Efficient Meta-heuristics for the Multi-Objective Time-Dependent Orienteering Problem

Yi Mei[a,*], Flora Salim[b], Xiaodong Li[b]

[a]*School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6012, New Zealand.*
[b]*School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3000, Australia.*

Abstract

In this paper, the Multi-Objective Time-Dependent Orienteering Problem (MOTDOP) is investigated. Time-dependent travel time and multiple preferences are two of the most important factors in practice, and have been handled separately in previous work. However, no attempts have been made so far to consider these two factors together. Handling both multiple preferences and time-dependent travel time simultaneously poses a challenging optimization task in this NP-hard problem. In this study, two meta-heuristic methods are proposed for solving MOTDOP: a Multi-Objective Memetic Algorithm (MOMA) and a Multi-objective Ant Colony System (MACS). Two sets of benchmark instances were generated to evaluate the proposed algorithms. The experimental studies show that both MOMA and MACS managed to find better solutions than an existing multi-objective evolutionary algorithm (FMOEA). Additionally, MOMA achieved better performance than MACS in a shorter time, and is less sensitive to the parameter setting. Given that MACS inherits promising features of P-ACO, which is a state-of-the-art algorithm for multi-objective orienteering problem, the advantage of MOMA over MACS and FMOEA demonstrates the efficacy of adopting the memetic algorithm framework to solve MOTDOP.

Keywords: Orienteering problem, multi-objective optimization,

*Corresponding author.
[1]Email address: yi.mei@ecs.vuw.ac.nz
[2]Email addresses: flora.salim@rmit.edu.au (Flora Salim), xiaodong.li@rmit.edu.au (Xiaodong Li).

meta-heuristics, memetic algorithm, ant colony system

1. Introduction

The orienteering problem is a significant problem that can be found in real-world application areas such as the home fuel delivery [1] and tourist trip design [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Given a set of Points-Of-Interest (POIs), including the starting and ending POI, each POI has a score that represents its desirability. The orienteering problem aims to design a tour leaving from the starting POI, visiting a subset of POIs and finally arriving at the ending POI. The objective is to maximize the total score of the visited POIs while the total travel time of the route does not exceed the predefined time budget.

In practice, the travel time between the POIs is time-dependent. For example, the peak-hour travel time should be longer than the travel time during the off-peak period. To better simulate the reality, the orienteering problem with time-dependent travel cost is considered in this paper. In addition, each POI may have multiple scores, each representing its desirability from a different perspective. For example, a POI may have a high score as an art gallery, but a low score as an architecture. Given multiple scores, one of the challenges is to obtain a set of Pareto-optimal solutions instead of a single global optimal solution.

The orienteering problem has been extensively investigated in literature. Golden et al. [1] proved that even the most basic version of the orienteering problem is NP-hard. Thus, no polynomial algorithm exists for solving the problem to optimality. There have been a number of exact methods and heuristics proposed for solving orienteering problems, such as the branch-and-bound approach [12, 13] and various heuristics (e.g., [14, 15, 16, 17]). There have been also investigations on various extension of orienteering problems that are closer to reality, such as the team orienteering problem [18, 19, 20, 21, 22, 23], the orienteering problem with time windows [24, 25, 26, 27, 28, 29], the multi-objective orienteering problem [30, 31, 5, 10], and the time-dependent orienteering problem [32, 33, 4, 34, 29, 35, 36, 11]. A more comprehensive survey can be found in [37]. However, the two aspects of the problem in this study, i.e., multiple objectives and time-dependent cost (travel time), have not been tackled together by any algorithm so far. This paper attempts to solve the Multi-Objective Time-Dependent Orienteering Problem (MOTDOP) for the first time.

Handling multiple objectives and time-dependent cost simultaneously poses a challenging optimization task in MOTDOP, which makes it more difficult than considering the multiple objectives or time-dependent cost alone. First, due to the conflict between objectives, it is necessary to explore the search space more thoroughly, which thus leads to a much higher demand in computational resource. Second, due to the time-dependent cost, even a minor change on the solution may lead to a severe difference on the total cost, making it very difficult to identify promising regions. To tackle these challenges, two meta-heuristic algorithms are designed for solving MOTDOP. The first one is the Multi-Objective Memetic Algorithm (MOMA), which has been demonstrated to be a competitive approach for graph-based optimization problems (e.g. shortest path finding) [38, 39, 40, 41]. To solve the MOTDOP with MOMA, problem-specific crossover and local search operators are designed to improve the search efficiency and effectiveness in handling the time-dependent travel time. The second one is the Multi-objective Ant Colony System (MACS). It is a well-known algorithm for solving graph-based optimization problems, including the single-objective time-dependent orienteering problem [35] and the multi-objective static orienteering problem [30], which are the two closest models to MOTDOP. MACS adopts most components of the Pareto Ant Colony Optimization (P-ACO) [30], whose efficacy has been demonstrated on the multi-objective static orienteering problem. The major difference between MACS and P-ACO is the iterative improvement (local search) procedure, where MACS takes the time-dependent travel time into account.

Since MOTDOP has not been studied before, there is no existing benchmark instance for testing the algorithms. Therefore, we generated two sets of benchmarks to facilitate experimental studies. The first set is extended from the single-objective time-dependent instances used in [35] by randomly adding an additional objective. The second one is extended from the multi-objective static instances used in [30] by applying the time-dependency model in [35] to transform each static travel time into a time-dependent one. Then, MOMA and MACS were evaluated on the newly generated benchmark instances, and compared with an existing algorithm called FMOEA [10]. The experimental results show that both algorithms find better sets of solutions than FMOEA. In addition, MOMA performs much better than MACS in terms of both solution quality and speed. Since ant colony system has been successfully applied to solving orienteering problems, the outperformance of MOMA over MACS suggests a great potential to use the memetic algorithm

framework in solving orienteering problems.

The rest of the paper is organized as follows: Section 2 gives the mathematical formulation of MOTDOP. Then, the proposed MOMA and MACS are described in Section 3. The experimental studies are carried out in Section 4 to evaluate MOMA and MACS. Finally, the conclusion and future work are given in Section 5.

2. Multi-Objective Time-Dependent Orienteering Problem

In MOTDOP, there is a set of vertices $V = \{v_1, \ldots, v_n\}$ located on a planar graph, representing the set of POIs on the map. The tourist is assumed to depart from v_1 (the *source node*) and arrive at v_n (the *target node*). Each vertex v_i has a score list $\mathbf{s}_i = (s_{i1}, \ldots, s_{im})$, indicating its degrees of desirability of the tourist from different perspectives. For example, a museum can fall into the categories of art gallery and architecture. Then, it should have two desirability scores, one as an art gallery, and the other as an architecture. The time-dependent matrix of travel time is denoted as $\mathbf{D}(t) = (d_{ij}(t))_{n\times n}$, where $d_{ij}(t)$ stands for the travel time from v_i to v_j at time t. Given a starting time t_0 and the time budget T, the task is to design a tour that starts from v_1 at t_0, visits a subset of vertices and finally returns v_n no later than $t_0 + T$, so that the total scores of the visited vertices are maximized.

The mathematical model of MOTDOP can be stated as follows:

$$\max \ f_k = \sum_{i=2}^{n-1} s_{ik} \left(\sum_{j=1}^{n} x_{ij} \right), \quad k = 1, \ldots, m, \tag{1}$$

$$\text{s.t.:} \ \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1, \tag{2}$$

$$\sum_{j=2}^{n} x_{j1} = \sum_{i=1}^{n-1} x_{ni} = 0, \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ji} \leq 1, \quad i = 2, \ldots, n-1, \tag{4}$$

$$y_1 = t_0, \tag{5}$$

$$y_n \leq t_0 + T, \tag{6}$$

$$y_j = \sum_{i=1}^{n} (y_i + d_{ij}(y_i)) \cdot x_{ij}, \quad j = 2, \ldots, n, \tag{7}$$

$$x_{ij} \in \{0, 1\}, \tag{8}$$

where the decision variables include x_{ij} and y_i $(i, j = 1, \ldots, n)$. Specifically, $x_{ij} = 1$ if there is an arc from v_i to v_j in the tour, and 0 otherwise. y_i stands for the departure (arrival) time of v_i in the tour.

Eq. (1) maximizes the m total scores of the solution in terms of the score list. Eq. (2) indicates that there is exactly one arc leaving the source node and entering the target node. Eq. (3) means that there is no arc leaving the target node or entering the source node. Eq. (4) guarantees that there is at most one arc entering and leaving each intermediate node, and the in-degree equals the out-degree for each intermediate node. Eqs. (5) and (6) imply that the tour starts from the source node at t_0, and arrives at the target node no later than $t_0 + T$. Eq. (7) calculates the arrival time for each intermediate node by its predecessor in the tour. When calculating the arrival time of v_j, it only counts a single term $y_i + d_{ij}(y_i)$, where $x_{ij} = 1$, i.e., v_i is the predecessor of v_j in the tour. Note that if v_j is not selected in the tour, then $y_j = 0$, without violating the optimality of the model. Eq. (8) indicates that the decision variables are binary. Due to Eq. (7), the above model is a non-linear mixed integer programming model with multiple objectives.

In Eq. (1), there are multiple scores to be maximized, which are usually in conflict with each other. Therefore, in MOTDOP, there may be multiple optimal solutions with different trade-offs among the scores, which are called the *Pareto-optimal solutions*. In multi-objective optimization, solution a is said to *dominate* solution b, if all the objective values of a are no worse than that of b, and there is at least one objective for which a has a better value than b. Then, a solution is called to be a Pareto-optimal solution, if it is dominated by no other solution in the entire solution space.

Fig. 1 shows an example of a solution of MOTDOP. The diamonds indicate the source and destination, and the circle points stand for the intermediate POIs, each of which is associated with two scores. The illustrated tour leaves the source node at 9:00, and arrives at the destination at 16:50. In order to finish the tour before the deadline (17:00), three POIs are skipped. The total score list of the tour is $(2, 4)+(3, 2)+(4, 2)+(3, 3)+(3, 4) = (15, 15)$.

It should be noted that the visiting time seems to be omitted in our work. However, the visiting time can be easily included in the model by defining the travel time between any two POIs as the sum of the commuting time (actual travel time) and the visiting time of the target POI. For example, in Fig. 1, assuming that the tourist departs from the source at 9:00AM and

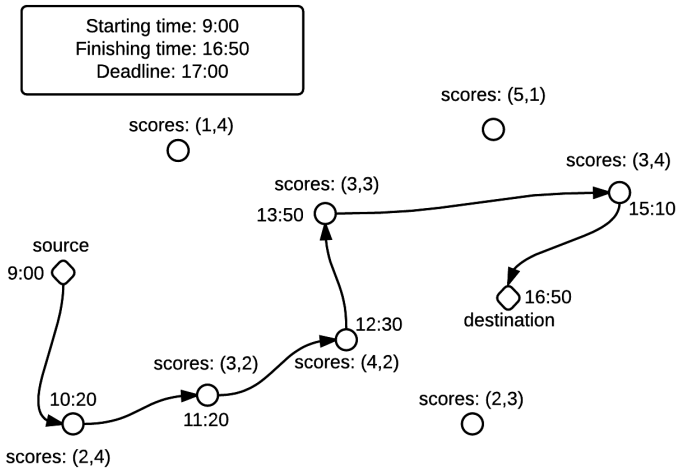


Figure 1: An example of a solution of the multi-objective time-dependent orienteering problem.

arrives at the first POI at 9:50AM, and stays half an hour to enjoy it. Then the corresponding travel time defined in Fig. 1 is 1 hour 20 minutes (50 minutes for commuting plus 30 minutes for visiting). This way, the visiting time can be directly addressed by the current model.

3. Meta-heuristics Designed

To facilitate the algorithm description, we first give the basic notations that will be used in both MOMA and MACS in Table 1. In the table, a solution Γ is represented as a sequence of POIs, which starts from v_1 and ends at v_n.

3.1. Multi-Objective Memetic Algorithm

Memetic algorithm [42] can be seen as a special case of the well-known evolutionary algorithm, in which the traditional mutation operator is replaced by the local search process. Due to the self-learning scheme brought by the local search, memetic algorithm particularly performs well in solving combinatorial optimization problems [43, 44, 45], in which the fitness landscape is rugged and the definition of solution neighborhood is untrivial because of the complicated solution structure such as permutation.

