Journal of Industrial Engineering International (2019) 15:679–693
https://doi.org/10.1007/s40092-019-0315-9

ORIGINAL RESEARCH

Solving multi-objective team orienteering problem with time windows using adjustment iterated local search

Indri Hapsari[1] · Isti Surjandari[1] · K. Komarudin[1]

Received: 10 December 2018 / Accepted: 29 May 2019 / Published online: 6 June 2019

Abstract

One of the problems tourism faces is how to make itineraries more effective and efficient. This research has solved the routing problem with the objective of maximizing the score and minimizing the time needed for the tourist's itinerary. Maximizing the score means collecting a maximum of various kinds of score from each destination that is visited. The profits differ according to whether those destinations are the favorite ones for the tourists or not. Minimizing time means traveling time and visiting time in the itinerary being kept to a minimum. Those are small case with 16 tourism destinations in East Java, and large case with 56 instances consists of 100 destinations each from previous research. The existing model is the Team Orienteering Problem with Time Window (TOPTW), and the development has been conducted by adding another objective, minimum time, become Flexible TOPTW. This model guarantees that an effective itinerary with efficient timing to implement will be produced. Modification of Iterated Local Search (ILS) into Adjustment ILS (AILS) has been done by replacing random construction in the early phase with heuristic construction, continue with Permutation, Reserved and Perturbation. This metaheuristic method will address this NP-hard problem faster than the heuristic method because it has better preparation and process. Contributing to this research is a multi-objective model that combines maximum score and minimum time, and a metaheuristics method to solve the problem faster and effectively. There are calibration parameter with 17 instances of 100 destinations each, small case test using Mixed Integer Linear Programming, and large case test comparing AILS with Multi-Start Simulated Annealing (MSA), Simulated Annealing (SA), Artificial Bee Colony (ABC), and Iterated Local Search. The result shows that the proposed model will provide itinerary with less number of visited destination 4.752% but has higher total score 8.774%, and 3836.877% faster, comparing with MSA, SA, and ABC. While AILS is compared with ILS, it has less visited destination 5.656%, less total score 56.291%, and faster 375.961%. Even though AILS has more efficient running time than other methods, it needs improvement in algorithm to create better result.

Keywords Multi-objective · Team orienteering problem · Time window · Iterated local search · Mixed integer linear programming

Introduction

Tourism in Indonesia has a promising future. Data from the Indonesian Ministry of Tourism (2017) show tourism generated 205.04 trillion rupiah in foreign currency while attracting 14.04 million international tourists and 277 million domestic tourists. Encouraging travel and spending is one of strategic targets of the ministry especially for domestic tourists. To encourage travel, the itineraries must be planned well so that it covers tourists' favorite destinations without using up too much time. Activities that are included initinerary preparation are choosing favorite destinations and arranging them while considering the destinations' operational times and tourists' limited time, then making it into a schedule to follow. The combination becomes more complex with more destinations, constraints, and objectives. This condition has been categorized as a tourism routing problem by previous researchers. This research is interesting because it attempts to fulfill tourists' needs, arrange tourists' favorite destinations, and guarantee the minimum time without breaking the constraints.

The Traveling Salesman Problem (TSP) is a basic routing arrangement that also applies to routing for tourism itineraries. TSP with profit is one of the TSP developments that considers

✉ Indri Hapsari
indri.hapsari63@ui.ac.id

[1] Industrial Engineering, Universitas Indonesia, Depok, Indonesia

every destination to have score or profit in order to prioritize visitations. The Orienteering Problem (OP) starts with a similar principle to TSP with its scores and profit for destinations, but with OP the different scores become the objective. This problem is more complex because it has constraints like time windows at each destination that must be followed to make the itinerary feasible. This condition is similar with tourism industry because tourists will visit their favorite destination or destinations with higher scores. OP is a basic model that is applicated widely in addressing tourism routing problems. There are many developments that arise so that it conforms to tourism situations. For example, the Orienteering Problem with Time Window (OPTW) is an OP model that considers destinations' operational times, or the Team Orienteering Problem (TOP) will apply to group traveling situations.

Previous research has focused on maximizing scores to guarantee that the favorite destinations are the priority of an itinerary. Unfortunately, time can be ineffective because such an itinerary reduces productive hours that can be used to visit other favorite destinations. The previous objective is maximizing scores which guarantees the favorite destinations are inserted in the itinerary, but the effect is a longer time spent traveling and some destinations cannot be visited. This research will combine two objectives: maximizing the score and minimizing the time. The previous models only used one objective. The metaheuristic method replaces the random phase in the heuristic construction to reduce the possibility of an ineffective result. The contribution of this research is more effective and efficient routing for tourists. It is referred to as more effective because it will contain more destinations, while more efficiency is achieved because it can save time for the tourists. The contribution of this model is a better result according to route development by modifying the Team Orienteering Problem with Time Window (TOPTW) model and metaheuristics method, Iterated Local Search (ILS). This modification is needed in order to accommodate another objective, which is minimum time. With ILS, it will start with an adjustment in the heuristic construction before it will perturb randomly.

This paper will be start with an introduction that shows problem and also the gap in previous research. The second section is a literature review to help design state of the art research that links the previous research and the proposed research. The third section is the methodology for the research steps. The results will be laid out in the fourth section along with the analysis. Then the fifth section will consist of the conclusions based on this research.

Literature review

Most of the routing algorithms for the tourism problem use the Orienteering Problem (OP) as a base model, then they are modified to adjust then to certain conditions. The OP is an algorithm that arranges a set of possible destinations with various scores and has a goal of maximizing the total score of the visited destinations. Tsiligirides (1984) approached two heuristics for OP: they are stochastic and deterministic algorithms. OP can be formulated as follows: $G=(V, E)$ is an edge-weighted graph with a score for the destination, s is a starting location, t is a terminal location, T is a positive time limit budget, and the goal is finding a path from s to t (or tour if $s \equiv t$) with a duration of, at most, T. The objective is collecting a maximum total profit from the visited destinations. Souffriau et al. (2008) said OP is the simplest routing model in which each destination has its own score and the objective is to create a single route that will maximize the collected score within a limited time. Li and Fu (2012) developed an equation with the objective of maximizing total profit (1). Let p_i be tourists' preference value or profit for destination V_i, while $x_{ij(t)}$ is 1 or 0, depending on it is visited or not. S_i is the set of successor destinations of destination V_i and P_i is set of predecessors. Equation (2.2) and (2.3) explain the flow conversion in model. Equation (2.4) ensures every destination is visited no more than once. Equation (2.5) and (2.6) guarantee that, if one destination is visited, the arrival time in that destination is the sum of preceding arrival time, service time, and travel time between those destinations. v_t is visiting time on node V_i. Equation (2.7) consists of the start time and end time constraint. If t_0 is starting time, t_i is arrival time at node V_i. In this equation, t_0 is equal to t_1 means V_1 is the starting location and time starts from there. Equation (2.8) and (2.9) are variables constraints. The result for this model is a route that satisfies tourist preference and limitation, and follows the destinations' time windows.

$$Max \sum_{t=t_0}^{T} \sum_{i=2}^{n-1} \sum_{j \in S(i)} p_i x_{ij}(t) \tag{2.1}$$

$$\sum_{t=t_0}^{T} \sum_{j \in S(1)} x_{ij}(t) = \sum_{t=t_0}^{T} \sum_{i \in P(n)} x_{1n}(t) = 1 \tag{2.2}$$

$$\sum_{t=t_0}^{T} \sum_{i \in P(k)} x_{1k}(t) = \sum_{t=t_0}^{T} \sum_{j \in S(k)} x_{kj}, \quad \forall k = 2, \ldots, n-1 \tag{2.3}$$

$$\sum_{t=t_0}^{T} \sum_{j \in S(1)} x_{ij}(t) \leq 1, \quad \forall i = 2, \ldots, n-1 \tag{2.4}$$

$$\sum_{t=t_0}^{T} \sum_{i \in P(j)} \big(t + t_{ij}(t)\big) x_{ij}(t) = t_j, \quad \forall j = 2, \ldots, n \tag{2.5}$$

$$\sum_{t=t_0}^{T} \sum_{j \in S(i)} t x_{1j}(t) = t_i + v t_i, \quad \forall i = 1, \ldots, n-1 \tag{2.6}$$

$$t_1 = t_0, \quad t_n \leq T \tag{2.7}$$

$$t_i > 0, \quad \forall i = 1, \ldots, n \tag{2.8}$$

$$x_{ij}(t) = 0, 1, \quad \forall e_{ij} \in E, \quad \forall t = 1, \ldots, T \tag{2.9}$$

OP can be adjusted to many situations; it can be continued to the metaheuristic method for the optimality, consider many constraints, and has faster computer time. Herzog and Wörndl (2014) defined OP as a score system based on collected scores for chosen destinations while considering the constraints. OP is related to the Knapsack Problem, The Maximum Collection Problem, and The Bank Robber Problem (Li and Fu 2012). A regular heuristic method cannot be used because considering all the constraints takes longer in terms of process time and has a greater possibility of yielding an infeasible result. Hence, the metaheuristic method will be used to run the algorithm. The metaheuristic method concerns efficiency in time and has the flexibility to consider the preferences and limitations of the tourists.

