I. M., Golden, B. L., and Wasil, E. A. (1996). The team orienteering problem. *European Journal of Operational Research*, 88:464–474. ISSN 0377-2217.
- Dang, D. C., Guibadj, R. N., and Moukrim, A. (2013). An effective PSO-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229:332–344. ISSN 03772217.
- Ferreira, J., Quintas, A., Oliveira, J. A., Pereira, G. A. B., and Dias, L. (2014). Solving the team orienteering problem: Developing a solution tool using a genetic algorithm approach. In Snášel, V., Krömer, P., Köppen, M., and Schaefer, G., editors, *Soft Computing in Industrial Applications*, p. 365–375, Cham. Springer International Publishing. ISBN 978-3-319-00930-8.
- Fischetti, M., Gonzalez, J. J. S., and Toth, P. (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148. ISSN 10919856.
- Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34:307–318.
- Halat, M. and Ozkan, O. (2021). The optimization of UAV routing problem with a genetic algorithm to observe the damages of possible Istanbul earthquake. *Pamukkale University Journal of Engineering Sciences*, 27:187–198. ISSN 1300-7009.
- Hammami, F., Rekik, M., and Coelho, L. C. (2020). A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. *Computers and Operations Research*, 123. ISSN 03050548.
- Ke, L., Archetti, C., and Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54:648–665. ISSN 0360-8352.
- Ke, L., Zhai, L., Li, J., and Chan, F. T. (2016). Pareto mimic algorithm: An approach to the team orienteering problem. *Omega (United Kingdom)*, 61:155–166. ISSN 03050483.
- Kim, B. I., Li, H., and Johnson, A. L. (2013). An augmented large neighborhood search method for solving the team orienteering problem. *Expert Systems with Applications*, 40: 3065–3072. ISSN 09574174.

- Lin, S.-W. (2013). Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing*, 13:1064–1073. ISSN 15684946.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The IRACE package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- Misir, M., Gunawan, A., and Vansteenwegen, P. (2022). Algorithm selection for the team orienteering problem. *Lecture Notes in Computer Science*, 13222 LNCS:33–45. ISSN 16113349.
- Murray, C. C. and Karwan, M. H. (2010). An extensible modeling framework for dynamic reassignment and rerouting in cooperative airborne operations. *Naval Research Logistics*, 57:634–652. ISSN 0894069X.
- Oliveira, J. D. and Scarpin, C. T. (2020). A relax-and-fix decomposition strategy based on adjacent nodes applied to the periodic capacitated arc routing problem (PCARP). *IEEE Latin America Transactions*, 18(03):573–580.
- Souffriau, W., Vansteenwegen, P., Berghe, G. V., and Oudheusden, D. V. (2010). A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37:1853–1859. ISSN 0305-0548.
- Subramanian, A. (2008). Metaheurística iterated local search aplicada ao problema de roteamento de veículos com coleta e entrega simultânea. Master's thesis, Universidade Federal da Paraíba, João Pessoa, PB. Brasil.
- Tang, H. and Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32:1379–1407. ISSN 0305-0548.
- Tsakirakis, E., Marinaki, M., Marinakis, Y., and Matsatsinis, N. (2019). A similarity hybrid harmony search algorithm for the team orienteering problem. *Applied Soft Computing*, 80:776–796. ISSN 1568-4946.
- Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35:797. ISSN 01605682.
- Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Oudheusden, D. V. (2009a). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196:118–127. ISSN 03772217.
- Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Oudheusden, D. V. (2009b). *Metaheuristics for Tourist Trip Planning*, p. 15–31. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-00939-6.
- Vidal, T., Maculan, N., Ochi, L. S., and Penna, P. H. V. (2015). Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science*, 50:720–734. ISSN 15265447.