Table 1: The notations used in algorithm description.

Notation	Description
v_i	The i^{th} vertex (POI) of the graph
$\delta(v_i, v_j)$	Euclidean distance between v_i and v_j
Γ	A solution to the problem
$\mathbf{\Gamma}$	A population (set of solutions)
Γ_l	the l^{th} ant or individual of $\mathbf{\Gamma}$
$t_{\text{arr}}(\Gamma)$	the arrival time of the destination v_n in Γ
$\Gamma(i)$	the i^{th} vertex in the tour of Γ
f_k	The k^{th} objective value defined by Eq. (1) ($k = 1, \ldots, m$)

As mentioned in [38], when applying memetic algorithm to multi-objective optimization problems, there are four important issues to be addressed: (1) fitness assignment; (2) diversity preservation; (3) elitism and (4) solution evaluation during local search. When considering all the above four issues in the context of combinatorial optimization, a decomposition-based memetic algorithm was proposed in [38] for solving the multi-objective capacitated arc routing problem. Due to its efficacy, the framework of this decomposition-based memetic algorithm is employed for solving MOTDOP as well. The pseudo-code of the Multi-Objective Memetic Algorithm (MOMA) is described in Algorithm 1.

At first, a population with *popsize* individuals is initialized by the function `initialize()`. Then, in each generation, an offspring population is generated by `crossover()` and `localSearch()`, and combined with the current population. Then, all the individuals in the combined population are sorted by the fast non-dominated sorting procedure, and the best *popsize* individuals are selected to go to the next generation.

The `initialize()` function is described in Algorithm 2. Starting from a tour directly from the source to the destination, all the intermediate POIs are randomly shuffled and inserted behind v_1 one by one, until no more POI can be inserted. The random shuffle enables the initialization to generate different initial individuals by allowing the vertices with larger indices to be inserted before those with smaller indices.

Then, the fast non-dominated sorting procedure [46] `nondominatedSort()` is applied to $\mathbf{\Gamma}$ to sort the individuals from best to worst in terms of the dominance relation and crowding distance. The brief idea of the fast non-dominated sorting is to rank the individuals based on the dominance relation, so that the individuals with a higher rank are always dominated by at least one individual with a lower rank, and the individuals with the same rank

Algorithm 1: The multi-objective memetic algorithm

Input: *popsize*
Output: A set of non-dominated solutions $\mathbf{\Gamma}^{\text{nd}}$
// Initialize the population
$\mathbf{\Gamma} \leftarrow \emptyset$, $\mathbf{\Gamma}^{\text{nd}} \leftarrow \emptyset$;
for $i = 1 \rightarrow$ *popsize* **do**
    $\Gamma_i \leftarrow$ `initialize()`; *// initialize a feasible solution*
    $\mathbf{\Gamma} \leftarrow \mathbf{\Gamma} \cup \Gamma_i$, update $\mathbf{\Gamma}^{\text{nd}}$ with Γ_i;
end
$\mathbf{\Gamma} \leftarrow$ `nondominatedSort`($\mathbf{\Gamma}$);
// Search process
while *Stopping criteria not met* **do**
    // Generate the offspring population
    $\mathbf{\Gamma}^{\text{off}} \leftarrow \emptyset$; *// the offspring population*
    for $i = 1 \rightarrow$ *popsize*/2 **do**
        $(\Gamma_1^{\text{par}}, \Gamma_2^{\text{par}}) \leftarrow$ `parentSelection`($\mathbf{\Gamma}$);
        $(\Gamma_1^{\text{off}}, \Gamma_2^{\text{off}}) \leftarrow$ `crossover`($\Gamma_1^{\text{par}}, \Gamma_2^{\text{par}}$);
        $\Gamma_1^{\text{off}} \leftarrow$ `localSearch`(Γ_1^{off});
        $\Gamma_2^{\text{off}} \leftarrow$ `localSearch`(Γ_2^{off});
        $\mathbf{\Gamma}^{\text{off}} \leftarrow \mathbf{\Gamma}^{\text{off}} \cup \{\Gamma_1^{\text{off}}, \Gamma_2^{\text{off}}\}$;
    end
    $\mathbf{\Gamma}^{\text{comb}} \leftarrow \mathbf{\Gamma} \cup \mathbf{\Gamma}^{\text{off}}$; *// the combination of the two populations*
    $\mathbf{\Gamma}^{\text{comb}} \leftarrow$ `nondominatedSort`($\mathbf{\Gamma}^{\text{comb}}$);
    $\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}^{\text{comb}}(1 : \textit{popsize})$;
    Update $\mathbf{\Gamma}^{\text{nd}}$ with $\mathbf{\Gamma}$;
end
return $\mathbf{\Gamma}^{\text{nd}}$;

Algorithm 2: The solution initialization

$\Gamma \leftarrow (v_1, v_n)$, $\Omega \leftarrow V \setminus \{v_1, v_n\}$;
Randomly shuffle the vertices in Ω;
foreach $v \in \Omega$ **do**
    $\Gamma' \leftarrow \Gamma$;
    Insert v behind v_1 in Γ';
    if $t_{arr}(\Gamma') \leq t_0 + T$ **then** $\Gamma \leftarrow \Gamma'$;
end
return Γ;

are non-dominated to each other. Then, within the same front (the set of individuals with the same rank), a crowding distance is calculated for each individual to reflect the crowdedness around it. The individuals with larger crowding distance values are considered to be in less crowded areas and thus have better fitness values. The pseudo-code of the non-dominated sorting is given in Algorithm 3, where $n_{\text{dom}}(\Gamma)$ and $\mathbf{\Gamma}_{\text{dom}}(\Gamma)$ stand for the number of individuals dominating Γ, and the set of individuals that Γ dominates, respectively. In Line 19 of Algorithm 3, for all the individuals in the front

Algorithm 3: The fast non-dominated sorting procedure

Input: A population $\mathbf{\Gamma}$
Output: A sorted population $\mathbf{\Gamma}^{\text{sort}}$

$\mathbf{\Gamma}^{\text{sort}} \leftarrow \emptyset$;
foreach $\Gamma \in \mathbf{\Gamma}$ **do** $n_{\text{dom}}(\Gamma) \leftarrow 0$, $\mathbf{\Gamma}_{\text{dom}}(\Gamma) \leftarrow \emptyset$;
foreach $(\Gamma_1, \Gamma_2) \in \mathbf{\Gamma} \times \mathbf{\Gamma}$ *and* $\Gamma_1 \neq \Gamma_2$ **do**
    if Γ_1 *dominates* Γ_2 **then**
        $n_{\text{dom}}(\Gamma_2) \leftarrow n_{\text{dom}}(\Gamma_2) + 1$, $\mathbf{\Gamma}_{\text{dom}}(\Gamma_1) \leftarrow \mathbf{\Gamma}_{\text{dom}}(\Gamma_1) \cup \Gamma_2$;
    else if Γ_2 *dominates* Γ_1 **then**
        $n_{\text{dom}}(\Gamma_1) \leftarrow n_{\text{dom}}(\Gamma_1) + 1$, $\mathbf{\Gamma}_{\text{dom}}(\Gamma_2) \leftarrow \mathbf{\Gamma}_{\text{dom}}(\Gamma_2) \cup \Gamma_1$;
    end
end
// Obtain the fronts
$rank \leftarrow 0$; *// initial rank*
repeat
    $\mathbf{\Gamma}^{\text{front}} \leftarrow \emptyset$;
    foreach $\Gamma \in \mathbf{\Gamma}$ **do**
        if $n_{dom}(\Gamma) = 0$ **then**
            $rank(\Gamma) \leftarrow rank$, $\mathbf{\Gamma}^{\text{front}} \leftarrow \mathbf{\Gamma}^{\text{front}} \cup \Gamma$;
            foreach $\Gamma' \in \mathbf{\Gamma}_{dom}(\Gamma)$ **do** $n_{\text{dom}}(\Gamma') \leftarrow n_{\text{dom}}(\Gamma') - 1$;
        end
    end
    // Sort the front by crowding distance
    $d_{\text{crowd}}(\mathbf{\Gamma}^{\text{front}}) \leftarrow$ `crowdingDistance`$(\mathbf{\Gamma}^{\text{front}})$;
    Sort $\mathbf{\Gamma}^{\text{front}}$ in the decreasing order of $d_{\text{crowd}}(\cdot)$;
    $\mathbf{\Gamma}^{\text{sort}} \leftarrow \mathbf{\Gamma}^{\text{sort}} \cup \mathbf{\Gamma}^{\text{front}}$, $\mathbf{\Gamma} \leftarrow \mathbf{\Gamma} \setminus \mathbf{\Gamma}^{\text{front}}$;
    $rank \leftarrow rank + 1$; *// rank increment*
until $\mathbf{\Gamma} = \emptyset$;
return $\mathbf{\Gamma}^{\text{sort}}$;

Algorithm 4: Crowding distance calculation

foreach $\Gamma \in \mathbf{\Gamma}$ **do** $d_{\text{crowd}}(\Gamma) \leftarrow 0$;
for $k = 1 \rightarrow m$ **do**
    Sort $\mathbf{\Gamma}$ in the increasing order of $f_k(\cdot)$;
    $d_1 \leftarrow \infty$, $d_n \leftarrow \infty$; *// n is the size of $\mathbf{\Gamma}$*
    for $i = 2 \rightarrow n - 1$ **do**
        $d_i \leftarrow \frac{f_k(\Gamma_{i+1}) - f_k(\Gamma_{i-1})}{f_k(\Gamma_n) - f_k(\Gamma_1)}$; *// Γ_i is the i^{th} individual in $\mathbf{\Gamma}$*
    end
    for $i = 1 \rightarrow n$ **do** $d_{\text{crowd}}(\Gamma_i) \leftarrow d_{\text{crowd}}(\Gamma_i) + d_i$;
end
return $d_{\text{crowd}}(\cdot)$;

$\mathbf{\Gamma}^{\text{front}}$, the calculation of the crowding distance is described in Algorithm 4.

In each generation, two parent individuals are first selected by the function `parentSelection`$(\mathbf{\Gamma})$. Here, the multi-objective binary tournament selection is adopted. When selecting each parent, two individuals are randomly picked, and the better one is selected to be the parent. After conducting `nondominatedSort()` to the population $\mathbf{\Gamma}$, an individual Γ_1 is considered to

Algorithm 5: Multi-objective binary tournament parent selection

Input: A population $\mathbf{\Gamma}$
Output: Two parents $(\Gamma_1^{\text{par}}, \Gamma_2^{\text{par}})$
// Generate the first parent
Randomly select two individuals Γ_1 and Γ_2 from $\mathbf{\Gamma}$;
if Γ_1 *is before* Γ_2 *in* $\mathbf{\Gamma}$ **then** $\Gamma_1^{\text{par}} \leftarrow \Gamma_1$;
else $\Gamma_1^{\text{par}} \leftarrow \Gamma_2$;
// Generate the second parent
repeat
    Randomly select two individuals Γ_3 and Γ_4 from $\mathbf{\Gamma}$;
    if Γ_3 *is before* Γ_4 *in* $\mathbf{\Gamma}$ **then** $\Gamma_2^{\text{par}} \leftarrow \Gamma_3$;
    else $\Gamma_2^{\text{par}} \leftarrow \Gamma_4$;
until $\Gamma_2^{par} \neq \Gamma_1^{par}$;
return $(\Gamma_1^{\text{par}}, \Gamma_2^{\text{par}})$;

be better than another individual Γ_2 if Γ_1 is before Γ_2 in $\mathbf{\Gamma}$. The multi-objective binary tournament selection is given in Algorithm 5.

The single-point crossover operator is applied to the parent individuals Γ_1^{par} and Γ_2^{par} to generate two offsprings Γ_1^{off} and Γ_2^{off}. The single-point crossover operator is described in Algorithm 6. At first, two cutting points r_1 and r_2 are randomly sampled for Γ_1^{par} and Γ_2^{par}. Then, the two offsprings Γ_1^{off} and Γ_2^{off} are generated by swapping two sub-tours $\Gamma_1^{\text{par}}(r_1 + 1 : l_1)$ and $\Gamma_2^{\text{par}}(r_2 + 1 : l_2)$. Starting from the tour (v_1, v_n), the offspring Γ_1^{off} inserts the elements from position 2 to r_1 of Γ_1^{par}, and then inserts the elements from position r_2 to $l_2 - 1$ of Γ_2^{par}. When inserting each element, the feasibility of the insertion is checked, and the insertion is only implemented if it is feasible. The same procedure is conducted when generating Γ_2^{off}.