OP development has expanded it widely but OP has remained as the basic model. If OP is usually for single route, the Team Orienteering Problem (TOP) that was developed by Vansteenwegen et al. (2011a, b) will produce multi-routes. Some models can be used to generate single or multi-routes. For example, the Orienteering Problem with Time Windows (OPTW) is for arranging a single route while considering operational hours of destinations (Tsitsiklis 1992; Kantor and Rosenwein 1992; Gendreau et al. 1998a, b; Bansal et al. 2004; Chekuri and Kumar 2004; Chekuri and Pal 2005; Righini and Salani 2009; Vansteenwegen et al. 2011a). OPTW can transform into the Team Orienteering Problem with Time Windows (TOPTW) for multi-routes (Vansteenwegen 2008; Vansteenwegen et al. 2009a; Lin and Yu 2012; Gunawan et al. 2015a). Both OPTW and TOPTW transform into many research topics because of their flexibility (Vansteenwegen et al. 2011b). A moving 2-opt is needed to get a qualified result for OP, but in the limited time this process cannot be implemented efficiently to solve the (T)OPTW model. The solution approach for (T)OPTW is the same as for (T)OP. This concept was confirmed by Tricoire et al. (2010) who modified (T)OP problem solving with limited time to get better results. Model TOPTW was used in Vansteenwegen et al. (2009b) paper, Vansteenwegen et al. (2009c), Labadie et al. (2010), Montemanni et al. (2011), Lin and Yu (2012), Hu and Lim (2014), Cura (2014), Gunawan et al. (2015a), and Gunawan et al. (2015b). The reason why the researchers used the same model was because they can compare the performance of the metaheuristic method in terms of result quality and run time. They used the same case as found here: http://www.mech.kuleuven.be/en/cib/op.

Vansteenwegen et al. (2009b) explained that TOPTW is generally like OP with m-route. For m-route, TOP must determine route m which starts from destination 1 and ends at destination n will maximize score for all routes. The equation for OPTW consists of a variable for each location $i = 1,\ldots, n$ is assigned to a score S_i, has visiting time T_i and a time window $[O_i, C_i]$. The starting point and the end point in every tour are the same and fixed. The time that is needed to travel from location i to j is known for all locations as t_{ij}. Every destination has their own score and total time from traveling and visiting time inside route m cannot exceed the limited time that is already determined as $T_{\max.}$. Each location can be visited at most once and formulated as an integer program (x_{ijd}= 1 if included is in a route, a visit to location i is followed by a visit to location j, 0 otherwise; y_{id}= 1 if location i is in a route, 0 otherwise; s_{id} is the start of the service at location i in route, and M is a large constant).

$$Max \sum_{d=1}^{m} \sum_{i=2}^{n-1} S_i y_{id}, \tag{2.10}$$

$$\sum_{d=1}^{m} \sum_{j=2}^{n-1} x_{1jd} = \sum_{d=1}^{m} \sum_{i=2}^{n-1} x_{ind} = m \tag{2.11}$$

$$\sum_{i=1}^{n-1} x_{ikd} = \sum_{j=2}^{n} x_{kjd} = y_{kd} (k = 2, \ldots, n-1, ; d = 1, \ldots, m) \tag{2.12}$$

$$s_{id} + T_i + c_{ij} - s_{jd} \leq M\left(1 - x_{ijd}\right) (i, j = 1, \ldots, n; d = 1, \ldots, m) \tag{2.13}$$

$$\sum_{d=1}^{m} y_{kd} \leq 1 \quad (k = 2, \ldots, n-1) \tag{2.14}$$

$$\sum_{i=1}^{n-1} \left(T_i y_{id} + \sum_{j=2}^{n} c_{ij} x_{ijd} \right) \leq T_{\max} (d = 1, \ldots, m) \tag{2.15}$$

$$O_i \leq s_{id} (i = 1, \ldots, n; vd = 1, \ldots, m) \tag{2.16}$$

$$s_{id} \leq C_i (i = 1, \ldots, n; vd = 1, \ldots, m) \tag{2.17}$$

$$x_{ijd} y_{id} \in \{0, 1\} (i, j = 1, \ldots, n; d = 1, \ldots, m) \tag{2.18}$$

Objective (2.10) maximizes the total score that is collected. Equation (2.11) guarantees that a route that starts at destination 1 will be ended at location n. Equation (2.12) and (2.13) determine the relationship and timeline of each tour. Equation (2.14) assures that each destination will be visited once. Equation (2.15) explains that each route will finish in limited time. Equations (2.16) and (2.17) guarantee every visit will start during operational hours. Equation (2.18) prevents same route planning.

Gendreau and Lourenco (2003) said that a metaheuristic method gets solutions by combining interaction between local improvement procedures and higher strategy to get an effective process. This process can escape from local optimal results and give better results from a bigger searching area. In the existing method, this has done but it created a trap in local optimal results. In a complex searching area, especially when using one or more neighborhoods to move from one solution to other solution, it could be a constructive or destructive process. To solve this problem, it needs an optimal method that will yield results immediately like metaheuristic methods do. Talbi

(2009) divided metaheuristic methods into several groups: one of them is the most developed single-solution metaheuristics like Simulated Annealing, Tabu Search, Variable Neighborhood Search, Iterated Local Search, and Guided Local Search. Population-based metaheuristics is another common group like Genetic Algorithm, Ant Colony, and Particle Swarm. Labadi et al. (2010) proposed a method that combines the Greedy Randomized Adaptive Search Procedure (GRASP) and the Evolutionary Local Search (ELS). This approach gives solutions with equal quality and reduces computer time significantly compared to the Ant Colony System. When GRASP-ELS compared to ILS, it gives better quality results but has a longer process time. Lin dan Yu (2012) used a heuristic algorithm for TOPTW with Simulated Annealing, and they modified it become Multi-Start Simulated Annealing (Lin and Yu 2017). Gunawan et al. (2016) stated the better solution of the TOPTW model is an equilibrium solution compared to an optimal solution that is centered. They said that models like Local Search and Simulated Annealing (Hu and Lim 2014), Artificial Bee Colony (Cura 2014), and Simulated Annealing and Iterated Local Search (SAILS) (Gunawan et al. 2015a) have worse results than the Well-Tuned—Iterated Local Search that they had designed (Gunawan et al. 2015b). Well-tuned ILS will yield improved solutions with short computerization. This is in line with the survey by Gavalas et al. (2013a) that said Iterated Local Search (ILS), Greedy Randomized Adaptive Search Procedure and Evolutionary Local Search (GRASP-ELS), Ant Colony Systems (ACS) and Iterative Three Components Heuristics (I3CH) can give satisfactory results for problems in TOPTW. Before those pieces of research, Vansteenwegen et al. (2011b) conducted a survey to compare the results of ILS and ACS. Gavalas et al. (2013b) said that TOPTW is a NP-hard problem and that the most efficient metaheuristic method is ILS. Vansteenwegen et al. (2009c) also made same statement because ILS also offers a fair compromise between execution time and a good quality traveling route.

The Iterated Local Search is the fastest algorithm to solve a TOPTW problem with good quality. According to Labadie et al. (2010), ILS has two steps. First, ILS will not take the initial phase randomly. ILS will keep the local optimal score temporarily and choose a new location around the local optimal one. Second, when ILS finds a new local optimal, ILS will decide whether to maintain the initial local optimal or adopt the new one as the basis. If the new one is adopted, it means exploration. If the existing one is used because it is better, it means exploitation. ILS mostly takes the result between the two. This pseudo code below is ILS algorithm with random phase.

```
1: T ← time interval distribution that is possible
2: S←  some solution candidate in random
3: H ←S (existing optimum local basis)
4: Best←S
5: repeat
6:      time← random time in the next term chosen from T
7:      repeat
8:           R← Tweak (Copy(S))
9:           if Quality(R) > Quality(S) then
10:               S← R
11:     until S is ideal solution, or time is up, or total time has accomplished
12:     if Quality(S) > Quality (Best) then
13:          Best←S
14:     H ← NewHomeBase(H, S)
15:     S← Perturb(H)
16: until Best is solution or time is up or total time accomplished
17: return Best
```

ILS combines an insertion step and a shaking step to avoid the optimum solution being local. ILS was developed by Lourenço et al. (2013) for the Traveling Salesman Problem and Scheduling Problem. A set of local solutions is developed rather than repeating a local search randomly. Good balancing between improvement and shaking the intermediate solution is the important thing. These two important steps in ILS, as seen in Vansteenwegen et al. (2009b), are Insertion and Shaking. The first step, Insertion, is a local heuristic step to add new destinations to a route one by one. Before adding the visit to a destination to a route, the verification for all the visitations to other destination must be conducted, thus the arrangement can follow the operational time after the insertion. To get a fast heuristic, there needs to be a fast evaluation for every possibility. The problem is this checking will take a lot of time. For every insertion of a visitation, the minimum insertion time will be determined. The second step, Shake, is needed to exit from the local optimal. As long as this step is one, one or more visitations will be removed. The shake uses two parameters as inputs, with first parameter indicating how many visitations in the sequence will be removed from each route and position in every route to start the removal process. If in the removal process the last destination is reached, removal still continues after the starting point. After the removal, all visitations that are removed will be diverted to the front as much as possible to avoid unimportant waiting time. If it reaches the final destination, removal will continue after the starting point.

