Computers & Operations Research 167 (2024) 106622

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A hybrid and adaptive evolutionary approach for multitask optimization of post-disaster traveling salesman and repairman problems

Ha-Bang Ban [a], Huynh Thi Thanh Binh [a,*], Tuan Anh Do [a], Cong Dao Tran [a,c], Su Nguyen [b]

[a] School of Information and Communication Technology, Hanoi University of Science and Technology, Viet Nam
[b] RMIT University, Australia
[c] FPT Software AI Center, Viet Nam

ARTICLE INFO

Keywords:
Traveling salesman problem
Traveling repairman problem
Post-disaster scenarios
Multifactorial evolutionary algorithm
Tabu search

ABSTRACT

Traveling Salesman Problem (TSP) and Traveling Repairman Problem (TRP) have been studied due to their real-world applications. While the TSP minimizes the total time to travel to all customers, the TRP minimizes the total waiting time between a main depot and customers. Solving two problems simultaneously is shown to obtain promising results due to their high similarity in solution representation and overlap in search space. However, these two problems in the context of post-disaster have not yet been considered in line with the three following restrictions. First, the salesman cannot travel on several destroyed roads. Second, the salesman needs additional time to remove debris, which means the debris removal time is added to the travel cost. Finally, we do not consider each vertex equally because they have different characteristics depending on their population or vulnerability. Therefore, reaching a vertex with higher priority takes more benefit overall. To tackle these problems, we first formalize TSP and TRP in post-disaster scenarios as TSPPD and TRPPD, respectively, and then propose the effective metaheuristic, MFEA-TS, combining a Multifactorial Evolutionary Algorithm (MFEA) and Tabu Search (TS) to solve them simultaneously and efficiently. In the proposed MFEA-TS, the MFEA can explore the search space well by transferring knowledge between two problems, while the TS can exploit good solutions in the search space. The proposed algorithm overcomes the drawbacks of the existing MFEA algorithms due to good exploitation capacity and prevention of revisiting previous solution spaces. We further conduct extensive experiments to verify the performance of our MFEA-TS with TRPPD and TSPPD. The empirical results show that the proposed algorithm can give high-quality solutions on a range of benchmarks, thus confirming the impressive efficiency of the proposed formulations and algorithm MFEA-TS.

1. Introduction

Traveling Salesman Problem (TSP) (Applegate et al., 2006; Gavish and Graves, 1978; Pramudita et al., 2014; Ruland, 1995) and Traveling Repairman Problem (TRP) (Abeledo et al., 2013; Lucena, 1990) are combinatorial optimization problems that have many practical applications. The difference between the two problems is that the TSP is server-oriented (Abeledo et al., 2013; Lucena, 1990; Silva et al., 2012) while the TRP is customer-oriented. Each problem is interested in a different aspect in the context of logistic operations: an economic aim in terms of minimizing the travel time of a salesman and a customer-oriented objective in terms of reducing the total waiting of customers or improving the quality of service. However, two problems in the literature were studied without disaster conditions. In recent years, disasters have occurred yearly and with higher frequency and impact. Approximately 95 million individuals were affected by a lack of essential goods due to these disasters in 2019 (Anon, 2020). Because disasters destroy infrastructure, we have massive amounts of debris on roads. As a result, we face a big problem in the response stage when roads completely or partially are blocked. In the case of small debris, debris removal operations must be completed to allow vehicles to pass through. However, it is impossible to remove big debris. We propose a new formulation to address the disaster conditions and debris removal operations. At first, small debris partially blocks some roads, and the salesman needs additional time to remove them. That means the debris removal time is added to the travel cost. Conversely, a salesman cannot move on completely destroyed roads because of large debris. In this case, a salesman must choose alternative roads to move. The additional aspects make the TRP and TSP become the TRPPD and TSPPD (the TRP and TSP in post-disaster), respectively. They are also harder than the original problems because they are the TRP's and TSP's generality. In

* Corresponding author.
E-mail addresses: bangbh@soict.hust.edu.vn (H.-B. Ban), binhht@soict.hust.edu.vn (H.T.T. Binh), anhdt@soict.hust.edu.vn (T.A. Do), dao.tc212433m@sis.hust.edu.vn (C.D. Tran), su.nguyen@rmit.edu.au (S. Nguyen).