The generated offsprings Γ_1^{off} and Γ_2^{off} then undergo a local search process, which is described in Algorithm 7. First, the unvisited vertices are randomly shuffled. Then, the vertices are inserted to the best feasible position which leads to the minimal additional travel time one by one. If no feasible position is found for a vertex, it is simply skipped.

3.2. Multi-objective Ant Colony System

Ant colony optimization was proposed by Dorigo [47], which was originally to find the shortest path by mimicking the behavior of the ant colony during the foraging process. Ant colony optimization has been demonstrated to be efficient to solve shortest-path-finding problems such as traveling salesman problem [48] due to the similarity between their underlying models. Generally speaking, ant colony optimization is an iterative search method where at each iteration, a set of solutions (ants) are generated based on

Algorithm 6: The single-point crossover

Input: Two parents $(\Gamma_1^{\text{par}}, \Gamma_2^{\text{par}})$
Output: Two offsprings $(\Gamma_1^{\text{off}}, \Gamma_2^{\text{off}})$
Randomly sample $r_1 \in \{2, \ldots, l_1 - 1\}$; // l_1 *is the length of* Γ_1^{par}
Randomly sample $r_2 \in \{2, \ldots, l_2 - 1\}$; // l_2 *is the length of* Γ_2^{par}
$\Gamma_1^{\text{off}} \leftarrow (v_1, v_n)$, $\Gamma_2^{\text{off}} \leftarrow (v_1, v_n)$;
// Generate the first offspring
for $i = 2 \rightarrow r_1$ **do**
    $\Gamma' \leftarrow \Gamma_1^{\text{off}}$;
    Insert $\Gamma_1^{\text{par}}(i)$ in the second last position of Γ';
    if $t_{arr}(\Gamma') \leq t_0 + T$ **then** $\Gamma_1^{\text{off}} \leftarrow \Gamma'$;
end
for $i = r_2 + 1 \rightarrow l_2 - 1$ **do**
    $\Gamma' \leftarrow \Gamma_1^{\text{off}}$;
    if $\Gamma_1^{par}(i)$ *is not in* Γ' **then**
        Insert $\Gamma_2^{\text{par}}(i)$ in the second last position of Γ';
        if $t_{arr}(\Gamma') \leq t_0 + T$ **then** $\Gamma_1^{\text{off}} \leftarrow \Gamma'$;
    end
end
// Generate the second offspring
for $i = 2 \rightarrow r_2$ **do**
    $\Gamma' \leftarrow \Gamma_2^{\text{off}}$;
    Insert $\Gamma_2^{\text{par}}(i)$ in the second last position of Γ';
    if $t_{arr}(\Gamma') \leq t_0 + T$ **then** $\Gamma_1^{\text{off}} \leftarrow \Gamma'$;
end
for $i = r_1 + 1 \rightarrow l_1 - 1$ **do**
    $\Gamma' \leftarrow \Gamma_2^{\text{off}}$;
    if $\Gamma_1^{par}(i)$ *is not in* Γ' **then**
        Insert $\Gamma_1^{\text{par}}(i)$ in the second last position of Γ';
        if $t_{arr}(\Gamma') \leq t_0 + T$ **then** $\Gamma_2^{\text{off}} \leftarrow \Gamma'$;
    end
end
return $(\Gamma_1^{\text{off}}, \Gamma_2^{\text{off}})$;

Algorithm 7: The local search process

$\Omega \leftarrow \{v \in V | v \text{ is not in } \Gamma\}$; // *the candidate vertices*
Randomly shuffle Ω;
foreach $v \in \Omega$ **do**
    $t^*_{\text{arr}} \leftarrow t_0 + T$, $i^* \leftarrow -1$;
    for $i = 2 \rightarrow |\Gamma| - 1$ **do**
        $\Gamma' \leftarrow \Gamma$;
        Insert v into position i of Γ';
        if $t_{arr}(\Gamma') < t^*_{arr}$ **then** $t^*_{\text{arr}} \leftarrow t_{\text{arr}}(\Gamma')$, $i^* \leftarrow i$;
    end
    if $i^* \neq -1$ **then** Insert v into position i^* of Γ;
end

the current pheromone distribution in the network, and then the pheromone distribution is updated based on the newly generated solutions.

Algorithm 8: The Multi-objective Ant Colony System

Input: α, β, ϕ, ρ, q_0
Output: A set of non-dominated solutions $\mathbf{\Gamma}^{\text{nd}}$
// Initialization
$\mathbf{\Gamma}^{\text{nd}} \leftarrow \emptyset$;
foreach $(v_i, v_j) \in V \times V$ **do**
  for $k = 1 \rightarrow m$ **do**
    $\tau_k(v_i, v_j) \leftarrow \tau_{\text{init}}$; *// initial pheromone*
    $\eta_k(v_i, v_j) \leftarrow s_{jk}/\delta(v_i, v_j)$; *// heuristic information*
  end
end
Construct n_{ant} uniformly distributed m-dimensional weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{n_{\text{ant}}}$, where n_{ant} is the number of ants;
// Search process
while *Stopping criteria not met* **do**
  for $l = 1 \rightarrow n_{ant}$ **do**
    // Generate the solution (tour) Γ_l *for ant* l
    $\Gamma_l = (v_1, v_n)$; *// tour from* v_1 *to* v_n
    repeat
      $v_{\text{next}} \leftarrow \texttt{nextVertex}(\Gamma_l, \boldsymbol{\tau}, \boldsymbol{\eta}, \mathbf{w}_l)$;
      Insert v_{next} to the second last position of Γ_l;
    until $v_{next} = null$;
    $\Gamma_l \leftarrow \texttt{localSearch}(\Gamma_l)$;
    Update $\mathbf{\Gamma}^{\text{nd}}$ with Γ_l;
    $\texttt{localUpdate}(\boldsymbol{\tau}, \Gamma_l)$; *// local pheromone update*
  end
  $\texttt{globalUpdate}(\boldsymbol{\tau})$; *// global pheromone update*
end
return $\mathbf{\Gamma}^{\text{nd}}$;

There have been various implementations of ant colony optimization, such as the ant system [49], ant colony system [48], and MAX-MIN ant system [50]. They differ from each other in generating the solutions and updating the pheromone distribution. In this paper, the Ant Colony System (ACS) framework is adopted due to its demonstrated efficacy on the traveling salesman problem. The framework of the proposed Multi-objective Ant Colony System (MACS) is described in Algorithm 8.

MACS adopts many components of P-ACO [30], which has been demonstrated to be very effective to solve the multi-objective orienteering problem. More specifically, the following mechanisms of P-ACO are adopted in MACS:

1. A set of pheromone structures $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_m)$ and heuristic information $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_m)$ are maintained, each for an objective;

2. The initialization, local update and global update of the pheromone $\boldsymbol{\tau}$ are the same as that in P-ACO;

3. The definition of $\boldsymbol{\eta}$ is the same as that in P-ACO;

4. For each ant Γ_l, a weight vector $\mathbf{w}_l = (w_{l1}, \ldots, w_{lm})$ is defined to aggregate the objective vector $\mathbf{f}(\Gamma_l) = (f_1(\Gamma_l), \ldots, f_m(\Gamma_l))$ into a single value by the weighted sum approach, i.e. $f_{\text{agg}}(\Gamma_l) = \sum_{k=1}^{m} w_{lk} f_k(\Gamma_l)$;

5. The weight vectors of the ants are uniformly distributed;

6. During the solution generation, the Best-or-Roulette-Wheel selection scheme is adopted. That is, the best feasible vertex is selected with a probability of q_0. Otherwise, the probability of selecting each vertex is defined in the same way as that in P-ACO.

The only difference between MACS and P-ACO is the iterative improvement procedure. P-ACO uses the 2-opt operator in the local search, which is effective in reducing the total length of the tour. However, it does not work well anymore in the time-dependent scenario, since a shorter length does not necessarily lead to a smaller travel time. To address this issue, MACS employs the simple insertion-based local search (Algorithm 7), which is much more effective and faster to handle the time-dependent travel time.

Compared with MOMA, there is not much novelty in MACS besides the adoption of the new iterative improvement procedure. However, due to the competitiveness of P-ACO for the multi-objective orienteering problem, MACS is expected to perform well on MOTDOP, and give a decent baseline performance for comparison.

3.3. Complexity Analysis

The complexity of MOMA can be calculated as

$$
\begin{aligned}
O(\text{MOMA}) &= FE_{\max} \cdot (O(\text{XO}) + O(\text{LS})) && (9)\\
&= FE_{\max} \cdot \left(O(a) + O\left(\frac{(n+a+1)(n-a)}{2} \right) \right) && (10)\\
&= FE_{\max} \cdot O\left(n^2 - a^2 + a \right), && (11)
\end{aligned}
$$

where $O(\text{XO})$ and $O(\text{LS})$ are the complexity of the crossover and local search, $FE_{\max}$ stand for the maximal number of fitness evaluations, n is the problem size and a is the number of vertices in the tour.

The computational complexity of MACS can be calculated as

$$
O(\text{MACS}) = N_{\max}(n_{\text{ant}}(O(\text{SG}) + O(\text{LS}) + O(\text{LU})) + O(\text{GU})),
$$

where $N_{\max}$ is the maximal number of iterations. $O(\text{SG})$, $O(\text{LS})$, $O(\text{LU})$ and $O(\text{GU})$ stand for the computational complexity of the solution generation, local search, local and global pheromone update, respectively. It is known that

$$
\begin{aligned}
O(\text{SG}) &= O\left(\sum_{i=1}^{a} i(n-i)\right) = O(a^3), \\
O(\text{LS}) &= O\left(\sum_{i=a+1}^{n} i\right) = O\left(\frac{(n+a+1)(n-a)}{2}\right), \\
O(\text{LU}) &= O(\text{GU}) = m \cdot O(n^2),
\end{aligned}
$$

where n is the problem size, i.e., the number of vertices in the graph, and a is the expected number of vertices in the tour, depending on the time budget. Therefore,

$$
\begin{aligned}
O(\text{MACS}) &= N_{\max} \cdot n_{\text{ant}} \cdot O(a^3 + mn^2) && (12) \\
&= FE_{\max} \cdot O(a^3 + mn^2). && (13)
\end{aligned}
$$

When comparing the complexity of MOMA (Eq. (11)) and MACS (Eq. 13)), one can see that given the same $FE_{\max}$ value, MOMA has a much lower complexity than MACS.

3.4. Handling Time-Dependent Travel Time

Note that there are a large number of vertex insertion operations in MOMA and MACS (e.g. in the crossover and local search of MOMA, and solution construction and local search in MACS). Before inserting a vertex, the feasibility of the insertion needs to be checked. In the static scenario, one can calculate the local change only. However, when the travel time is time-dependent, an insertion influences the travel time of all the subsequent vertices, and may dramatically change the final arrival time. It is time-consuming to calculate for all the subsequent vertices. To address this issue, we store the latest arrival time $\overline{\tau}(\Gamma(i))$ for each vertex in the tour of Γ. It is initialized as follows:

$$
\begin{aligned}
&\overline{\tau}(\Gamma(L)) = t_0 + T, && (14) \\
&\overline{\tau}(\Gamma(i)) = \overline{\tau}(\Gamma(i+1)) - d'_{\Gamma(i),\Gamma(i+1)}(\overline{\tau}(\Gamma(i+1))),\ j = L-1, \ldots, 1, && (15)
\end{aligned}
$$

where $d'_{\Gamma(i),\Gamma(i+1)}(\overline{\tau}(\Gamma(i+1)))$ is the travel time from $\Gamma(i)$ to $\Gamma(i+1)$ when arriving at time $\overline{\tau}(\Gamma(i+1))$.

When inserting a new vertex v between $\Gamma(j-1)$ and $\Gamma(j)$, the feasibility can be simply checked by examining whether the new arrival time of $\Gamma(j)$ is later than $\overline{\tau}(\Gamma(i))$. This way, there is no need to calculate the new arrival time of all the subsequent vertices, and the feasibility check is made much more efficient.