From previous research, we knew the problem that has not been fulfilled is the addition of another objective to the previous model TOPTW, making a multi-objective model with two objectives, which are maximizing score and minimizing time. This additional objective is needed because it will make

the itinerary more effective and efficient. Effective because it can accommodate the tourists' favorite destinations, and efficient because it can save the tourists time. The novelty of this research is the modification of the model by using a mathematical formula. This formula will help to achieve the optimal result by considering the real situation or reducing the assumptions. Another novelty is the design of the adjustment to the ILS, because it has a construction phase to replace the random phase. The heuristic construction will give better preparation by producing better results and faster that tourist only needs few time to wait for the result.

Methodology

This research methodology consists of three steps. The first step is analyzing and modifying the previous TOPTW model, second is modifying the ILS method, and third is validating and comparing the result. The new TOPTW model will accommodate multi-objectives, which are the maximum score and minimum time for finding the shortest total time that contains visiting time V_i and traveling time T_{ij}. Constraints will be added to make sure the starting time s_i, and visiting time V_i at a destination will not pass the $T_{\max}$. A complete directed network is $G=(k,i)$, where $k=\{1,2,\ldots,m\}$ is the set of destinations in a route; destinations $\{(i\,j,\}: i\neq j, i\,j\in N\}$ are the set of arcs. Destination 0 is the starting location in each route. Destination $i(i=1,\ldots,n)$ is a tourism destination with a nonnegative score S_i, a nonnegative visiting time V_i, and a time window $[O_i, C_i]$ in each node. Each path must start from and end at node 0. The traveling time from destination i to destination j is known as T_{ij} for each pair of destinations i and j. The maximum time in a route cannot exceed the budget time $T_{\max}$. Every destination can be visited only once. The objective is maximizing the total score from visiting the destinations, and minimizing the time spent. The minimum time is found by adding the visiting and traveling times for all destinations and routes that are visited, and the traveling time from the starting location to the first location. FTOPTW can be modeled as a mixed integer linear program as follows.

Decision Variables:

$y_{ik}=1$ if destination i is visited in route k, 0 otherwise
$x_{ijk}=1$ if in route k, a visitation to destination i is followed by a visit to destination j, 0 otherwise

Parameters:

i: number of destinations.
k: number of routes.
S_i: score of customer i.
$T_{\max}$: budget time in a route.
T_{ij}: traveling time from destination i to destination j.
s_i: arrival time for destination i.
V_i: visiting time for destination i.
O_i: opening time in the time window for destination i.
C_i: closing time in the time window for destination i.
L: Large constants.

Model:

$$Max \sum_{k=1}^{m}\sum_{i=1}^{n} S_i y_{ik} \tag{3.1}$$

$$Min \sum_{k=1}^{m}\sum_{i=1}^{n} s_{ik} + V_i \tag{3.2}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n} x_{ijk} = \sum_{k=1}^{m}\sum_{i=1}^{n} x_{jik} = m \tag{3.3}$$

$$\sum_{i=1,i\neq l}^{n} x_{ilk} = \sum_{j=1,j\neq l}^{n} x_{ljk} = y_{lk} \quad \forall l = 1,\ldots,n; \quad \forall k = 1,\ldots,m \tag{3.4}$$

$$s_{ik} + V_i + T_{ij} - s_{jk} \leq L(1 - x_{ijk}) \quad \forall i,j = 1,\ldots,n; \quad \forall k = 1,\ldots,m \tag{3.5}$$

$$\sum_{i=1}^{n}\sum_{d=1}^{m} y_{ik} \leq 1 \tag{3.6}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n}\left(V_i y_{ik} + \sum_{j=1}^{n} T_{ij} x_{ijk}\right) \leq T_{\max} \tag{3.7}$$

$$O_i y_{ik} \leq s_{ik} \quad \forall i = 1,\ldots,n; \quad \forall k = 1,\ldots,m \tag{3.8}$$

$$C_i y_{ik} \geq s_{ik} \quad \forall i = 1,\ldots,n; \quad \forall k = 1,\ldots,m \tag{3.9}$$

$$s_{ik} + V_i \leq T_{\max} \quad \forall i = 1,\ldots,n; \quad \forall k = 1,\ldots,m \tag{3.10}$$

$$x_{ijk}, y_{ik} \in \forall\{0,1\}; \quad i,j = 1,\ldots,n; \quad \forall k = 1,\ldots,m \tag{3.11}$$

The objective function (3.1) maximizes the total score and (3.2) minimizes the time. The total score comes from the score that is summed from every destination that is in the schedule. Total time comes from all the starting times and all the visiting time spent at the places visited. Maximum total score is the priority before continuing with the minimum total time. Constraint (3.3) determines the number of routes used. Constraints (3.4) and (3.5) maintain the connectivity and keep track of the timeline of each route. Constraint (3.6) makes sure that every destination is visited at most once. Constraint (3.7) guarantees that the time budget of each route is not violated. Constraints (3.8) and (3.9) are

time window constraints. Constraint (3.10) guarantees that the starting time and visiting time for each destination is not greater than the time budget. Constraint (3.11) specifies that all variables are binary. To make sure this formulation is feasible to run, we used a case study with 16 tourism destinations in East Java.

The second step is analyzing and modifying the previous ILS method. The previous metaheuristics method still uses a random phase at the beginning, which makes the probability of getting feasible results small. For modifying the existing model, we replaced the random phase with a heuristic construction that considers all the constraints, to make a more feasible result and become a candidate for optimization. The Adjusted Iterated Local Search (AILS) pseudo code can be read as below.

```
1: Tmax, Oi, Ci← time budget in every route k and time window for destination i
2: Allocation i with the biggest score in every route with considering Tmax, Oi, Ci
3: repeat
4: Generate best solution candidate with minimum time by permutation for every route
5: Check Tmax, Oi, Ci
6:      if destination breaks the line
7:              then return to feasible route → continue
8:      else
9:              keep the total time
10.             choose the minimum time → BS1
11: until all route has accomplished
12: repeat
13: Generate best solution candidate with minimum time by reverse for every route
14: Check Tmax, Oi, Ci
15:     if destination breaks the line
16:             then return to BS1→ continue
17:     else
18:             count the total time
19:                     if BS1 < total time
20:                             then BS1 → BS2
21:                             return to previous route
22:                     else
23:                             total time → BS2
24: until all route has accomplished
25: Generate best solution candidate with minimum time by Perturbation for all routes
26: repeat
27: Check Tmax, Oi, Ci
28:     if destination breaks the line
29:             then return to BS2 → continue
30:     else
31:             count the total time
32:                     if BS2 < total time
33:                             then BS2 → BS3
34:                             return to previous route
35:                     else
36:                             total time → BS3
37: until all route has accomplished
38: Best solution accomplished
```

In the beginning, AILS will take visiting and traveling times from the database, then the destination with the biggest score will be allocated to each route without violating the maximum time $T_{\max}$ and operational hours $[O_i, C_i]$ of the destination. Permutation will guarantee every destination has the same probability of contributing the best solution. The maximum score (Total Score or TS) and minimum time (Total Time or TT) become BS1. BS1 continues to the second process, reverse, in which we will flip the sequence in every route, and recount the TT. If this is less than BS1 and fulfills the time requirement, the result will be placed into BS2. Otherwise, BS1 will become BS2. Repeat this step until all the routes have been flipped. The last step is Perturbation for all the routes. Based on the percentage, the destinations will be moved to another route randomly. If there is a better TT, it could be BS3. If they are getting worse, BS2 will be set as BS3.

The third step is comparing the metaheuristics AILS with other methods in previous papers. Some researches proposed multi-objective team orienteering problem to maximize various profit from each destinations (Martín-Moreno and Vega-Rodríguez 2018; Schilde et al. 2009; Rezki and Aghezzaf 2017), while Hu et al. (2018) developed it with time windows. Mirzaei et al. (2017) research has objective to balance the profit among the routes. Another multi-objective for time dependent orienteering problem was designed by Mei et al. 2016). Multi-objective that is combined maximum profit and minimum travel cost was developed by Bederina and Hifi (2017) as proposed multi-objective team orienteering problem without time windows. This research is focused with multi-objective orienteering problem with time window, and the objectives are maximum profit and minimum time. The previous multi-objectives research did not use single profit for each destination and did not have time window, while this research will compare with single objective research, because they presented the multi-result like total destination visited, total profit, total time, and running time. AILS will compare to other methods like MSA, SA and ABC in Lin and Yu (2017) and ILS in Vansteenwegen et al. (2009b). There is a calibration parameter that is required before we run AILS. There are 17 problem instances, each consisting of 100 destinations, from Solomon's datasets (Lin and Yu 2017). The percentage will be set at 20%, 40%, 60% and 80% to find out which one will give the shorter running time. The comparison that will be used is total score, visited destination and running time. All the result for each problem are recorded and the gap for the running time is calculated as the average $(RT_{\text{BS}}–RT_{\text{AILS}})/RT_{\text{AILS}}$ where RT_{BS} is the best solution running time in comparing methods and RT_{AILS} is the running time from the AILS method. It will do the same for Visited Destination (VD) and Total Score (TS). Other methods that will be compared are Multi-Start Simulated Annealing (MSA) (Lin and Yu 2017), Simulated Annealing

(SA) (Lin and Yu 2012), Artificial Bee Colony (ABC) (Cura 2014), and Iterated Local Search (ILS) (Vansteenwegen et al. 2009b). We use the same case study from Solomon's datasets, and follow the number of routes previous papers used, for a fair comparison.