https://doi.org/10.1016/j.cor.2024.106622
Received 24 October 2023; Received in revised form 9 February 2024; Accepted 12 March 2024
Available online 3 April 2024
0305-0548/

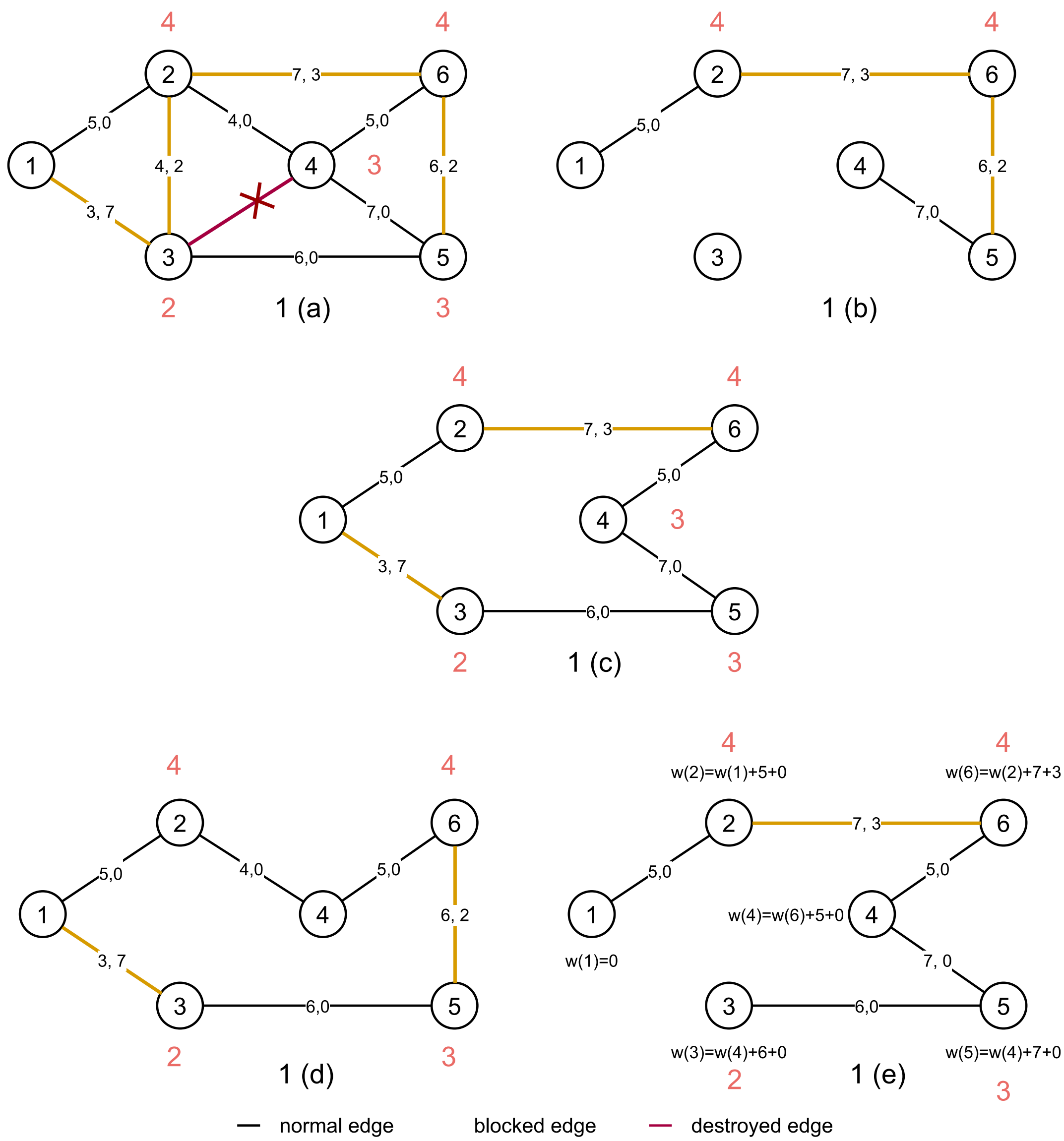


Fig. 1. The example of TSPPD and TRPPD.

addition, each location has a different important level depending on its population or vulnerability. For instance, hospitals, schools, etc should be reached firstly because it takes more benefit overall.