Once a new vertex is inserted, the latest arrival time of all its precedent vertices in the tour are updated by Eq. (15) with the index from $j-1$ to 1, where j is the inserted position.

4. Experimental Studies

In this section, two sets of benchmark instances are generated and the proposed MOMA and MACS are evaluated on them in terms of a number of multi-objective performance measures.

4.1. Benchmark Generation

Two sets of benchmark instances were generated from existing benchmarks. The first one was extended from Verbeeck's single-objective time-dependent instances, which can be downloaded from the orienteering website of the Centre for Industrial Management, Katholieke Universiteit Leuven[3]. The second one was extended from Schilde's multi-objective static instances, which can be downloaded from the Production and Operations Management homepage[4]. The piece-wise time-dependent travel speed model proposed in [35] is adopted here. Specifically, the entire single-day time period from 7AM to 9PM is divided into four period, and the streets are divided into five categories. The travel speed matrix is given in Table 2.

The first benchmark set is extended from single-objective time-dependent instances. In this study, two objectives are considered. The second score vector is obtained by randomly shuffling the first score vector in the original instance. The second benchmark set is extended from multi-objective static instances. This is done by applying the time-dependent travel time matrix to the instances, and assign one of the five street categories to each edge. To this end, for each edge, the category is first randomly sampled from {1,

[3] `http://www.mech.kuleuven.be/en/cib/op/#section-20`

[4] `http://prolog.univie.ac.at/research/OP/`

Table 2: The time-dependent travel speed matrix.

Street Category	**Morning Peak** (7AM–9AM)	**Day Normal** (9AM–5PM)	**Evening Peak** (5PM–7PM)	**Evening Normal** (7PM–9PM)
1. Always Busy	0.5	0.81	0.5	0.81
2. Morning Peak	0.5	0.7	1.0	1.5
3. Two Peaks	0.5	1.5	0.5	1.5
4. Evening Peak	1.0	1.5	0.5	0.7
5. Seldom Traveled	1.5	1.5	1.5	1.5

Table 3: The features of Verbeeck's MOTDOP dataset.

Name	**#Instances**	**#Vertices**	**Time budget**
p1	9	32	5, 6, 7, 8, 9, 10, 11, 12, 13
p2	9	21	5, 6, 7, 8, 9, 10, 11, 12, 13
p3	9	33	5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5
p4	10	100	5, 6, 7, 8, 9, 10, 11, 12, 13, 14
p5	3	66	5.5, 6, 7
p6	9	64	6.5, 7, 8, 9, 10, 11, 12, 13, 14
p7	10	102	4, 5, 6, 7, 8, 9, 10, 11, 12, 13

..., 5}. Then, based on the assumption that adjacent streets are likely to be in the same category, we repeat the category propagation of each edge to its neighbors for 50 iterations. For the sake of simplicity, the two generated benchmark sets will be denoted as Verbeeck's and Schilde's MOTDOP datasets hereafter.

Tables 3 and 4 show the features of the generated Verbeeck's and Schilde's MOTDOP datasets. Both sets include seven groups. The instances within the same group share the same number of vertices, but with different time budgets. All the instances have a starting time of 7AM. The generated two MOTDOP datasets can be downloaded from the author's homepage[5].

4.2. Experiment Settings

MOTDOP is a new problem, and there are no existing methods directly solving it. However, to show the efficacy of the proposed MOMA and MACS, it is necessary to still compare them with existing methods on MOTDOP. To this end, the MOEA proposed by De Falco *et al.* [10] (denoted as FMOEA) for solving the static problem model is included in the comparison as well.

[5] `http://homepages.ecs.vuw.ac.nz/~yimei/Research-JP.html`

Table 4: The features of Schilde's MOTDOP dataset.

Name	#Instances	#Vertices	Time budget
p21	6	21	1.5, 2, 2.5, 3, 3.5, 4
p32	7	32	2, 3, 4, 5, 6, 7, 8
p33	9	33	2, 3, 4, 5, 6, 7, 8, 9, 10
p64	7	64	2, 3, 4, 5, 6, 7, 8
p66	12	66	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
p559	11	559	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
p562	13	562	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

FMOEA adopts a steady-state multi-objective evolutionary algorithm framework. In each generation, the uniform crossover operator is first applied to two randomly selected non-dominated solutions in the current population to generate an offspring. Then, the offspring undergoes the mutation with a certain probability, with either the exchange or the 2-opt operator. Finally, the offspring replaces the corresponding individual in the current population if it dominates that individual, and the search goes to the next generation.

Two modifications were made on FMOEA to adapt to the time-dependent problem. First, due to the efficacy of local search in solving many combinatorial optimization problems (e.g. [44, 38]), the local search used in MOMA and MACS (Algorithm 7) was added after the mutation of FMOEA to further improve the solution quality. Second, the parameter values of FMOEA were changed to make a fair comparison with MOMA and MACS and adapt to the included local search. Specifically, the population size was changed from 500 to 30, and the number of fitness evaluations were reduced from 50000 to 3750. Despite the significant computational reduction, the solution quality can still be guaranteed by the added local search.

The three compared algorithms are characterized as follows:

- FMOEA is the latest algorithm for multi-objective orienteering problem, which can be seen as the state-of-the-art existing algorithm;
- MACS adopts many promising components of P-ACO [30], which is the most recent and state-of-the-art ant colony system for multi-objective orienteering problem. Thus, it can be seen as an extension of P-ACO from static orienteering problem to the time-dependent variant;
- MOMA is a newly proposed memetic algorithm specifically designed for MOTDOP. Problem-specific crossover, local search and selection

operators are designed or employed from relevant work for efficiently solving MOTDOP.

The parameter settings of MOMA, MACS and FMOEA are given in Table 5, respectively. The maximal number of fitness evaluations is set to 3750 for all the compared algorithms. For MOMA and FMOEA, the population size is set to 30, which is the best value found by preliminary study. For MACS, the parameters of τ_{init}, n_{ant}, α, β and ϕ are set according to the recommendations in the ACS for solving single-objective time-dependent orienteering problem [35]. After comparing different ρ and q_0 values in preliminary study, we chose $\rho = 0.005$ and $q_0 = 0.25$. which yielded the best results among a wide range of values. The crossover, exchange and 2-opt probabilities for FMOEA are set to 0.4, 0.8 and 1.0, following the suggestions in [10]. For all the algorithms, the local search is always applied to the generated offsprings.

Table 5: Parameter settings of MOMA and MACS.

Parameter	**Description**	**Value**
$FE_{\max}$ (All)	Maximal fitness evaluations	3750
popsize (MOMA&FMOEA)	Population size	30
τ_{init} (MACS)	Initial pheromone value	1
n_{ant} (MACS)	Number of ants	75
α (MACS)	Relative influence factor of the pheromone values	4
β (MACS)	Relative influence factor of the heuristic information	$0.07 \cdot n$
ϕ (MACS)	Local pheromone evaporation rate	0.01
ρ (MACS)	Global pheromone evaporation rate	0.005
q_0 (MACS)	Probability of selecting the best vertex	0.25
(*CR*, *EM*, *FV*) (FMOEA)	Crossover, exchange and 2-opt probabilities	(0.4, 0.8, 1.0)

Note that MOMA has fewer parameters than MACS and FMOEA. In fact, given the computational budget (maximal number of fitness evaluations), there is only one parameter, i.e. the population size[6], that affects the search behaviour of MOMA. This makes MOMA less sensitive to the parameter settings than MACS and FMOEA.

For each parameter setting, 30 independent runs of each compared algorithm were conducted on all the test instances. The algorithms were pro-

[6]The number of generations, which is the other intertwined parameter, is set accordingly to the number of fitness evaluations over the population size.

grammed in Java, compiled in JRE 1.8.0_40 and run in a platform with Intel Core i5 CPU, 8 GB DDR3 memory and OS X Yosemite.

4.3. Performance Metrics

The hypervolume, unary epsilon, spacing and range cover indicators are used to evaluate the proposed algorithms from different aspects. All these metrics have been adopted in the experimental studies for the multi-objective static orienteering problem [30], reflecting different aspects of the evaluated set of solutions. They are described as follows:

4.3.1. Hypervolume Indicator (I_H)

The hypervolume indicator implies the area in the objective space that is dominated by the given set of solutions $\mathcal{X}$. For maximizing the scores in MOTDOP, the nadir point is set to $\mathbf{0}$. For the bi-objective problems, given a set of non-dominated solutions $\mathcal{X}$, the two-dimensional hypervolume $I_H(\mathcal{X})$ can be calculated as in Algorithm 9. A larger I_H value indicates that the set dominates a larger area, and thus is better.

Algorithm 9: Calculation of two-dimensional hypervolume

```
1 Sort the solutions x ∈ X in the increasing order of f_1;
2 I_H(X) ← f_1(x_1) f_2(x_1);                    // calculate the first area
3 for i = 2 → |X| do
4     I_H(X) ← I_H(X) + (f_1(x_i) − f_1(x_{i−1})) · f_2(x_i);
5 end
6 return I_H(X);
```

4.3.2. Unary Epsilon Indicator (I_ϵ)

This indicator is based on the ϵ-dominance relation ($\succ_\epsilon$). In a maximization problem, a solution x_1 is said to ϵ-dominate another solution x_2 ($x_1 \succ_\epsilon x_2$), if

- $\forall\, i \in \{1, \ldots, m\}$, $\epsilon \cdot f_i(x_1) \geq f_i(x_2)$, and
- $\exists\, i \in \{1, \ldots, m\}$, $\epsilon \cdot f_i(x_1) > f_i(x_2)$.

Then, the I_ϵ value of a set of solutions $\mathcal{X}$ with respective to an approximated Pareto-optimal set $\mathcal{R}$ is defined as follows:

$$I_\epsilon(\mathcal{X}, \mathcal{R}) = \inf_{\epsilon \in \mathbb{R}} \{\forall\, x_2 \in \mathcal{R},\ \exists\, x_1 \in \mathcal{X},\ x_1 \succ_\epsilon x_2\}. \tag{16}$$

A smaller I_ϵ value implies a closer set to the Pareto front.

4.3.3. Spacing Indicator (I_S)

The spacing indicator evaluates how uniformly the solutions in the given set $\mathcal{X}$ are distributed in the objective space. It is defined as follows:

$$I_S(\mathcal{X}) = \sqrt{\frac{1}{|\mathcal{X}| - 1} \sum_{x \in \mathcal{X}} (D(x) - \overline{D})}, \tag{17}$$

where $D(x)$ stands for the Euclidean distance between x and its nearest neighbor in $\mathcal{X}$ in the objective space. A smaller value indicates a more uniformly distributed set. This indicator requires at least two solutions in $\mathcal{X}$.

4.3.4. Range Cover Indicator (I_R)

The range cover indicator reflects the spread of the set $\mathcal{X}$ in the objective space. It is simply calculated as follows:

$$I_R(\mathcal{X}) = \frac{1}{m} \sum_{i=1}^{m} \left(\max_{x \in \mathcal{X}} f_i(x) - \min_{x \in \mathcal{X}} f_i(x) \right). \tag{18}$$

A larger I_R value implies a wider spread of the set. If there is only one solution in the set, I_R will be zero.

4.3.5. Normalization and Pareto Front Approximation

All the above indicators were calculated after the normalization for the objective values. Each objective is normalized between 1 and 2, where 1 and 2 stand for the worst and best values obtained by all the runs of all the algorithms. Since the true Pareto front is unknown, we combined the solutions obtained by all the runs of all the algorithms together, and extracted the non-dominated solutions out of them to be the approximated Pareto-optimal set $\mathcal{R}$. This will be used for calculating the I_ϵ value.

4.4. Results and Discussions

Due to the space limit, we only provide the summary of the comparisons, which does not affect our discussion and analysis. The full details of the results are given in the Appendix. For each dataset, the summary was obtained in the following steps:

1. For each algorithm and each instance, we calculated the mean value of the performance metrics over the 30 independent runs;

2. For each group, we calculated the overall mean value of the performance metrics by averaging the corresponding mean values of all the instances in that group;

3. For each instance and each performance metric, we conducted the Wilcoxon's rank sum test [51] between the 30 results of FMOEA, MACS and MOMA under the significance level of 0.05.