Algorithm and metaheuristic

This section will describe the calculation process and solution that is obtained from mathematic formulation for the TOPTW algorithm. However, the proposed TOPTW has more objectives. In this new algorithm, the objectives are to maximize the score and minimize the time. The addition objectives make the optimization program perform it as two serial process. When we run the program in Lingo 17 with 16 destinations in Mixed Integer Linear Programming, the program runs for more than 45 min to get the global optimum total score. This time is too long for tourists, who need a very short time to give them a result. The program's running time must be limited, at most to just 1 s, to get a maximum score and minimum time. That is why we need a metaheuristics method like the Adjustment Iterated Local Search (AILS) to speed up the algorithm to get a solution that is near the global optimal. Because the objective of the program is to minimize its running time, so the tourist is only waiting a short time, we will focus on this. But to make sure the program is good enough, compared to other methods, we also record the total score and number of visited destinations.

Solution representation

A solution for FTOPTW is started by sorting the priority destinations in descending order. The sorting of n destinations (1,2,…,n) and m routes. Destination 1 is the starting and ending destination as the route will return to the same destination it began at. Each destination has the score S_i which represents the priorities to visit. Destinations with the highest scores will be distributed in each route. For example, the 16 destinations with the highest scores are shown in Table 1. Because tourists only have 2 days to complete their journey, there are 2 routes. The allocation would be destination 2 in route 1; destination 8 in route 2, destination 6 in route 1, and so on. If the score is the same, the destination can be chosen arbitrarily. The allocation will be based on minimum visiting times spent in each route, and continue until it almost reaches the maximum time $T_{\max}$. Traveling time between destinations will be inserted, then it continues with checking $T_{\max}$, to return destinations with the lowest scores back to the list. Time window constraints [O_i, C_i] will be checked to make sure there is no violation. For every destination that is returned to the list, and there is still some time remaining in a route, it could be replaced with another destination that has a visiting time which is less than the remaining time. Every time a new destination is inserted, the process to check $T_{\max}$ and [O_i, C_i] will be repeated.

Illustration of solution representation

Table 1 gives a FTOPTW instance with 16 tourism destinations in East Java, and a tourist decides to do the journey in 2 days or two routes. Each destination has coordinates (X, Y), visiting time (V), score (S), opening time (O) and closing time (C) for its time window. All the information is taken from Google Maps.

An example of a solution for this FTOPTW is given in Fig. 1. First the destinations will be arranged, based on the highest score to the lowest. Then one by one the destinations will be included in each route by considering their visiting time, to make sure it will not violate the time budget or $T_{\max}$. The number of days is the same as the number of routes, so how many days the tourists want to spend is the same as the number of routes that will be created. Each pair of destinations has X and Y coordinates that include the traveling time that comes multiplied by the Euclidian distances with the vehicle's speed. For running the formulation in Lingo 17, we used the traveling times provided in Google Maps.

As can be seen in Fig. 1, the first destination in the first route is destination 2. Line represents traveling time, bar represents visiting time for each destination. It needs 52 min to reach destination 2 from destination 1 (the starting point). It takes 120 min to visit destination 2. Then it continues to destination 4, which takes 12 min, so the starting time at destination 4 is 184 min. The visitation then goes on to destinations 9 and then 12. The time budget $T_{\max}$ will end after or equal with end of the visiting time at the last destination. After the last destination, the tourist will return to destination 1. All the visited destinations must be checked for not violating the time window. The second route will do the same. The total score of a given initial solution can be easily calculated from the total score collected in all the routes.

The AILS procedure in Fig. 2 is started with the permutation. All the destinations in one route will get pairwise exchanges and for each exchange the total time must be counted. Total scores are not an issue anymore because all the destinations that are included in the route are already the highest scoring ones. After all the exchanges then every sequence that is created is compared, then it will choose the minimum total time. Checking must be done for the time window, because the time budget is already guaranteed at the beginning. For example the permutation process gives the same result as the optimization. The second step is to reverse the sequence for each route. For example in route 1, the first sequence is 2–4–9–12. When reversed, the sequence becomes 12–9–4–2. Then it must

Table 1 A FTOPTW example with 16 destinations

Destination	No	*X*	*Y*	*V*	*S*	*O*	*C*
Starting point	1	−7.70	111.00	0	0	0	600
Banyulawe Waterfall	2	−7.75	111.68	120	4.7	0	960
Krecek Ndenu Waterfall	3	−7.71	111.66	120	4.0	0	960
Slampir Waterfall	4	−7.73	111.69	120	4.1	0	960
Dumilah Waterpark	5	−7.62	111.53	120	3.7	0	960
Wilis Mountain	6	−7.81	111.75	120	4.3	0	960
Brem Citra Rasa	7	−7.51	111.68	90	4.0	1	540
Great mosque Taman	8	−7.63	111.52	60	4.5	0	780
Kresek Monument	9	−7.70	111.63	60	4.2	0	900
Punden Lambang Kuning	10	−7.61	111.57	60	4.4	1	540
Sun City Waterpark	11	−7.62	111.53	210	4.1	1	540
Madiun Umbul Square	12	−7.78	111.52	150	3.8	1	900
Tirtonirmolo Waterpark	13	−7.46	111.42	180	4.0	0	780
Bening Reservoir	14	−7.54	111.8	180	4.0	1	780
Dawuhan Reservoir	15	−7.58	111.62	180	3.9	1	780
Kedung Brubus Reservoir	16	−7.45	111.7	180	3.6	1	780
Grape Agritourism	17	−7.69	111.64	120	4.1	1	780

be checked for the time budget and time window, and if it is appropriate, it will continue with comparing which one has the minimum total time. We will use the sequence with the new minimum time, otherwise it will be back to the previous sequence. As we can see in Fig. 3, the reverse has a better result so we continue with this sequence. For the third process, Perturbation, there is a certain percentage that represents the number of destinations that will be moved randomly. For example in our case study, there are 16 destinations. If we use 20%, it means 3.2 destinations. This must be rounded up to an even number, so there are 4 destinations that must be moved. The moving could be inside one route or across to another route. In our example destination 9 in route 1, and originally in the second sequence, is moved to route 2 and becomes a third sequence. Destination 17 from route 2 will replace destination 9 in route 1. Time budgets and time windows must be checked for every route, and if they give a better total time the route will change. Figure 3 shows the illustration how the route can change. The red pin shows the starting point and ending point. In the left figure, there are two routes and each is distinguish by the line style, solid for route 1 and dashes for route 2. Then in the right figure, the routes are changed.

Experimental results and discussion

The proposed FTOPTW algorithm and AILS method was implemented in Java by Eclipse. The experiments are carry out on a computer with an Intel Core m51.10 GHz CPU. To demonstrate the applicability of the FTOPTW algorithm,

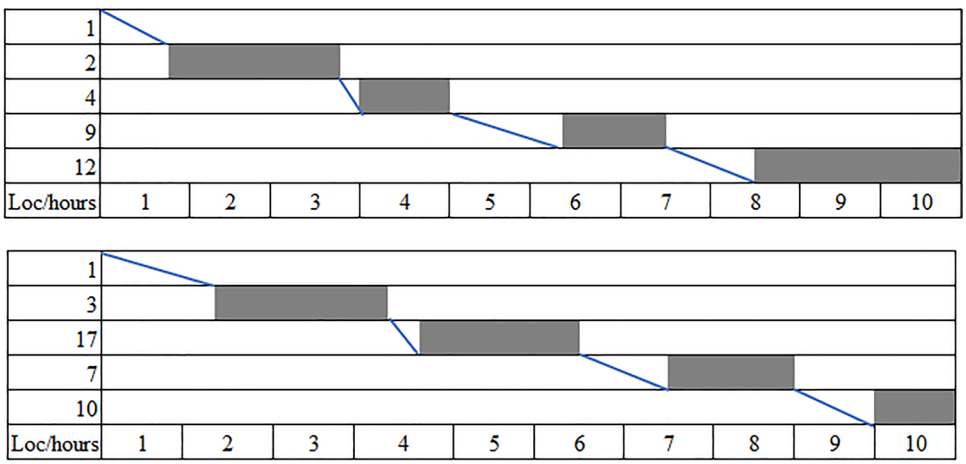


Fig. 1 The FTOPTW result

94 (17 + 56 + 21) problem instances are generated, based on Solomon's datasets. These problems can be classified as the problems for parameter calibration, comparing problems with Simulated Annealing (MSA and SA) and Artificial Bee Colony (ABC), and the last is comparing problems with Iterated Local Search (ILS). Each instances consists of 100 destinations with coordinates, visiting times, scores and time windows.

Parameter calibration

Since the computational results of the FTOPTW algorithm are affected by the parameter settings, the parameter calibration used 17 problem instances. In the experiment, several values of the perturbation percentages are used. Each parameter has four levels (20%, 40%, 60% and 80%) so 68 values were generated. We compared the running times as the important factor to achieve. As presented in Table 2, the 40% perturbation has the minimum average time from the 68 trials. It has a 1.159 s running time for each instance that has 100 destinations that could be visited. It means a lower perturbation does not guarantee a shorter running time. Many factors may influence the computational running time, including CPU speed, memory size, operating system, compiler, computer program, and precision.