We consider an example of the TSPPD and TRPPD in Fig. 1. We have three types of edges: (1) yellow lines depict the blocked edges having two values, namely the cost of travel and the time required for debris removal; (2) the destroyed edges are red lines that the salesman cannot move; (3) the normal edges not affected by the disaster are demonstrated in black lines. Each vertex has a priority value indicating its important level (In Fig. 1, they are values in pink). That means a vertex with a higher priority value needs to be reached sooner than the others. We have two feasible solutions for the TSPPD and TRPPD: 1-2-4-6-5-3-1 (Fig. 1 (d)) and 1-2-6-4-5-3 (Fig. 1 (e)). The costs of these solutions are 38 $(= (5+0)+(4+0)+(5+0)+(6+2)+(6+0)+(3+7))$ and $100(= w(1)+w(2)+w(6)+w(4)+w(5)+w(3) = 0+5+15+20+27+33)$, respectively. We also have infeasible solutions such as 1-2-6-5-4-3 and 1-3-5-4-6-2 when the edge (4, 3) is destroyed completely, and the salesman violates priority constraints with the edge (4, 6), respectively (see Fig. 1 (b) 1(c)).

For NP-hard problems, two main approaches for solving the problem: (1) exact algorithm; (2) metaheuristic algorithm. The exact algorithm is often applied to small instances, while the metaheuristic is suitable for larger ones. In the traditional perspective, a metaheuristic is developed to solve only a specific problem. It is good for the problem but cannot be good for the other problems. With the advent of the MFEA framework, an algorithm can simultaneously solve various tasks by transferring good knowledge between tasks. Currently, several effective state-of-the-art MFEAs Ban and Dang-Hai (2022), Osaba et al. (2020), Yuan et al. (2016) were proposed for the class of the TSP and TRP. However, the drawbacks of these algorithms are that either they do not have an effective mechanism to exploit the good solution space explored by MFEA (Osaba et al., 2020; Yuan et al., 2016) or they can return to solution spaces explored previously (Ban and Dang-Hai, 2022). Therefore, it can get stuck into local optima in many cases. The above drawbacks are also investigated in two works (Ban and Nguyen, 2017; D'Angelo and Palmieri, 2021). To address the issues, we propose a hybrid algorithm combining MFEA and Tabu Search (TS) to address these issues. Therefore, it maintains a good combination between exploration and exploitation. The major contributions of this work are as follows:

- We develop the first formulations for both TSPPD and TRPPD and obtained experimental results by MIP-Solver suggested that the instances with up to 30 vertices can be solved exactly.

- We then propose a new metaheuristic algorithm called MFEA-TS, combining the MFEA framework and Tabu Search to solve TSPPD and TRPPD. The proposed algorithm not only enables good transferrable knowledge between tasks from the MFEA but also enhances the exploitation capacity from the TS.
- We further conduct various experiments, and the empirical results demonstrate that our proposed MFEA-TS obtains near-optimal solutions for both TRPPD and TSPPD simultaneously at a reasonable time. Moreover, our MFEA-TS also performs better than previous strong algorithms for TSP and TRP, indicating the good generalization and application of our proposed algorithm for similar problems.

The rest of this paper is organized as follows. Sections 2 and 3 present related works and the formulations of two problems, respectively. Section 4 describes the proposed algorithm. Computational evaluations are reported in Section 5. Section 6 concludes the paper.

2. Related works

The TSP and TRP are popular problems in the literature. In the exact approaches, there are many algorithms (Abeledo et al., 2013; Ban et al., 2013; Gavish and Graves, 1978) for both problems. In the metaheuristic approaches, many algorithms, including heuristics or metaheuristics (Ban and Dang-Hai, 2022; Ban and Nguyen, 2017; Feo and Resende, 1995; Salehipour et al., 2011; Silva et al., 2012), were applied for two problems with larger sizes in a short time. Generally speaking, these algorithms produce promising solution quality at a reasonable amount of time. However, they only solve each problem separately and independently. That means they cannot solve two problems simultaneously. By experiments, Salehipour et al. (2011) indicated that an algorithm to solve this problem well might not be good to solve the other. Developing an algorithm to solve two problems well simultaneously is necessary where both the waiting time of customers and disaster relief costs are prioritized.