4. For each group, each performance metric and each algorithm, we count the number of instances on which the algorithm obtained significantly better results than the other compared algorithms.

Table 6 gives the summary of the comparison between MOMA, MACS and FMOEA on the groups of Verbeeck's MOTDOP dataset. It is clear that MOMA significantly outperforms FMOEA and MACS in terms of I_H and I_ϵ on most of Verbeeck's MOTDOP instances. In summary, MOMA obtains significantly better results on 51/59 instances in terms of I_H and 56/59 instances in terms of I_ϵ. MACS performs worse than MOMA, obtaining slightly smaller I_H value and larger I_ϵ value. FMOEA performs the worst in terms of I_H and I_ϵ.

In terms of I_S, FMOEA fails to found trade-off solutions in many cases, resulting in many invalid I_S values (shown as "-" in the table). MACS and MOMA perform better than FMOEA and achieve trade-off solutions on more instances. In terms of I_R, MACS performs better than MOMA and FMOEA on more instances, indicating that the extreme points of MACS are more distant from each other than that of MOMA and FMOEA.

Then, we compare FMOEA, MACS and MOMA on Schilde's MOTDOP dataset. The summary of the results are given in Table 7. From the table, one can see that the relative performances of the compared algorithms are similar to that in Table 6. MOMA significantly outperforms MACS and FMOEA on 62/65 instances in both I_H and I_ϵ. MACS performs better than FMOEA when the problem size is not large. However, for the p559 and p562 large-scale groups, FMOEA outperforms MACS. In terms of I_S and I_R, the relative performances of the compared algorithms are mixed. MOMA performs slightly better in terms of uniformity and spread.

In summary, we can conclude that for the tested MOTDOP benchmarks, MOMA manages to achieve solutions that are closer to the Pareto front and more uniformly distributed. However, the solutions obtained by MOMA may spread more narrowly than that of MACS. This may be due to the nature

Table 6: The summary of the performance of FMOEA, MACS and MOMA on the seven groups of Verbeeck's MOTDOP dataset in terms of the four selected performance metrics.

Group		I_H			I_ϵ			I_S			I_R		
		FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p1	Mean	3.14	3.67	**3.77**	1.20	1.10	**1.06**	-	0.04	0.04	0.13	**0.22**	0.16
	#S.B.	0/9	2/9	**7**/9	0/9	0/9	**9**/9	0/9	0/9	0/9	1/9	**6**/9	0/9
p2	Mean	3.54	3.35	**3.71**	1.11	1.20	**1.05**	0.06	**0.05**	0.09	0.36	0.40	**0.44**
	#S.B.	0/9	1/9	**8**/9	0/9	1/9	**8**/9	1/9	**2**/9	0/9	0/9	4/9	4/9
p3	Mean	2.94	3.20	**3.72**	1.25	1.19	**1.07**	-	-	**0.08**	0.15	0.14	**0.18**
	#S.B.	0/9	0/9	**9**/9	0/9	0/9	**9**/9	0/9	0/9	0/9	0/9	1/9	**4**/9
p4	Mean	2.07	3.34	**3.53**	1.46	1.14	**1.09**	-	0.03	**0.02**	0.13	**0.36**	0.20
	#S.B.	0/10	0/10	**10**/10	0/10	0/10	**10**/10	0/10	0/10	0/10	0/10	**10**/10	0/10
p5	Mean	2.83	3.26	**3.56**	1.23	1.14	**1.08**	-	0.05	**0.02**	0.15	0.19	0.19
	#S.B.	0/3	0/3	**3**/3	0/3	0/3	**3**/3	0/3	0/3	0/3	0/3	0/3	**1**/3
p6	Mean	2.35	3.35	**3.71**	1.39	1.13	**1.06**	0.04	0.03	0.03	0.16	**0.20**	0.18
	#S.B.	0/9	0/9	**9**/9	0/9	0/9	**9**/9	0/9	0/9	**1**/9	2/9	**5**/9	0/9
p7	Mean	1.97	3.39	**3.40**	1.49	1.13	**1.11**	**0.02**	0.03	0.03	0.13	**0.34**	0.21
	#S.B.	0/10	5/10	5/10	0/10	2/10	**8**/10	0/10	0/10	0/10	0/10	**10**/10	0/10

· "Mean" stands for the overall mean value of each algorithm on each group of instances;
· "#S.B." indicates the fraction of instances in each group on which one algorithm performed significantly better than the other under the significance level of 0.05;
· For each group and each performance metric, the result of the better algorithm is marked in bold.

of the true Pareto front, which may be narrowly distributed itself. MACS performs better than FMOEA when the problem size is not large. However, its performance deteriorates with the increase of the problem size.

Note that FMOEA can be seen as an existing state-of-the-art algorithm for multi-objective orienteering problems. MACS can be considered as an extension of P-ACO [30] from the static problem to the time-dependent one. Then, the outperformance of MOMA over FMOEA and MACS, especially in terms of I_H and I_ϵ, indicates that MOMA performs significantly better than the state-of-the-art algorithms for MOTDOP.

Figs. 2 and 3 show the objectives of all the solutions obtained by the 30 runs of FMOEA, MACS and MOMA on the representative Verbeeck's and Schilde's instances. For Verbeeck's dataset, the two largest groups (p4 and p7) are selected, and the instances with the shortest and longest time budgets were selected. For Schilde's dataset, four groups with problem size from medium to large were selected. For each group, the instance with a medium time budget was selected. From the figures, one can see that MOMA managed to find better sets of solutions than MACS and FMOEA. This is

Table 7: The summary of the performance of FMOEA, MACS and MOMA on the seven groups of Schilde's MOTDOP dataset in terms of the four selected performance metrics.

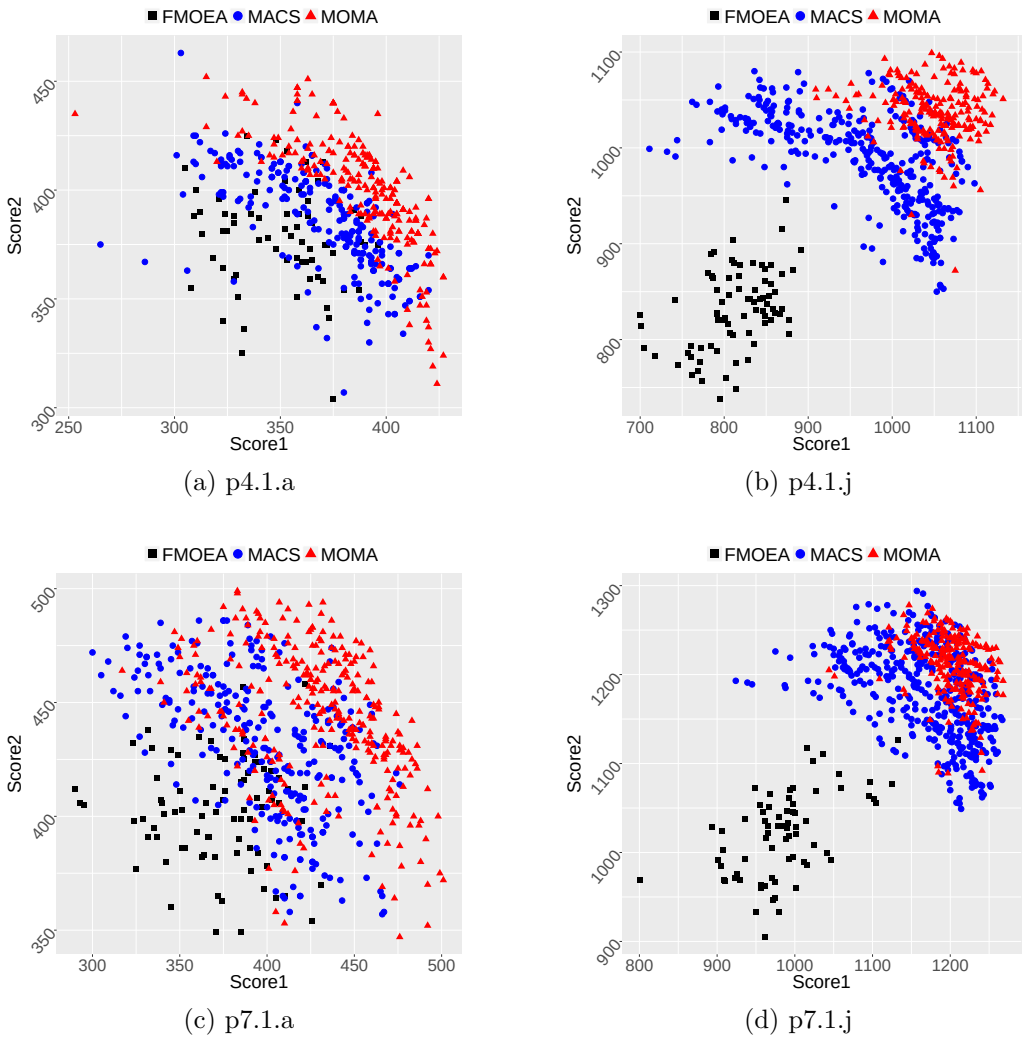
Group		I_H			I_ϵ			I_S			I_R		
		FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p21	Mean	3.40	2.95	**3.61**	1.17	1.33	**1.10**	-	-	-	0.03	0.15	**0.17**
	#S.B.	1/6	0/6	**4**/6	0/6	0/6	**4**/6	0/6	0/6	0/6	0/6	2/6	**3**/6
p32	Mean	2.74	3.33	**3.42**	1.32	1.15	**1.14**	0.31	**0.07**	0.12	0.24	0.38	**0.43**
	#S.B.	0/7	1/7	**6**/7	0/7	1/7	**6**/7	0/7	**1**/7	0/7	0/7	1/7	**6**/7
p33	Mean	3.15	3.42	**3.77**	1.20	1.15	**1.05**	0.08	0.13	**0.05**	0.19	**0.39**	0.37
	#S.B.	0/9	0/9	**9**/9	0/9	0/9	**9**/9	1/9	0/9	1/9	0/9	4/9	4/9
p64	Mean	2.57	3.28	**3.62**	1.33	1.15	**1.07**	0.05	0.05	**0.04**	0.19	**0.36**	0.33
	#S.B.	0/7	0/7	**7**/7	0/7	0/7	**7**/7	0/7	0/7	**2**/7	0/7	**4**/7	2/7
p66	Mean	2.69	3.38	**3.63**	1.32	1.15	**1.07**	0.05	0.05	**0.03**	0.18	**0.46**	0.32
	#S.B.	0/12	0/12	**12**/12	0/12	0/12	**12**/12	0/12	0/12	**3**/12	2/12	**9**/12	1/12
p559	Mean	2.68	1.51	**3.41**	1.30	1.73	**1.12**	-	**0.02**	0.03	0.07	0.06	**0.17**
	#S.B.	0/11	0/11	**11**/11	0/11	0/11	**11**/11	0/11	0/11	0/11	0/11	0/11	**10**/11
p562	Mean	2.48	2.35	**3.33**	1.38	1.42	**1.14**	-	0.05	**0.05**	0.13	0.20	**0.26**
	#S.B.	0/13	0/13	**13**/13	0/13	0/13	**13**/13	0/13	0/13	0/13	0/13	4/13	**9**/13

· "Mean" stands for the overall mean value of each algorithm on each group of instances;
· "#S.B." indicates the fraction of instances in each group on which one algorithm performed significantly better than the other under the significance level of 0.05;
· For each group and each performance metric, the result of the better algorithm is marked in bold.

consistent with the superior performance metric values of MOMA over both MACS and FMOEA in Table 7. In some figures, (e.g. Figs. 2b and 2d), it is shown that the front obtained by MOMA tends to be narrower than the front obtained by MACS. This might be due to the non-dominated sorting procedure adopted by MOMA, which tends to focus the search in the middle of the Pareto front.

In addition, it can be seen that the sets of solutions showed a better distribution on the smaller instances (p64 and p66). However, the distributions for p559 and p562 were either too narrow (p559) or with a poor shape (MOMA in p562). This implies that 3750 fitness evaluations might not be sufficient for the algorithms to converge to a good-shaped front for the large instances.