Comparing results obtained by AILS, MSA, SA, ABC and ILS

Computational experiments were carried out on a large problem set to compare the performance of AILS with the other metaheuristic methods like MSA, SA, ABC, and ILS. Given the computational complexity of FTOPTW, a global optimal solution needs 45 min to solve for 16 destinations. Because it is done in sequence, the total score is first achieved with the global optimum, then followed by the minimum time. But in an iterated local search, the total score is near optimal, and the total time will be adjusted. The sequence will be exchanged to find the minimum time.

The best solutions obtained by AILS, MSA, SA, and ABC for each instance are shown in Table 3. In this table, the instance column displays the name of the problem instance, while the numbers are number of routes, total score, number of visited destinations, and running time in seconds. With this number of tours, it should be feasible to visit every location, and hence the optimal result equals the sum of all the

Fig. 2 The AILS procedure

	Route 1					Route 2					
Permutation	1	2	4	9	12	1	3	17	7	10	1
	Route 1					Route 2					
Reverse	1	12	9	4	2	1	10	7	17	3	1
	Route 1					Route 2					
Pertubation	1	12	17	4	2	1	10	7	9	3	1

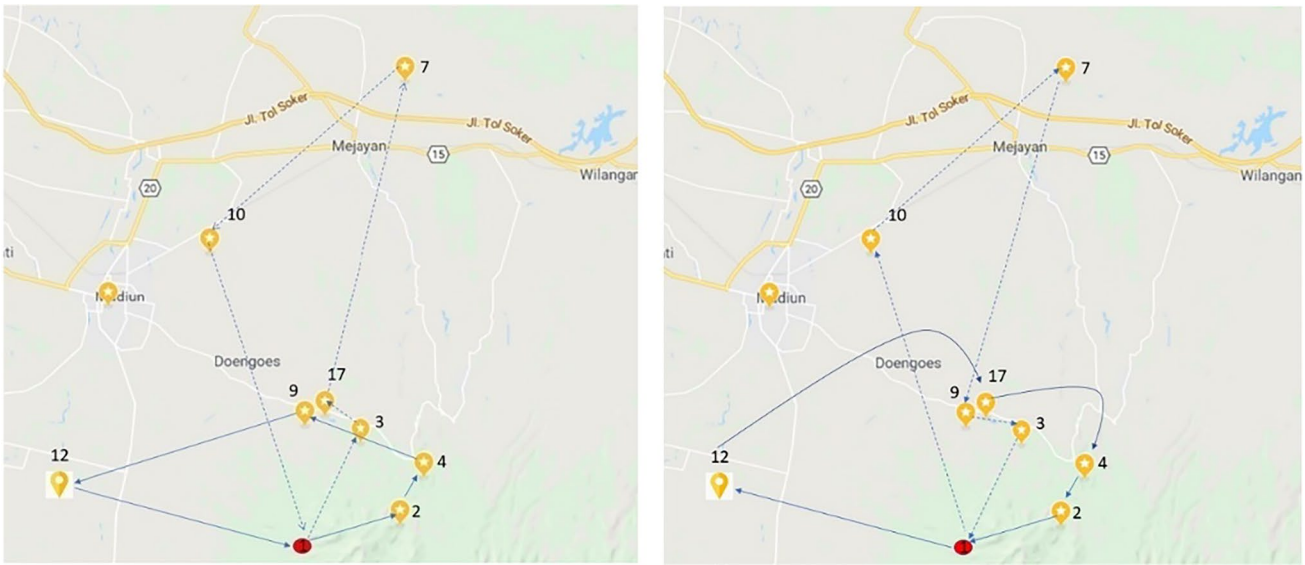


Fig. 3 A visual illustration of the example solution given in

scores. Because this is an orienteering problem, it is possible not to visit all the locations. It tries to select as many visits as possible, and to design a feasible route between them. The difficulty of solving instances is not only related to the number of locations that can be visited in each route, but also to the number of routes that can be generated. Table 3 compares the score obtained by the AILS with the best known solution from comparing the other methods. The best know result for each instance is either the optimal solution or the best result of the runs of the method. Columns 1 and 2 give the instance's name and total score. The third column presents the score obtained by the AILS. In the next column the number of visited locations in the solutions is presented and the running time in the last column is expressed in seconds.

There are three kinds of gaps that we will measure, they are the gaps for the number of Visited Destinations (*VD*), Total Score (*TS*), and Running Time (*RT*). The gap for the running time is calculated as the average $(RT_{BS}–RT_{AILS})/RT_{AILS}$ where RT_{BS} is the best solution running time in the compared method and RT_{AILS} is the running time from the AILS method. The same formula can also be done for the *VD* gap and *TS* gap. Then all the instances' gaps will be averaged again and converted into a percentage. The percentage could be positive because the other methods have a higher result, or negative because the other methods have a lower result. In comparison with Table 3, the VD gap is 4.752%. It means the other methods visited more destinations than AILS. But if we compare it with the TS gap, it has −8.744%, showing AILS has a higher score than the other methods. It means even though the other methods visited more destinations, they were not effective journeys because they only visited more lower scoring destinations. AILS will give a more effective journey by visiting only the higher ones, so tourist spend less time traveling. The percentage of total time gives a very good result, the difference being 3836.877%. It means the AILS running time is much more efficient than that of the other methods.

In comparison with ILS, we also measured the gaps for the number of visited destinations, total score and running time. Because there are three routes (m = 2, m = 3, m = 4) the gap is obtained from the average gap of all the routes. In comparison with Table 4, the number of visited destinations has a 5.656% difference. It means ILS visited more destinations than AILS. The total score difference is 56.291% higher for ILS, thus ILS can visit more destinations and collect higher scores. But if we compare this with the running times, ILS needs more time to give a good result, the different being 375.961%. It means the AILS running time is more efficient than ILS, and this is important for the tourist that has no time to wait for the program to load.

Table 2 Parameter values tested in the calibration

Instance	Percentage perturbation (*p*)			
	20	40	60	80
c101	1.923	1.279	1.343	1.086
c102	0.611	0.811	0.749	0.725
c103	0.860	0.986	0.829	0.820
c104	1.094	1.355	1.268	1.388
c105	1.484	1.210	1.324	2.125
c106	1.235	1.310	1.155	1.337
c107	1.350	1.193	1.654	1.116
c108	1.348	1.332	1.293	1.559
c109	1.240	1.378	1.206	1.241
c201	1.192	1.143	1.420	1.141
c202	1.135	1.140	1.122	1.281
c203	1.028	1.367	1.071	1.191
c204	1.029	1.082	1.089	1.057
c205	0.967	0.975	1.016	1.039
c206	1.091	1.050	1.157	1.056
c207	1.153	0.984	1.048	1.037
c208	1.089	1.104	1.038	1.026
average	1.166	**1.159**	1.164	1.190

Conclusion and future research

The main contribution of this paper is an algorithm that solves the Team Orienteering Problem with Time Windows (TOPTW) fast and effectively. This work concerns the FTOPTW, which is a challenging extension of TOPTW. To reduce the gap between theory and industrial practice, this work develops an MILP model of FTOPTW, which incorporates the maximum score and minimum time. In a small set the algorithm can give a global optimal, based on the maximum score and minimum time. But because the running time was more than 45 min, it will not be suitable as a tourist's mobile application. There are calibration parameters, a small case test using Mixed Integer Linear Programming, and a large case test comparing AILS with MSA, SA, ABC and ILS. On a large set of test instances, AILS is faster than MSA, SA, ABC and ILS, which will satisfy the tourist. This is achieved by the permutation of all the destinations, that guarantees the best pairwise will be found, which can be reversed as an alternative to get an even better result, and perturbation with a certain percentage will move the destination randomly between routes. The gap for the number of destinations visited is better than MSA, SA, and ABC, but slightly lower than the ILS result.

For future research, having an algorithm that has a near global optimum and a minimum running time is needed to make tourists more satisfied. Combining the algorithm with a metaheuristic method, or hybridize other algorithms is good, but the best fit must pass much research to achieve the objective. The constraints, such as the number