Few recent studies have incorporated post-disaster aspects in popular combinatorial optimization problems (Ban, 2021; Berktas et al., 2016; Sahina et al., 2016; M. Çelik et al., 2015; Shuanglin et al., 2020). As we know, disaster management includes four stages: preparation, response, recovery, and reconstruction. The preparation stage aims to minimize negative situations before a disaster. The response starts immediately when the disaster occurs. The recovery focuses on restoring the disaster's transportation and infrastructure. Finally, the reconstruction's main objective is to ensure that victims' lives are normal. In disaster management, debris removal is an important step because it blocks roads from accessing disaster-affected areas. Delays in debris removal cause disruptions in providing necessary services to disaster victims. Many researchers are interested in debris removal in the reconstruction and restoration phases (Boese, 1995; M. Çelik et al., 2015; Fetter and Rakes, 2012; Pramudita et al., 2014). However, when our main aim is reaching affected areas as quickly as possible, debris removal completely takes much time. It is the main drawback of this approach. Sahina et al. (2016) first introduced debris removal to reach destroyed areas as soon as possible in the response phase. In this context, they considered the problem of finding a minimum-cost route to visiting critically affected areas by removing debris. The problem is called the Debris Removal in the Response Phase (DRR). The new variant involves extra effort, that is, debris removal time to make enough space for vehicles to pass. They proposed models and metaheuristics to minimize the time to travel critical vertices. Berktas et al. (2016) then proposed mathematical models and heuristics to improve the solution quality. M. Ajam (Ajam et al., 2019) then developed a metaheuristic to minimize the total waiting of the critical vertices during the response stage. Later, they considered the problem in the case of multiple vehicles (Akbari and Salman, 2017). The two problems in Ajam et al. (2019), Berktas et al. (2016), Sahina et al. (2016) are NP-hard problems because they are reduced to TSP and TRP when there are no blocked edges and all vertices are critical. The drawbacks of the three algorithms are that they either did not have been evaluated with larger instances or worked only with complete graphs when the triangular inequality holds. Recently, Ban (2021) have also incorporated debris removal time for the Time Dependent Traveling Salesman Problem. However, their problem requires visiting all victims. That means all vertices in the graph are critical. The experimental results show that their algorithms obtained good solutions fast.

The Multifactorial Evolutionary Algorithms (MFEA) (Bali et al., 2019; Gupta et al., 2016; Osaba et al., 2020; Yuan et al., 2016; Xu et al., 2021; Thang et al., 2021) have been known as an efficient framework for solving many optimization problems. The important advantage of the MFEA is to allow us to solve multiple problems simultaneously in which genetic transfer occurs between multitasking tasks. Recently, several variants of the MFEA have been introduced to solve directly permutation-based discrete optimization problems related to TSP and TRP. Y. Yuan (Yuan et al., 2016) developed multitasking in permutation-based optimization problems. Their experiment showed the good results of evolutionary multitasking in many-task environments. After that, Osaba et al. (2020) proposed the dMFEA-II controlling the knowledge transfer by changing the crossover probability. The results from experiments indicate their algorithm minimizes negative knowledge transfer. Though the results are promising, there is a lack of a method to exploit the good solution space in two algorithms (Osaba et al., 2020; Yuan et al., 2016). Recently, Ban and Dang-Hai (2022) have just successfully combined the MFEA with Local Search (LS) to solve the TSP and TRP at the same time. Their algorithms improve exploitation capacity better than two algorithms in Osaba et al. (2020), Yuan et al. (2016). However, two disadvantages can be found in this work: the fixed *rmp* used in Ban and Dang-Hai (2022) makes use of the positive knowledge transfer in some special cases, but it may intuitively bring negative effects in general cases (Xu et al., 2021). In addition, their algorithm (Ban and Dang-Hai, 2022) can sometimes get trapped in a cycle. That means it returns to the solution space explored previously. As a result, it gets stuck into local optima.

In overall problems, the TSP and TRP are often solved independently using metaheuristics to obtain good results in a reasonable time. In this paper, we aim to investigate two problems of TSPPD and TRPPD in the context of post-disaster and propose a new approach combining MFEA and TS to solve these simultaneously. Due to the similarity of solution space and representation of the two problems, MFEA is a suitable approach to encouraging good transfer between them. The formulations of these problems and the corresponding approach called MFEA-TS are carefully designed and presented in the following sections.