Fig. 4 shows the problem size (number of POIs) versus computational time (in seconds) of the compared algorithms. From the figure, it is clear that both MOMA and FMOEA have a much better scalability than MACS. When the problem size is more than 500, the computational time of MACS increases to over 2000 seconds, while that of MOMA and FMOEA are still



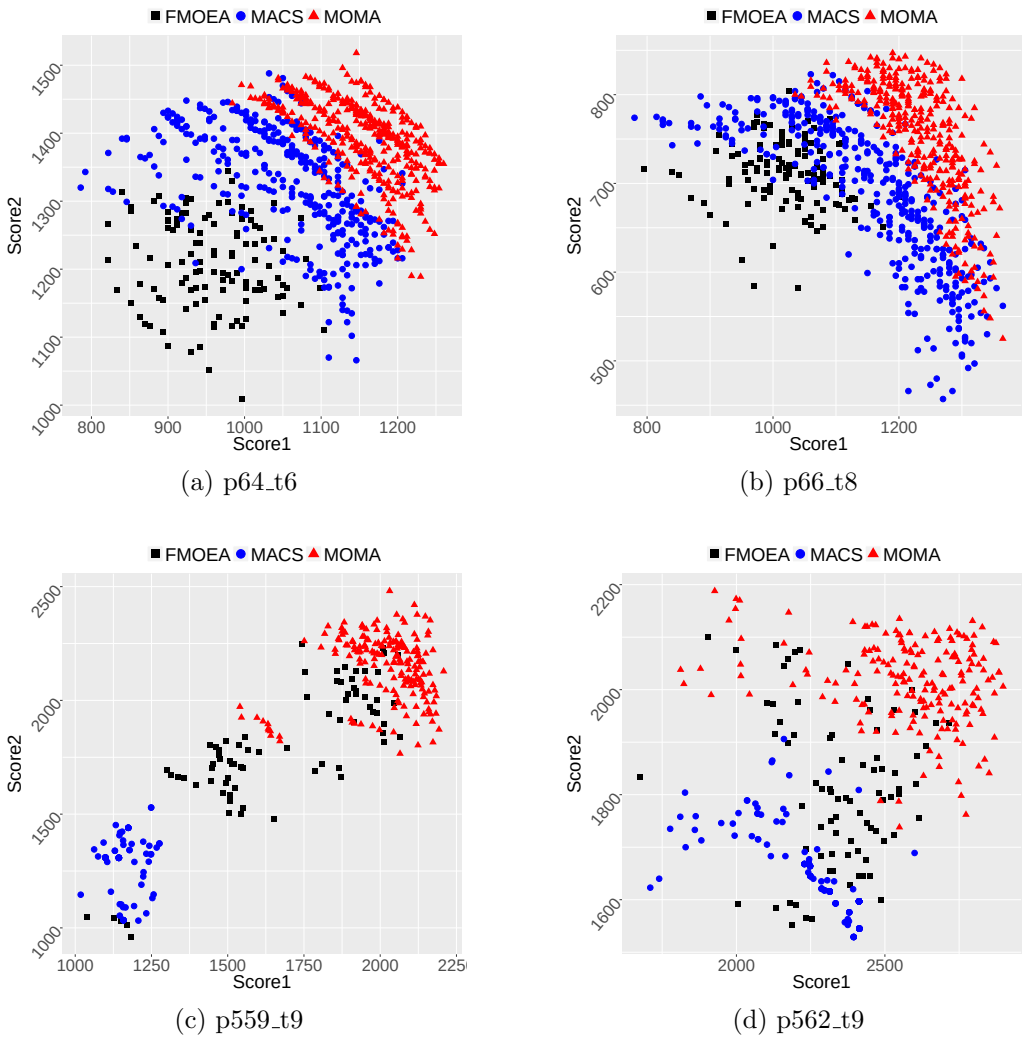
(a) p4.1.a (b) p4.1.j

(c) p7.1.a (d) p7.1.j

Figure 2: Objective values of all the solutions obtained by the 30 runs of FMOEA, MACS and MOMA on some representative Verbeeck's MOTDOP instances.

less than one minute. This is consistent with the complexity analysis, which shows that MOMA has a much less computational complexity than MACS (Eq. (11) versus Eq. (13)). FMOEA has a similar complexity as MOMA, as they adopt similar frameworks and search operators.

To further verify the efficacy of MOMA, it is compared with the state-of-the-art ACS [35] on single-objective time-dependent benchmark instances.



(a) p64_t6 (b) p66_t8

(c) p559_t9 (d) p562_t9

Figure 3: Objective values of all the solutions obtained by the 30 runs of FMOEA, MACS and MOMA on some representative Schilde's MOTDOP instances.

For MOMA, 30 independent runs were conducted with the parameter settings given in Table 5. The results of ACS were directly obtained from [35]. It is important to note that MOMA was specifically designed for MOTDOP, and may not be perfectly suitable for the single-objective problem (e.g. the selection scheme becomes the truncation selection, which might make the algorithm too greedy). The results are shown in Table 8. It can be seen that

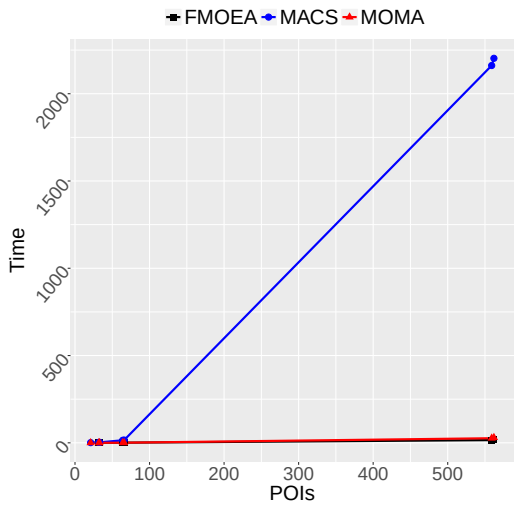


Figure 4: The number of POIs versus computational time of the compared algorithms.

MOMA still performs similarly to ACS. It manages to achieve the optimal solution for all the instances, which is better than ACS, although its mean and worst-case performances are slightly worse. This demonstrates the efficacy of MOMA, not to mention its advantage over single-objective-based methods in obtaining a set of trade-off solutions in a single run.

In summary, the following conclusions can be drawn from the experimental studies:

- Both MACS and MOMA performed much better than FMOEA. This demonstrates the efficacy of the problem-specific search framework of MACS (ant colony system) and MOMA (non-dominated sorting);
- MOMA achieved better results than MACS in most cases. More importantly, MOMA is much more efficient than MACS. This is due to the advantageous framework MOMA employs, which replaces the computationally expensive solution generation procedure with a more efficient local modification (crossover). This demonstrates the efficacy of using the memetic algorithm framework to solve MOTDOP.

5. Conclusion

In this paper, the multi-objective time-dependent orienteering problem is investigated, and two meta-heuristic algorithms are designed for solving this

Table 8: The best, mean and worst gap % of ACS and MOMA to the optimal value on the single-objective time-dependent benchmark instances.

Instance	Best Gap %		Mean Gap %		Worst Gap %	
	ACS	MOMA	ACS	MOMA	ACS	MOMA
p1.1.a	0.0	0.0	0.0	1.4	0.0	4.3
p1.1.b	0.0	0.0	0.0	0.1	0.0	3.7
p1.1.c	0.0	0.0	0.0	0.1	0.0	3.1
p1.1.d	0.0	0.0	2.2	2.6	5.4	5.4
p1.1.e	0.0	0.0	0.5	1.5	2.4	7.1
p1.1.f	0.0	0.0	0.4	0.1	2.2	2.2
p1.1.g	0.0	0.0	0.0	0.7	0.0	4.0
p1.1.h	1.9	0.0	1.9	2.1	1.9	5.6
p2.1.a	0.0	0.0	0.0	0.0	0.0	0.0
p2.1.b	0.0	0.0	0.0	0.0	0.0	0.0
p2.1.c	0.0	0.0	0.0	0.0	0.0	0.0
p2.1.d	0.0	0.0	0.0	0.0	0.0	0.0
p2.1.e	0.0	0.0	0.0	0.5	0.0	3.8
p2.1.f	0.0	0.0	0.0	0.2	0.0	4.8
p2.1.g	0.0	0.0	0.0	1.2	0.0	5.9
p2.1.h	0.0	0.0	0.0	1.3	0.0	9.3
p2.1.i	0.0	0.0	0.0	0.1	0.0	3.5
p3.1.a	0.0	0.0	4.3	2.1	5.4	8.1
p3.1.b	0.0	0.0	0.0	0.1	0.0	2.4
p3.1.c	4.0	0.0	4.0	2.1	4.0	6.0
p3.1.d	0.0	0.0	2.9	1.9	3.6	5.4
p3.1.e	0.0	0.0	1.9	2.0	4.8	6.5
p3.1.f	0.0	0.0	0.3	0.7	1.5	6.2
p3.1.g	0.0	0.0	1.2	2.3	2.9	8.7

complex problem. Although the features of multi-objective optimization and time-dependent travel time have been considered separately, they have not been included in the problem simultaneously before. In this work, they are considered together for the first time. Specifically, we addressed the issues of multi-objective local search and rugged fitness landscape caused by the time-dependent travel time. Then, a Multi-Objective Memetic Algorithm (MOMA) and a Multi-objective Ant Colony System (MACS) are proposed.

To evaluate the proposed algorithms, two sets of MOTDOP benchmark instances with two objectives were generated from the single-objective time-dependent orienteering problem [35] benchmark instances and the multi-objective static orienteering problem benchmark instances [30]. MACS and MOMA were compared with each other, and with another latest multi-objective evolutionary algorithm (FMOEA) for orienteering problem. The experimental results show that both MACS and MOMA can find better sets of solutions than FMOEA. In addition, MOMA performed much better than

MACS in terms of both solution quality and speed. Due to the better scalability, MOMA can obtain a set of better solutions than MACS in a much shorter time. Also, MOMA is much less sensitive to the parameter setting, due to the small number of parameters. Since MACS adopts many promising features of the state-of-the-art ant colony system (P-ACO) for the multi-objective orienteering problem, it can be considered as a baseline and state-of-the-art algorithm for MOTDOP. Therefore, the advantage of MOMA over MACS demonstrates the efficacy of MOMA in solving MOTDOP, and suggests a great potential of using the memetic algorithm framework for solving this challenging problem efficiently.

In the future, more realistic factors such as time windows, lunch time and real-time tour changing will be considered in our model, allowing representation of more realistic orienteering problem scenarios.

Acknowledgment

This work was supported by an RMIT near-miss fund, an ARC Discovery grant (No. DP120102205) and an iCO2mmunity research project, funded by RMIT Sustainable Urban Precincts Program (SUPP).

References

[1] B. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Research Logistics (NRL) 34 (3) (1987) 307–318.

[2] P. Vansteenwegen, D. Van Oudheusden, The mobile tourist guide: an or opportunity, OR Insight 20 (3) (2007) 21–27.

[3] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. Berghe, D. Van Oudheusden, A personalized tourist trip design algorithm for mobile tourist guides, Applied Artificial Intelligence 22 (10) (2008) 964–985.

[4] R. Abbaspour, F. Samadzadegan, Time-dependent personal tour planning and scheduling in metropolises, Expert Systems with Applications 38 (10) (2011) 12439–12452.

[5] D. Gavalas, M. Kenteris, C. Konstantopoulos, G. Pantziou, Web application for recommending personalised mobile tourist routes, IET Software 6 (4) (2012) 313–322.

[6] K. Sylejmani, J. Dorn, N. Musliu, A tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning, in: Hybrid Intelligent Systems (HIS), 2012 12th International Conference on, IEEE, 2012, pp. 300–305.

[7] A. Garcia, P. Vansteenwegen, O. Arbelaitz, W. Souffriau, M. Linaza, Integrating public transportation in personalised electronic tourist guides, Computers & Operations Research 40 (3) (2013) 758–774.

[8] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, Mobile recommender systems in tourism, Journal of Network and Computer Applications 39 (2014) 319–333.

[9] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, A survey on algorithmic approaches for solving tourist trip design problems, Journal of Heuristics 20 (3) (2014) 291–328.

[10] I. De Falco, U. Scafuri, E. Tarantino, A multiobjective evolutionary algorithm for personalized tours in street networks, in: Applications of Evolutionary Computation, Springer, 2015, pp. 115–127.