Table 3 Computational results for AILS, MSA, SA and ABC method

Instance	AILS				MSA				SA				ABC				GAP		
	Path	Visited	Score	Time	Path	Visited	Score	Time	Path	Visited	Score	Time	Path	Visited	Score	Time	Visited	Score	Time
c101	5	56	920	1.159	5	50	805	28.340	5	51	815	29.440	5	51	815	27.540	−0.095	−0.118	23.538
c102	5	56	920	0.899	5	56	945	30.760	5	55	935	26.180	5	56	945	32.490	−0.006	0.024	32.159
c103	5	56	920	1.121	5	56	995	29.820	5	56	985	28.310	5	57	995	29.560	0.006	0.078	25.075
c104	5	56	920	1.086	5	57	1005	25.370	5	58	1015	32.500	5	59	1025	37.030	0.036	0.103	28.128
c105	5	56	920	1.059	5	52	855	33.520	5	51	845	28.280	5	52	855	31.060	−0.077	−0.074	28.229
c106	5	56	920	1.069	5	52	875	30.170	5	52	875	31.040	5	52	875	38.900	−0.071	−0.049	30.216
c107	5	56	920	1.203	5	54	905	30.770	5	54	905	25.650	5	54	905	34.050	−0.036	−0.016	24.068
c108	5	56	920	0.957	5	55	915	33.950	5	55	915	35.720	5	56	925	38.800	−0.012	−0.002	36.781
c109	5	56	920	0.973	5	59	1015	38.570	5	56	985	29.010	5	59	1015	45.910	0.036	0.092	37.880
r101	5	47	679	1.036	5	36	310	23.820	5	34	308	31.380	5	34	304	24.520	−0.262	−0.547	24.650
r102	5	47	679	0.741	5	49	560	35.830	5	48	557	25.620	5	51	566	37.970	0.050	−0.174	43.723
r103	5	47	679	1.160	5	52	654	36.660	5	53	651	34.470	5	54	657	42.850	0.128	−0.037	31.753
r104	5	47	679	0.940	5	57	728	32.730	5	57	709	31.800	5	56	737	52.950	0.206	0.067	40.660
r105	5	47	679	1.081	5	43	504	27.990	5	43	506	27.870	5	45	511	32.540	−0.071	−0.253	26.259
r106	5	47	679	0.919	5	51	655	26.070	5	50	643	32.760	5	50	649	47.560	0.071	−0.044	37.589
r107	5	47	679	1.090	5	54	670	30.280	5	52	667	25.630	5	54	669	51.630	0.135	−0.015	31.887
r108	5	47	679	1.020	5	56	736	39.730	5	55	728	28.340	5	55	728	50.770	0.177	0.076	37.837
r109	5	47	679	0.977	5	49	616	27.540	5	50	621	25.620	5	50	617	34.820	0.057	−0.090	29.017
r110	5	47	679	1.028	5	51	650	26.440	5	52	651	32.970	5	51	650	47.930	0.092	−0.042	33.805
r111	5	47	679	0.906	5	56	680	32.730	5	53	684	35.350	5	54	676	42.280	0.156	0.001	39.603
r112	5	47	679	0.977	5	56	745	35.750	5	56	748	29.740	5	55	743	50.680	0.184	0.098	38.635
rc101	5	46	790	1.226	5	45	583	22.710	5	45	583	20.510	5	45	583	33.310	−0.022	−0.262	19.808
rc102	5	46	790	1.125	5	50	693	26.390	5	51	689	27.480	5	48	683	34.630	0.080	−0.129	25.222
rc103	5	46	790	1.010	5	51	755	29.810	5	49	740	26.380	5	50	755	55.210	0.087	−0.051	35.766
rc104	5	46	790	1.298	5	54	846	28.770	5	53	851	34.050	5	53	847	38.100	0.159	0.073	24.917
rc105	5	46	790	1.155	5	45	644	31.170	5	48	651	25.630	5	48	656	36.390	0.022	−0.177	25.895
rc106	5	46	790	1.501	5	50	666	29.730	5	50	666	25.400	5	48	661	41.920	0.072	−0.159	29.780
rc107	5	46	790	1.190	5	49	730	29.810	5	49	724	33.360	5	50	724	36.690	0.072	−0.081	26.972
rc108	5	46	790	0.976	5	52	832	40.160	5	51	828	27.000	5	53	839	50.290	0.130	0.054	39.113
c201	3	100	1760	1.934	4	100	1510	32.550	3	93	1515	34.510	4	100	1510	25.500	−0.023	−0.141	14.953
c202	3	100	1760	1.797	3	94	1525	40.570	4	100	1510	32.080	4	100	1510	26.900	−0.020	−0.139	17.466
c203	3	100	1760	1.917	4	100	1510	28.360	4	100	1510	27.980	4	100	1510	24.830	0.000	−0.142	13.114
c204	3	100	1760	1.383	4	100	1510	20.210	4	100	1510	23.540	4	100	1510	25.750	0.000	−0.142	15.751
c205	3	100	1760	1.062	3	95	1535	29.070	3	95	1535	30.660	4	100	1510	22.500	−0.033	−0.133	24.810

Table 3 (continued)

Instance	AILS				MSA				SA				ABC				GAP		
	Path	Visited	Score	Time	Path	Visited	Score	Time	Path	Visited	Score	Time	Path	Visited	Score	Time	Visited	Score	Time
c206	3	100	1760	0.658	3	93	1515	25.840	3	94	1525	28.960	4	100	1510	23.990	−0.043	−0.138	38.914
c207	3	100	1760	0.528	3	96	1545	32.780	4	100	1510	25.610	5	100	1435	21.620	−0.013	−0.150	49.511
c208	3	100	1760	0.486	4	100	1510	22.370	3	94	1525	30.100	4	100	1510	21.160	−0.020	−0.139	49.501
r201	3	100	1312	0.792	3	93	1183	46.280	3	93	1198	48.130	4	100	1158	30.620	−0.047	−0.101	51.622
r202	3	100	1312	0.556	3	97	1226	36.720	3	97	1226	37.990	4	100	1158	22.440	−0.020	_−0.083	57.243
r203	3	100	1312	0.578	3	100	1233	35.060	3	100	1233	29.230	4	100	1158	23.020	0.000	−0.079	49.352
r204	3	100	1312	0.546	4	100	1158	21.360	4	100	1158	20.810	4	100	1158	19.960	0.000	−0.117	36.930
r205	3	100	1312	0.651	3	100	1233	37.390	3	100	1233	35.790	4	100	1158	20.110	0.000	−0.079	46.768
r206	3	100	1312	0.512	3	97	1227	27.660	3	100	1233	36.060	4	100	1158	22.270	−0.010	−0.081	54.983
r207	3	100	1312	0.620	3	98	1230	23.540	3	100	1233	24.640	3	100	1233	25.580	−0.007	−0.061	38.656
r208	3	100	1312	0.459	3	100	1233	20.170	4	100	1158	20.010	4	100	1158	19.380	0.000	−0.098	42.253
r209	3	100	1312	0.542	3	94	1218	26.810	3	100	1233	41.530	4	100	1158	23.960	−0.020	−0.083	55.765
r210	3	100	1312	0.522	3	100	1233	34.060	3	99	1231	29.870	4	100	1158	20.560	−0.003	−0.080	52.953
r211	3	100	1312	0.485	3	100	1233	24.520	3	100	1233	26.900	4	100	1158	20.830	0.000	−0.079	48.656
rc201	3	82	1684	0.430	3	92	1434	47.050	3	93	1448	39.480	4	100	1424	24.270	0.159	−0.148	84.891
rc202	3	82	1684	0.473	3	99	1489	34.310	3	98	1484	33.670	4	100	1424	21.510	0.207	−0.130	62.066
rc203	3	82	1684	0.468	4	100	1424	21.150	4	100	1424	26.010	4	100	1424	20.450	0.220	−0.154	47.155
rc204	3	82	1684	0.506	4	100	1424	24.690	4	100	1424	24.390	4	100	1424	19.220	0.220	−0.154	43.993
rc205	3	82	1684	0.447	3	92	1449	38.650	3	94	1459	38.840	4	100	1424	23.980	0.163	−0.143	74.667
rc206	3	82	1684	0.444	3	98	1494	34.490	4	100	1424	28.580	4	100	1424	23.030	0.211	−0.141	63.640
rc207	3	82	1684	0.479	[4]	100	1424	29.220	4	100	1424	29.630	4	100	1424	19.940	0.220	−0.154	53.830
rc208	3	82	1684	0.435	[4]	100	1424	22.200	4	100	1424	25.510	4	100	1424	19.140	0.220	−0.154	50.226
														Average gap percentage			4.752	−8.744	3836.847

Table 4 Computational results for AILS and ILS method

Instance	AILS											
	Path	Visited	Score	Time	Path	Visited	Score	Time	Path	Visited	Score	Time
c101	2	23	450	0.435	3	32	590	0.374	4	40	770	0.324
c102	2	23	450	0.412	3	32	590	0.328	4	40	770	0.298
c103	2	23	450	0.468	3	32	590	0.300	4	40	770	0.327
c104	2	23	450	0.499	3	32	590	0.320	4	40	770	0.291
c105	2	23	450	0.437	3	32	590	0.358	4	40	770	0.343
c106	2	23	450	0.466	3	32	590	0.310	4	40	770	0.316
c107	2	23	450	0.595	3	32	590	0.318	4	40	770	0.281
c108	2	23	450	0.397	3	32	590	0.362	4	40	770	0.400
c109	2	23	450	0.448	3	32	590	0.330	4	40	770	0.309
r101	2	19	281	0.339	3	30	434	0.498	4	37	553	0.340
r102	2	19	281	0.313	3	30	434	0.364	4	37	553	0.315
r103	2	19	281	0.335	3	30	434	0.564	4	37	553	0.331
r104	2	19	281	0.338	3	30	434	0.436	4	37	553	0.339
r105	2	19	281	0.355	3	30	434	0.395	4	37	553	0.353
r106	2	19	281	0.297	3	30	434	0.342	4	37	553	0.308
r107	2	19	281	0.328	3	30	434	0.329	4	37	553	0.342
r108	2	19	281	0.381	3	30	434	0.352	4	37	553	0.332
r109	2	19	281	0.314	3	30	434	0.338	4	37	553	0.404
r110	2	19	281	0.337	3	30	434	0.392	4	37	553	0.295
r111	2	19	281	0.341	3	30	434	0.403	4	37	553	0.316
r112	2	19	281	0.336	3	30	434	0.320	4	37	553	0.328