[11] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, N. Vathis, Heuristics for the time dependent team orienteering problem: Application to tourist route planning, Computers & Operations Research 62 (2015) 36–50.

[12] G. Laporte, S. Martello, The selective travelling salesman problem, Discrete applied mathematics 26 (2) (1990) 193–207.

[13] A. Leifer, M. Rosenwein, Strong linear programming relaxations for the orienteering problem, European Journal of Operational Research 73 (3) (1994) 517–523.

[14] T. Tsiligirides, Heuristic methods applied to orienteering, Journal of the Operational Research Society (1984) 797–809.

[15] I. Chao, B. Golden, E. Wasil, A fast and effective heuristic for the orienteering problem, European Journal of Operational Research 88 (3) (1996) 475–489.

[16] M. Gendreau, G. Laporte, F. Semet, A tabu search heuristic for the undirected selective travelling salesman problem, European Journal of Operational Research 106 (2) (1998) 539–545.

[17] X. Wang, B. Golden, E. Wasil, Using a genetic algorithm to solve the generalized orienteering problem, in: The vehicle routing problem: latest advances and new challenges, Springer, 2008, pp. 263–274.

[18] H. Tang, E. Miller-Hooks, A tabu search heuristic for the team orienteering problem, Computers & Operations Research 32 (6) (2005) 1379–1407.

[19] S. Boussier, D. Feillet, M. Gendreau, An exact algorithm for team orienteering problems, 4or 5 (3) (2007) 211–230.

[20] L. Ke, C. Archetti, Z. Feng, Ants can solve the team orienteering problem, Computers & Industrial Engineering 54 (3) (2008) 648–665.

[21] P. Vansteenwegen, W. Souffriau, G. Berghe, D. Van Oudheusden, Iterated local search for the team orienteering problem with time windows, Computers & Operations Research 36 (12) (2009) 3281–3290.

[22] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, D. Van Oudheusden, The multiconstraint team orienteering problem with multiple time windows, Transportation Science 47 (1) (2013) 53–63.

[23] N. Labadie, R. Mansini, J. Melechovskỳ, R. W. Calvo, The team orienteering problem with time windows: An lp-based granular variable neighborhood search, European Journal of Operational Research 220 (1) (2012) 15–27.

[24] R. Montemanni, L. Gambardella, An ant colony system for team orienteering problems with time windows, Foundations of computing and Decision Sciences 34 (2009) 287–306.

[25] G. Righini, M. Salani, Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming, Computers & Operations Research 36 (4) (2009) 1191–1203.

[26] F. Tricoire, M. Romauch, K. Doerner, R. Hartl, Heuristics for the multi-period orienteering problem with multiple time windows, Computers & Operations Research 37 (2) (2010) 351–367.

[27] N. Labadie, J. Melechovskỳ, R. W. Calvo, Hybridized evolutionary local search algorithm for the team orienteering problem with time windows, Journal of Heuristics 17 (6) (2011) 729–753.

[28] S.-W. Lin, F. Y. Vincent, A simulated annealing heuristic for the team orienteering problem with time windows, European Journal of Operational Research 217 (1) (2012) 94–107.

[29] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, N. Vathis, Efficient heuristics for the time dependent team orienteering problem with time windows, in: Applied Algorithms, Springer, 2014, pp. 152–163.

[30] M. Schilde, K. Doerner, R. Hartl, G. Kiechle, Metaheuristics for the bi-objective orienteering problem, Swarm Intelligence 3 (3) (2009) 179–201.

[31] L. Cotfas, A. Diosteanu, S. Dumitrescu, A. Smeureanu, Semantic search itinerary recommender system, International Journal of Computers 5 (3) (2014) 370–377.

[32] F. Fomin, A. Lingas, Approximation algorithms for time-dependent orienteering, Information Processing Letters 83 (2) (2002) 57–62.

[33] J. Li, Q. Wu, X. Li, D. Zhu, Study on the time-dependent orienteering problem, in: E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on, IEEE, 2010, pp. 1–4.

[34] J. Li, Model and algorithm for time-dependent team orienteering problem, in: Advanced Research on Computer Education, Simulation and Modeling, Springer, 2011, pp. 1–7.

[35] C. Verbeeck, K. Sörensen, E. Aghezzaf, P. Vansteenwegen, A fast solution method for the time-dependent orienteering problem, European Journal of Operational Research 236 (2) (2014) 419–432.

[36] A. Gunawan, H. C. Lau, Z. Yuan, A Mathematical Model and Metaheuristics for Time Dependent Orienteering Problem, Helmut-Schmidt-Univ., Univ. der Bundeswehr Hamburg, 2014.

[37] P. Vansteenwegen, W. Souffriau, D. Van Oudheusden, The orienteering problem: A survey, European Journal of Operational Research 209 (1) (2011) 1–10.

[38] Y. Mei, K. Tang, X. Yao, Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem, IEEE Transactions on Evolutionary Computation 15 (2) (2011) 151–165.

[39] Y. Mei, K. Tang, X. Yao, A Memetic Algorithm for Periodic Capacitated Arc Routing Problem, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41 (6) (2011) 1654–1667.

[40] K. Tang, Y. Mei, X. Yao, Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1151–1166.

[41] Y. Mei, X. Li, X. Yao, Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems, IEEE Transactions on Evolutionary Computation 18 (3) (2014) 435–449.

[42] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Caltech Concurrent Computation Program, C3P Report 826.

[43] H. Ishibuchi, T. Murata, A multi-objective genetic local search algorithm and its application to flowshop scheduling, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 28 (3) (1998) 392–403.

[44] A. Jaszkiewicz, Genetic local search for multi-objective combinatorial optimization, European Journal of Operational Research 137 (1) (2002) 50–71.

[45] M. Tang, X. Yao, A memetic algorithm for VLSI floorplanning, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 37 (1) (2007) 62–69.

[46] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[47] M. Dorigo, M. Birattari, Ant colony optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.

[48] M. Dorigo, L. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.

[49] M. Dorigo, Optimization, learning and natural algorithms, Ph.D. thesis, Politecnico di Milano, Milan, Italy (1992).

[50] T. Stützle, H. Hoos, Max–min ant system, Future generation computer systems 16 (8) (2000) 889–914.

[51] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.

Appendix

Table 9: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOTDOP group p1. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p1.1.a	3.30	**3.73**	3.68	1.17	1.16	**1.08**	-	-	-	0.25	**0.48**	0.41
p1.1.b	3.68	**3.98**	3.95	1.10	1.03	**1.01**	-	-	-	**0.18**	0.11	0.01
p1.1.c	3.30	3.64	**3.78**	1.17	1.10	**1.06**	-	-	-	0.10	**0.22**	0.07
p1.1.d	3.12	3.59	**3.68**	1.19	1.10	**1.08**	-	-	-	0.11	0.19	0.17
p1.1.e	2.98	3.46	**3.57**	1.22	1.12	**1.09**	-	-	-	0.10	**0.17**	0.09
p1.1.f	2.90	3.67	**3.82**	1.24	1.09	**1.06**	-	-	-	0.11	**0.21**	0.17
p1.1.g	3.04	3.62	**3.76**	1.19	1.11	**1.06**	-	0.04	0.04	0.10	**0.33**	0.30
p1.1.h	2.84	3.55	**3.72**	1.28	1.12	**1.07**	-	-	-	0.10	**0.18**	0.13
p1.1.i	3.12	3.81	**3.96**	1.22	1.07	**1.04**	-	-	-	0.09	0.08	0.07

Table 10: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOTDOP group p2. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p2.1.a	3.53	**3.70**	3.60	1.10	**1.00**	1.05	-	0.00	-	0.33	**0.62**	0.45
p2.1.b	3.36	3.20	**3.69**	1.15	1.20	**1.01**	0.10	0.11	0.14	0.56	0.57	**0.72**
p2.1.c	3.60	3.10	**3.62**	1.05	1.24	**1.03**	0.12	**0.03**	0.11	0.68	0.47	**0.70**
p2.1.d	3.94	3.93	**3.97**	1.01	1.01	**1.00**	0.01	**0.01**	0.02	0.22	0.18	**0.25**
p2.1.e	3.21	2.28	**3.23**	1.25	1.79	**1.21**	-	-	-	0.37	0.24	0.33
p2.1.f	3.42	3.24	**3.77**	1.15	1.22	**1.04**	-	0.04	0.12	0.37	0.35	**0.56**
p2.1.g	3.51	3.39	**3.89**	1.12	1.18	**1.00**	-	0.09	0.12	0.32	**0.57**	0.45
p2.1.h	3.40	3.56	**3.72**	1.14	1.10	**1.07**	-	-	-	0.27	**0.40**	0.36
p2.1.i	3.86	3.77	**3.91**	1.04	1.06	**1.02**	**0.01**	0.05	0.01	0.15	**0.20**	0.14

Table 11: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOT-DOP group p3. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p3.1.a	3.42	3.20	**3.63**	1.12	1.20	**1.09**	-	-	0.08	0.34	0.32	0.42
p3.1.b	3.39	3.33	**3.83**	1.16	1.16	**1.05**	-	-	0.08	0.26	0.32	**0.37**
p3.1.c	3.03	3.11	**3.63**	1.19	1.20	**1.07**	-	-	-	0.22	0.22	**0.28**
p3.1.d	3.08	3.38	**3.73**	1.21	1.14	**1.07**	-	-	-	0.14	0.08	0.13
p3.1.e	3.06	3.05	**3.46**	1.23	1.24	**1.12**	-	-	-	0.13	0.06	**0.15**
p3.1.f	2.73	2.92	**3.72**	1.38	1.31	**1.09**	-	-	-	0.11	0.08	0.09
p3.1.g	2.57	3.01	**3.63**	1.35	1.20	**1.08**	-	-	-	0.04	**0.12**	0.07
p3.1.h	2.49	3.43	**3.93**	1.36	1.12	**1.02**	-	-	-	0.05	0.06	0.03
p3.1.i	2.67	3.35	**3.92**	1.29	1.13	**1.04**	-	-	-	0.03	0.04	**0.12**

Table 12: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOT-DOP group p4. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p4.1.a	2.68	3.31	**3.52**	1.27	1.13	**1.10**	-	0.04	0.04	0.20	**0.31**	0.28
p4.1.b	2.37	3.35	**3.50**	1.35	1.11	**1.10**	-	0.04	0.03	0.14	**0.30**	0.27
p4.1.c	2.00	3.18	**3.44**	1.49	1.16	**1.11**	-	0.04	0.02	0.18	**0.38**	0.24
p4.1.d	2.14	3.26	**3.42**	1.42	1.15	**1.10**	-	0.03	0.02	0.13	**0.39**	0.20
p4.1.e	1.91	3.32	**3.47**	1.52	1.16	**1.10**	-	0.04	0.02	0.14	**0.42**	0.23
p4.1.f	2.04	3.39	**3.60**	1.47	1.15	**1.08**	-	0.03	0.02	0.10	**0.38**	0.20
p4.1.g	2.00	3.41	**3.62**	1.47	1.14	**1.07**	-	0.02	0.01	0.11	**0.35**	0.15
p4.1.h	1.87	3.41	**3.62**	1.54	1.14	**1.07**	-	0.03	0.02	0.10	**0.37**	0.17
p4.1.i	1.90	3.39	**3.53**	1.51	1.15	**1.08**	-	0.04	0.02	0.09	**0.37**	0.15
p4.1.j	1.76	3.39	**3.62**	1.58	1.14	**1.07**	-	0.03	0.02	0.11	**0.32**	0.13

Table 13: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOT-DOP group p5. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p5.1.a	3.08	3.42	**3.67**	1.18	1.12	**1.06**	-	-	-	0.14	0.19	**0.23**
p5.1.b	2.92	3.25	**3.56**	1.21	1.14	**1.08**	-	0.05	-	0.15	0.22	0.15
p5.1.c	2.51	3.12	**3.45**	1.30	1.16	**1.10**	-	-	0.02	0.15	0.17	0.19

Table 14: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOTDOP group p6. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p6.1.a	2.42	3.12	**3.51**	1.35	1.19	**1.09**	-	0.04	0.02	0.21	**0.32**	0.31
p6.1.b	2.48	3.25	**3.61**	1.32	1.16	**1.07**	-	0.03	0.02	0.19	**0.35**	0.32
p6.1.c	2.19	3.16	**3.59**	1.43	1.18	**1.08**	0.04	-	**0.02**	0.19	**0.36**	0.25
p6.1.d	2.34	3.22	**3.62**	1.36	1.15	**1.07**	-	0.03	0.03	0.15	**0.25**	0.25
p6.1.e	2.11	3.23	**3.63**	1.46	1.15	**1.07**	-	-	-	0.16	0.18	0.19
p6.1.f	2.18	3.38	**3.71**	1.45	1.13	**1.06**	-	0.03	0.03	0.16	**0.22**	0.21
p6.1.g	2.34	3.30	**3.69**	1.41	1.14	**1.07**	-	-	-	0.14	0.09	0.10
p6.1.h	2.30	3.65	**4.00**	1.43	1.07	**1.00**	-	-	-	**0.13**	0.04	0.00
p6.1.i	2.82	3.86	**4.00**	1.27	1.03	**1.00**	-	-	-	**0.10**	0.01	0.00

Table 15: The mean metric values of FMOEA, MACS and MOMA on Verbeeck's MOTDOP group p7. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p7.1.a	2.36	3.09	**3.35**	1.37	1.19	**1.13**	-	0.05	0.03	0.18	**0.39**	0.33
p7.1.b	2.10	3.14	**3.33**	1.45	1.19	**1.13**	-	0.04	0.03	0.19	**0.37**	0.27
p7.1.c	2.22	3.39	**3.52**	1.42	1.15	**1.09**	-	0.03	0.03	0.16	**0.34**	0.23
p7.1.d	1.82	3.37	**3.38**	1.55	1.14	**1.12**	0.02	0.03	0.03	0.15	**0.33**	0.25
p7.1.e	1.98	**3.49**	3.40	1.47	1.11	**1.11**	-	0.02	0.03	0.11	**0.32**	0.21
p7.1.f	1.92	**3.52**	3.39	1.50	**1.10**	1.11	-	0.02	0.02	0.10	**0.34**	0.19
p7.1.g	1.85	**3.43**	3.38	1.51	**1.11**	1.11	-	0.02	0.02	0.10	**0.31**	0.19
p7.1.h	1.88	**3.53**	3.37	1.53	1.12	**1.11**	-	0.02	0.02	0.12	**0.32**	0.16
p7.1.i	1.88	**3.53**	3.43	1.55	1.11	**1.10**	-	0.02	0.03	0.09	**0.35**	0.16
p7.1.j	1.67	3.45	**3.49**	1.60	1.11	**1.09**	-	0.02	0.02	0.08	**0.32**	0.15

Table 16: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p21. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p21_t1.5	4.00	3.89	4.00	1.00	1.07	1.00	-	-	-	0.00	0.04	0.00
p21_t2.0	**3.00**	2.76	2.98	1.33	1.40	1.29	-	-	-	0.00	0.07	**0.24**
p21_t2.5	3.32	3.14	**3.68**	1.21	1.35	**1.06**	-	-	-	0.03	0.26	**0.34**
p21_t3.0	3.35	2.88	**3.60**	1.15	1.32	**1.11**	-	-	-	0.04	**0.31**	0.19
p21_t3.5	3.38	2.42	**3.69**	1.17	1.50	**1.10**	-	-	-	0.06	0.13	**0.26**
p21_t4.0	3.32	2.65	**3.71**	1.16	1.36	**1.06**	-	-	-	0.02	**0.10**	0.00

Table 17: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p32. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p32_t2.0	2.72	**3.30**	2.81	1.37	**1.13**	1.33	0.31	**0.08**	0.31	0.81	0.80	**0.82**
p32_t3.0	2.81	3.12	**3.48**	1.31	1.18	**1.09**	-	0.06	0.10	0.28	0.45	**0.64**
p32_t4.0	3.01	3.58	**3.75**	1.25	1.15	**1.09**	-	0.09	0.03	0.18	0.51	**0.53**
p32_t5.0	2.75	3.37	**3.47**	1.28	1.12	**1.09**	-	0.03	0.03	0.13	0.30	**0.32**
p32_t6.0	2.75	3.33	**3.44**	1.33	1.16	**1.15**	-	-	-	0.14	0.22	**0.27**
p32_t7.0	2.70	3.30	**3.46**	1.32	1.19	**1.14**	-	-	-	0.10	**0.29**	0.27
p32_t8.0	2.48	3.30	**3.53**	1.37	1.15	**1.11**	-	-	-	0.04	0.12	**0.16**

Table 18: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p33. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p33_t2.0	3.60	3.34	**3.89**	1.11	1.18	**1.00**	-	0.18	0.00	0.05	**0.68**	0.36
p33_t3.0	3.11	3.28	**3.66**	1.19	1.16	**1.04**	**0.08**	0.16	0.09	0.42	0.54	**0.70**
p33_t4.0	2.96	3.27	**3.70**	1.22	1.23	**1.04**	0.07	0.09	**0.05**	0.39	**0.70**	0.60
p33_t5.0	3.00	3.32	**3.66**	1.23	1.17	**1.07**	-	0.07	0.05	0.35	**0.54**	0.51
p33_t6.0	3.17	3.52	**3.77**	1.16	1.11	**1.05**	-	-	0.06	0.11	**0.36**	0.35
p33_t7.0	3.08	3.47	**3.71**	1.17	1.11	**1.06**	-	-	-	0.09	0.27	**0.28**
p33_t8.0	3.00	3.32	**3.71**	1.23	1.16	**1.06**	-	-	-	0.13	0.29	**0.31**
p33_t9.0	2.97	3.48	**3.83**	1.30	1.16	**1.10**	-	-	-	0.10	0.12	**0.25**
p33_t10.0	3.46	3.78	**4.00**	1.19	1.08	**1.00**	-	-	-	0.06	0.05	0.00

Table 19: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p64. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p64_t2.0	2.88	3.13	**3.46**	1.27	1.17	**1.13**	0.06	0.08	0.08	0.33	0.47	**0.51**
p64_t3.0	2.55	3.04	**3.39**	1.34	1.19	**1.11**	0.06	0.05	**0.03**	0.25	**0.51**	0.51
p64_t4.0	2.35	3.03	**3.45**	1.36	1.20	**1.09**	0.04	0.04	**0.03**	0.18	0.47	**0.50**
p64_t5.0	2.23	3.10	**3.50**	1.38	1.19	**1.09**	0.03	0.03	0.02	0.17	**0.48**	0.43
p64_t6.0	2.19	3.12	**3.58**	1.41	1.20	**1.08**	0.04	0.03	0.03	0.19	**0.42**	0.32
p64_t7.0	2.45	3.62	**3.97**	1.36	1.09	**1.01**	-	0.03	-	0.16	**0.19**	0.02
p64_t8.0	3.36	3.95	**4.00**	1.18	1.01	**1.00**	-	-	-	0.09	0.01	0.00

Table 20: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p66. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p66_t2.0	3.07	3.07	**3.43**	1.21	1.25	**1.12**	0.06	0.10	**0.05**	0.37	0.54	**0.54**
p66_t3.0	2.61	2.99	**3.32**	1.31	1.20	**1.12**	0.05	0.05	0.04	0.25	**0.59**	0.54
p66_t4.0	2.54	3.10	**3.30**	1.36	1.19	**1.13**	0.04	0.04	0.04	0.22	**0.64**	0.48
p66_t5.0	2.52	3.30	**3.52**	1.36	1.19	**1.10**	0.05	0.05	**0.03**	0.24	**0.70**	0.53
p66_t6.0	2.54	3.28	**3.51**	1.35	1.18	**1.09**	0.04	0.04	**0.03**	0.18	**0.65**	0.43
p66_t7.0	2.58	3.30	**3.53**	1.34	1.18	**1.09**	-	0.03	0.02	0.15	**0.61**	0.35
p66_t8.0	2.57	3.36	**3.65**	1.35	1.17	**1.08**	-	0.04	0.02	0.16	**0.58**	0.36
p66_t9.0	2.67	3.34	**3.68**	1.30	1.16	**1.07**	-	0.04	0.03	0.15	**0.49**	0.30
p66_t10.0	2.53	3.41	**3.77**	1.44	1.14	**1.06**	-	0.03	-	0.12	**0.34**	0.15
p66_t11.0	2.64	3.60	**3.90**	1.35	1.11	**1.03**	-	0.05	-	0.13	**0.24**	0.11
p66_t12.0	2.62	3.86	**4.00**	1.33	1.03	**1.00**	-	-	-	**0.14**	0.12	0.00
p66_t13.0	3.36	3.99	**4.00**	1.18	1.00	**1.00**	-	-	-	**0.11**	0.00	0.00

Table 21: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p559. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p559_t4.0	3.57	1.11	**3.77**	1.13	1.98	**1.05**	-	-	-	0.08	0.02	**0.50**
p559_t5.0	3.41	1.92	**3.71**	1.12	1.52	**1.06**	-	-	0.04	0.14	0.08	**0.20**
p559_t6.0	3.15	1.50	**3.61**	1.22	1.73	**1.09**	-	-	-	0.09	0.02	**0.19**
p559_t7.0	2.66	1.59	**3.20**	1.31	1.67	**1.17**	-	-	-	0.07	0.06	**0.16**
p559_t8.0	2.41	1.51	**3.05**	1.34	1.67	**1.18**	-	-	-	0.09	0.09	0.13
p559_t9.0	2.62	1.47	**3.56**	1.30	1.74	**1.09**	-	-	-	0.06	0.07	**0.14**
p559_t10.0	2.45	1.47	**3.23**	1.36	1.71	**1.16**	-	-	0.03	0.06	0.07	**0.15**
p559_t11.0	2.14	1.40	**3.10**	1.47	1.81	**1.19**	-	-	0.02	0.06	0.05	**0.12**
p559_t12.0	2.32	1.49	**3.41**	1.38	1.81	**1.12**	-	-	-	0.07	0.06	**0.12**
p559_t13.0	2.35	1.58	**3.44**	1.35	1.66	**1.10**	-	-	0.02	0.05	0.05	**0.10**
p559_t14.0	2.41	1.55	**3.44**	1.32	1.69	**1.10**	-	0.02	0.01	0.05	0.07	**0.09**

Table 22: The mean metric values of FMOEA, MACS and MOMA on Schilde's MOTDOP group p562. The significantly better values are marked in bold.

Instance	I_H			I_ϵ			I_S			I_R		
	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA	FMOEA	MACS	MOMA
p562_t2.0	2.78	2.91	**3.63**	1.44	1.28	**1.10**	-	-	0.12	0.38	0.23	**0.64**
p562_t3.0	3.26	2.98	**3.63**	1.13	1.22	**1.08**	-	-	-	0.13	0.24	**0.31**
p562_t4.0	2.83	2.26	**3.30**	1.24	1.40	**1.14**	-	-	-	0.08	**0.18**	0.17
p562_t5.0	2.80	2.31	**3.43**	1.24	1.38	**1.10**	-	-	0.04	0.17	0.19	**0.26**
p562_t6.0	2.60	2.51	**3.41**	1.30	1.37	**1.12**	-	-	0.04	0.17	0.28	**0.30**
p562_t7.0	2.84	2.73	**3.53**	1.22	1.31	**1.09**	-	-	0.04	0.10	**0.24**	0.23
p562_t8.0	2.40	1.83	**3.33**	1.39	1.64	**1.13**	-	0.04	-	0.10	0.18	**0.23**
p562_t9.0	2.44	2.03	**3.47**	1.40	1.61	**1.11**	-	0.07	0.04	0.10	**0.26**	0.25
p562_t10.0	2.20	2.40	**3.15**	1.50	1.40	**1.20**	-	-	0.04	0.08	0.19	**0.24**
p562_t11.0	2.12	2.24	**3.13**	1.46	1.38	**1.20**	-	-	0.04	0.09	0.19	**0.20**
p562_t12.0	1.96	2.07	**2.87**	1.56	1.51	**1.24**	-	-	0.03	0.10	0.11	**0.18**
p562_t13.0	1.96	2.08	**3.07**	1.52	1.50	**1.20**	-	-	-	0.07	0.14	**0.14**
p562_t14.0	1.99	2.18	**3.37**	1.53	1.50	**1.14**	-	-	0.03	0.05	**0.16**	0.15