Instance	ILS												GAP		
	Path	Visited	Score	Time	Path	Visited	Score	Time	Path	Visited	Score	Time	Visited	Score	Time
c101	2	21	590	1.4	3	29	790	1.1	4	39	1000	3.8	−0.063	0.315	4.560
c102	2	22	650	0.9	3	32	890	2.1	4	43	1090	1.8	0.021	0.453	3.624
c103	2	22	700	1.2	3	33	960	2.2	4	44	1150	2.5	0.042	0.552	4.388
c104	2	22	750	1.5	3	34	1010	1.3	4	45	1220	3.0	0.063	0.646	4.225
c105	2	21	640	0.8	3	30	840	1.0	4	40	1030	1.8	−0.042	0.387	2.1633
c106	2	20	620	0.8	3	30	870	1.1	4	40	1040	2.1	−0.053	0.398	2.663
c107	2	22	670	1.4	3	33	900	1.5	4	43	1100	2.0	0.032	0.475	3.104
c108	2	22	670	0.8	3	33	900	1.2	4	44	1100	3.6	0.042	0.475	3.832
c109	2	22	710	0.9	3	33	950	2.0	4	45	1180	2.5	0.053	0.569	3.968
r101	2	13	330	0.4	3	31	481	0.8	4	28	601	1.4	−0.163	0.114	1.209
r102	2	21	508	0.9	3	31	685	1.0	4	39	807	1.7	0.058	0.577	2.629
r103	2	20	513	0.9	3	31	720	2.0	4	42	878	2.2	0.081	0.665	3.146
r104	2	22	539	1.5	3	34	765	1.5	4	45	941	3.8	0.174	0.771	5.110
r105	2	18	430	0.8	3	27	609	2.3	4	35	735	2.9	−0.070	0.399	4.440
r106	2	21	529	0.9	3	32	719	2.1	4	41	870	3.5	0.093	0.670	5.864
r107	2	21	529	1.0	3	33	747	1.1	4	44	927	3.3	0.140	0.737	4.405
r108	2	24	549	1.4	3	36	790	3.1	4	47	983	3.2	0.244	0.831	6.230
r109	2	22	498	0.5	3	31	699	1.8	4	40	866	2.1	0.081	0.627	3.167
r110	2	22	515	1.0	3	32	711	1.4	4	42	870	2.0	0.116	0.653	3.297
r111	2	23	535	0.6	3	34	764	1.8	4	45	935	2.0	0.186	0.762	3.151
r112	2	21	515	0.5	3	34	758	1.1	4	44	939	3.1	0.151	0.744	3.776
									Average gap percentage				5.656	56.291	375.961

of destinations, the number of routes, time budget and operating hours determine the difficulties in arranging the instances.

Acknowledgements The author would like to thank United States Agency for International Development (USAID) for the training and mentoring support in writing this article, through the Sustainable Higher Education Research Alliance (SHERA) Program for Universitas Indonesia's Scientific Modeling, Application, Research and Training for City-centered Innovation and Technology (SMART CITY) Project.

References

Bansal N, Blum A, Chawla S, Meyerson A (2004) Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In: Proceedings of the thirty-sixth annual ACM symposium on theory of computing - STOC'04. https://doi.org/10.1145/1007352.1007385

Bederina H, Hifi M (2017) A hybrid multi-objective evolutionary algorithm for the team orienteering problem. In: 2017 4th international conference on control, decision and information technologies (CoDIT). https://doi.org/10.1109/codit.2017.8102710

Chekuri C, Kumar A (2004) Maximum coverage problem with group budget constraints and applications. Lecture Notes in Computer Science, pp 72–83. https://doi.org/10.1007/978-3-540-27821-4_7

Chekuri C, Pal M (2005) A recursive Greedy algorithm for walks in directed graphs. In: 46th annual IEEE symposium on foundations of computer science (FOCS'05). https://doi.org/10.1109/sfcs.2005.9

Cura T (2014) An artificial bee colony algorithm approach for the team orienteering problem with time windows. Comput Ind Eng 74:270–290. https://doi.org/10.1016/j.cie.2014.06.004

Gavalas D, Kasapakis V, Konstantopoulos C, Mastakas K, Pantziou G (2013a) A survey on mobile tourism recommender systems. In: 2013 third international conference on communications and information technology (ICCIT). https://doi.org/10.1109/iccitechnology.2013.6579536

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G, Tasoulas Y (2013b) Cluster-based heuristics for the team orienteering problem with time windows. Lecture Notes in Computer Science, pp 390–401. https://doi.org/10.1007/978-3-642-38527-8_34

Gendreau M, Lourenço HR (2003) Handbook of metaheuristics. Springer

Gendreau M, Laporte G, Semet F (1998a) A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks 32:263–273. https://doi.org/10.1002/(sici)1097-0037(199812)32:4<263::aid-net3>3.0.co;2-q

Gendreau M, Laporte G, Semet F (1998b) A tabu search heuristic for the undirected selective travelling salesman problem. Eur J Oper Res 106(2–3):539–545. https://doi.org/10.1016/s0377-2217(97)00289-0

Gunawan A, Chuin H, Kun L (2015a) SAILS: hybrid algorithm for the team orienteering problem with time windows. In: Proceedings 7th multidisciplinary international scheduling conference, pp 276–295

Gunawan A, Lau HC, Lu K (2015b) Well-tuned ILS for extended team orienteering problem with time windows. LARC Technical Report Series: http://smu.edu/sg/centres/larc/larc-technical-reports-series

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332. https://doi.org/10.1016/j.ejor.2016.04.059

Herzog D, Wörndl W (2014) A travel recommender system for combining multiple travel regions to a composite trip. Content-Based Recomm Syst 1245:42–47

Hu Q, Lim A (2014) An iterative three-component heuristic for the Team Orienteering Problem with Time Windows. Eur J Oper Res

Hu W, Fathi M, Pardalos PM (2018) A multi-objective evolutionary algorithm based on decomposition and constraint programming for the multi-objective team orienteering problem with time windows. Appl Soft Comput 73:383–393. https://doi.org/10.1016/j.asoc.2018.08.026

Indonesian Ministry of Tourism (2017) 2016 Tourism ministry performance accountability report

Kantor MG, Rosenwein MB (1992) The orienteering problem with time windows. J Oper Res Soc 43(6):629–635. https://doi.org/10.1057/jors.1992.88

Labadie N, Melechovský J, Wolfler Calvo R (2010) Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. J Heuristics 17(6):729–753. https://doi.org/10.1007/s10732-010-9153-z

Li J, Fu P (2012) A label correcting algorithm for dynamic tourist trip planning. J Softw 7(12):2899–2905. https://doi.org/10.4304/jsw.7.12.2899-2905

Lin S-W, Yu VF (2012) A simulated annealing heuristic for the team orienteering problem with time windows. Eur J Oper Res 217(1):94–107. https://doi.org/10.1016/j.ejor.2011.08.024

Lin S-W, Yu VF (2017) Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing. Comput Ind Eng 114:195–205. https://doi.org/10.1016/j.cie.2017.10.020

Lourenço HR, Martin OC, Stützle T (2013) Iterated local search. In: Hillier FS, Price CC (eds) International series in operations research and management science. Springer, Berlin, pp 320–353. https://doi.org/10.1007/0-306-48056-5_11

Martín-Moreno R, Vega-Rodríguez MA (2018) Multi-objective artificial bee colony algorithm applied to the bi-objective orienteering problem. Knowl-Based Syst 154:93–101. https://doi.org/10.1016/j.knosys.2018.05.005

Mei Y, Salim FD, Li X (2016) Efficient meta-heuristics for the multi-objective time-dependent orienteering problem. Eur J Oper Res 254(2):443–457. https://doi.org/10.1016/j.ejor.2016.03.053

Mirzaei MH, Ziarati K, Naghibi M-T (2017) Bi-objective version of team orienteering problem (BTOP). In: 2017 7th international conference on computer and knowledge engineering (ICCKE). https://doi.org/10.1109/iccke.2017.8167930

Montemanni R, Gambardella LM (2009) An ant colony system for team orienteering problems with time windows. FoundComput Decis Sci 34(4):287

Rezki H, Aghezzaf B (2017) The bi-objective orienteering problem with budget constraint: GRASP_ILS. In: 2017 international colloquium on logistics and supply chain management (LOGISTIQUA). https://doi.org/10.1109/logistiqua.2017.7962868

Righini G, Salani M (2009) Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Comput Oper Res 36(4):1191–1203. https://doi.org/10.1016/j.cor.2008.01.003

Souffriau W, Vansteenwegen P, Vertommen J, Berghe GV, Oudheusden DV (2008) A personalized tourist trip design algorithm for mobile tourist guides. Appl Artif Intell 22(10):964–985

Schilde M, Doerner KF, Hartl RF, Kiechle G (2009) Metaheuristics for the bi-objective orienteering problem. Swarm Intell 3(3):179–201. https://doi.org/10.1007/s11721-009-0029-5

Talbi E-G (2009) Metaheuristics from design to implementation. http://onlinelibrary.wiley.com/book/10.1002/9780470496916

Tricoire F, Romauch M, Doerner KF, Hartl RF (2010) Heuristics for the multi-period orienteering problem with multiple time windows. Comput Oper Res 37(2):351–367. https://doi.org/10.1016/j.cor.2009.05.012

Tsiligirides T (1984) Heuristic methods applied to orienteering. J Oper Res Soc 35(9):797. https://doi.org/10.2307/2582629

Tsitsiklis JN (1992) Special cases of traveling salesman and repairman problems with time windows. Networks 22(3):263–282. https://doi.org/10.1002/net.3230220305

Vansteenwegen P (2008) Planning in tourism and public transportation. 4OR 7(3):293–296. https://doi.org/10.1007/s10288-008-0086-4

Vansteenwegen P, Souffriau W, Berghe GV, Oudheusden DV (2009a) A guided local search metaheuristic for the team orienteering problem. Eur J Oper Res 196(1):118–127. https://doi.org/10.1016/j.ejor.2008.02.037

Vansteenwegen P, Souffriau W, Berghe Vanden G V, Oudheusden DV (2009b) Iterated local search for the team orienteering problem with time windows. Comput Oper Res 36(12):3281–3290. https://doi.org/10.1016/j.cor.2009.03.008

Vansteenwegen P, Souffriau W, Sörensen K (2009c) The mobile mapping van problem: a matheuristic for capacitated arc routing with soft time windows and depot selection. In: Proceedings 13th information control problem in manufacturing, pp 1119–1124

Vansteenwegen P, Souffriau W, Berghe GV, Oudheusden DV (2011a) The city trip planner: an expert system for tourists. Expert Syst Appl 38(6):6540–6546. https://doi.org/10.1016/j.eswa.2010.11.085

Vansteenwegen P, Souffriau W, Oudheusden DV (2011b) The orienteering problem: a survey. Eur J Oper Res 209(1):1–10. https://doi.org/10.1016/j.ejor.2010.03.045

Journal of Industrial Engineering International

JIEI
Journal of Industrial Engineering International
(Indexed in Scopus)
Islamic Azad University
South Tehran Branch

Articles in Press
Current Issue

Journal Archive

Journal of Industrial Engineering International; JIEI was launched in 2003. The journal aims to introduce new achievements in theoretical studies as a well practical approach to solve industrial engineering problems. To accomplish its goals, the journal has been publishing quarterly about 10 full-text articles per each issue in English since 2005. JIEI rapidly obtained the scientific-research journal score by the Iranian Ministry of Science, Research and Technology. From 2012 to 2019, this journal was published by **Springer** and also, ranked as a Q_1 journal by Scopus in 2018.

OPEN ACCESS

JIEI is aimed at an audience of researchers, educators and practitioners of industrial engineering and associated fields. It publishes original contributions on the development of methodologies for solving industrial engineering disciplines problems after the peer-review process. The journal encourages submissions that expand the frontiers of the fundamental theories and concepts underlying industrial engineering techniques.

Most Visited Articles

Publication Information

Publisher
Islamic Azad University, South Tehran Branch

Print ISSN 1735-5702
Online ISSN 2251-712X


Journal of
Industrial
Engineering
International

Journal of Industrial Engineering International

Journal of Industrial Engineering International is now archived and no longer receiving submissions with this publisher. All articles published in the journal during its time with Springer will remain fully searchable through our websites. Looking for other Springer journals? Please have a look at our journal list.

Search within journal

Volumes and issues

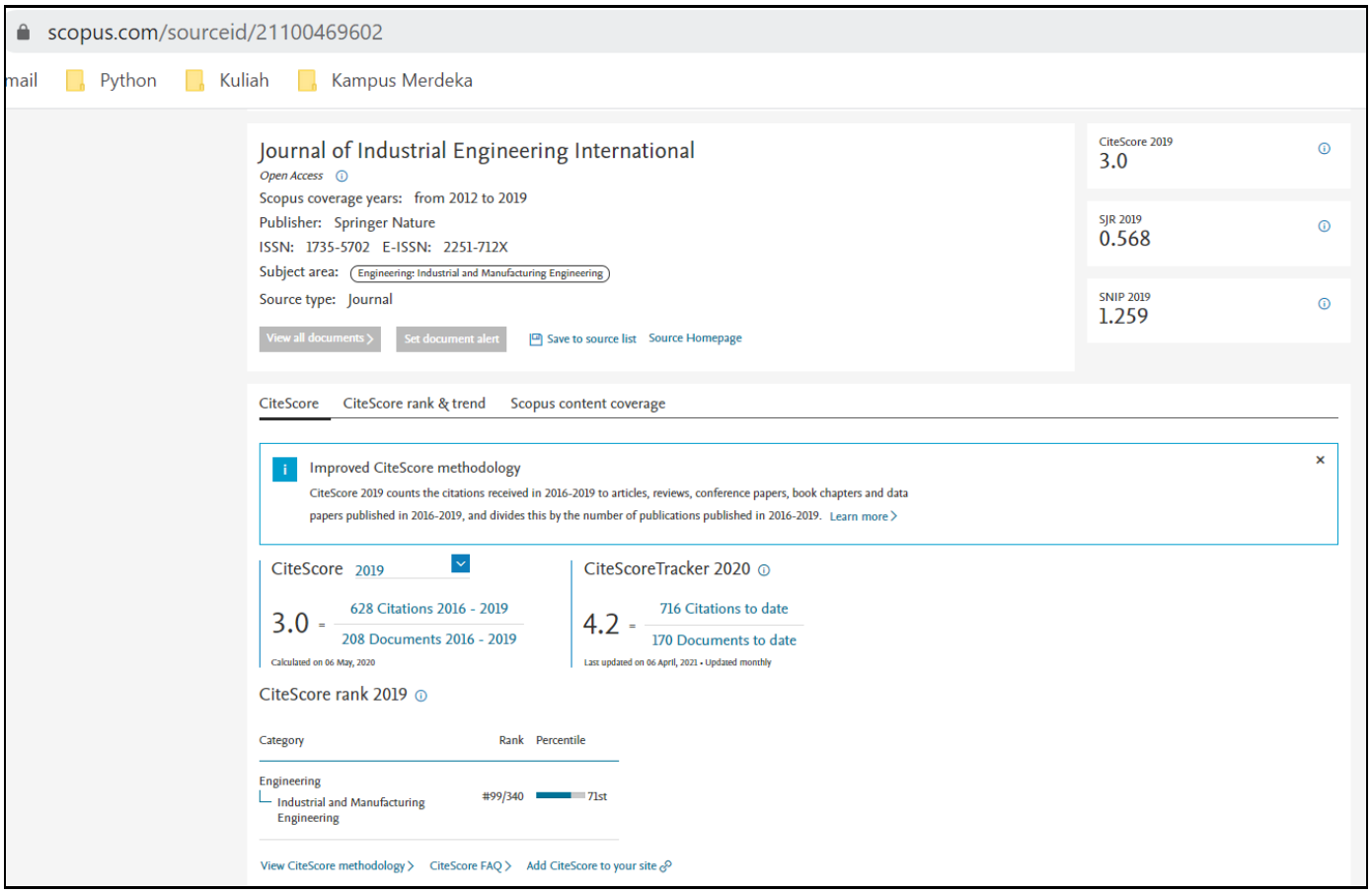
Volume 15

March - December 2019

Schimago


scimagojr.com/journalsearch.php?q=21100469602&tip=sid&clean=0
Update
Apps
Gmail
Python
Kuliah
Kampus Merdeka
Other bookmarks
Reading
4
2
0
2012 2013 2014 2015 2016 2017 2018 2019
22.5
15
7.5
0
2012 2013 2014 2015 2016 2017 2018 2019
0.6
0
2012 2013 2014 2015 2016 2017 2018 2019
Cites / Doc. (4 years)
Cites / Doc. (3 years)
Cites / Doc. (2 years)
Citable documents
Non-citable documents
140
70
0
2012 2013 2014 2015 2016 2017 2018 2019
Cited documents
Uncited documents
140
70
0
2012 2013 2014 2015 2016 2017 2018 2019
Journal of Industrial Engineering International
Q2
Industrial and Manufacturing Engineering
best quartile
SJR 2019
0.57
powered by scimagojr.com
← Show this widget in your own website
Just copy the code below and paste within your html code:
<a href="https://www.scimag

SCOPUS


scopus.com/sourceid/21100469602
mail
Python
Kuliah
Kampus Merdeka
Journal of Industrial Engineering International
Open Access
Scopus coverage years: from 2012 to 2019
Publisher: Springer Nature
ISSN: 1735-5702 E-ISSN: 2251-712X
Subject area: Engineering: Industrial and Manufacturing Engineering
Source type: Journal
View all documents >
Set document alert
Save to source list
Source Homepage
CiteScore 2019
3.0
SJR 2019
0.568
SNIP 2019
1.259
CiteScore
CiteScore rank & trend
Scopus content coverage
Improved CiteScore methodology
CiteScore 2019 counts the citations received in 2016-2019 to articles, reviews, conference papers, book chapters and data papers published in 2016-2019, and divides this by the number of publications published in 2016-2019. Learn more >
CiteScore 2019
3.0 = 628 Citations 2016 - 2019 / 208 Documents 2016 - 2019
Calculated on 06 May, 2020
CiteScoreTracker 2020
4.2 = 716 Citations to date / 170 Documents to date
Last updated on 06 April, 2021 • Updated monthly
CiteScore rank 2019
Category
Rank
Percentile
Engineering
Industrial and Manufacturing Engineering
#99/340
71st
View CiteScore methodology >
CiteScore FAQ >
Add CiteScore to your site