3. The formulations

Our problems in this paper have some key aspects. First, small debris partially blocks some roads, and the salesman needs additional time to remove them. Second, our problems also consider completely blocked edges. That means a salesman cannot move on these edges. This aspect makes our problems close to the TSP and TRP in the non-complete graph. Intuitively, a complete graph has many Hamilton cycles, while a non-complete graph may have no any Hamilton cycle. Checking whether a non-complete graph has a Hamilton cycle is also NP-complete. Our methods work for any incomplete graph, thus covering more realistic cases. Thirdly, the problem divides vertices into critical and non-critical vertices. That means the salesman must visit critical vertices while the salesman can bypass non-critical ones. Finally, we do not consider each vertex equally because they have different characteristics depending on their population or vulnerability. Therefore, reaching a higher priority vertex takes more benefit overall. Therefore, our problems are generality of the problems in Ajam et al. (2019), Berktas et al. (2016), Sahina et al. (2016).

To cover the above aspects, we transform an initial graph G into a new graph G' by using a simple operator as follows: For an initial graph

$G = (V, E)$ with critical and non-critical vertices, we remove edges that are destroyed by disaster complete from it. We then build a directed graph $G' = (V', E')$ on all critical vertices and a main depot in which the arc goes from a vertex with high priority to a vertex with a low one. The travel time between two vertices is calculated by the shortest path between them in the original graph G (note that the debris removal time is taken into account in the travel time). The problems are then solved on G', including only critical vertices.

3.1. The traveling repairman problem in post-disaster

The formulation is obtained from the formulation proposed by Gavish and Graves (1978) for the Minimum Latency Problem. Let V_C be a set of vertices that include the depot, 0, and the others $1, 2, \ldots, n$. Let c_{ij}, and w_{ij} be the traveling time, debris removal time of arc (i, j) while let $D = \{e_1 = (i, j), e_2, \ldots, e_l\}$ be a set of the arcs that man cannot move from vertex i to vertex j because of their priority level. We have several decision variables as follows:

$$x_{ik} = \begin{cases} 1 & \text{if vertex } i \text{ in a position } k \text{ in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ijk} = \begin{cases} 1 & \text{if vertex } i \text{ in a position } k \text{ and vertex } j \text{ in a position} \\ & k+1 \text{ in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$\min z = n \times \sum_{i \in V_C \setminus \{0\}}^{n} (c_{0i} + w_{0i}) \times x_{i1} + \sum_{k \in K \setminus \{n\}} \sum_{i \in V_C \setminus \{0\}} \sum_{j \in V_C \setminus \{0\}, i \neq j} (n-k) \times (c_{ij} + w_{ij}) \times y_{ijk}$$

Subject to:

$$\sum_{k=1}^{n} x_{ik} = 1; (i = 1, 2, \ldots, n) \tag{1}$$

$$\sum_{i=1}^{n} x_{ik} = 1; (k = 1, 2, \ldots, n) \tag{2}$$

$$\sum_{j=1}^{n} y_{ijk} = x_{ik}; (i = 1, 2, \ldots, n, j \neq i, k = 1, 2, \ldots, n-1) \tag{3}$$

$$\sum_{j=1}^{n} y_{ijk} = x_{i,k+1}; (i = 1, 2, \ldots, n, j \neq i, k = 1, 2, \ldots, n) \tag{4}$$

$$\sum_{k=0}^{n} y_{ijk} = 0; (i = 1, 2, \ldots, n, i \neq j, j = 1, 2, \ldots, n, (i, j) \in D) \tag{5}$$

$$x_{ik} \in \{0, 1\}; (i = 1, \ldots, n, k = 1, 2, \ldots, n) \tag{6}$$

$$y_{ijk} \geq 0; (i = 1, \ldots, n, j = 1, 2, \ldots, n, i \neq j, k = 1, 2, \ldots, n-1) \tag{7}$$

Constraint (1) shows that each vertex must occupy a single position. Constraint (2) indicates a single vertex captures each position. Constraint (3) shows that only one arc leaves from position k. Constraint (4) guarantees that at position $k+1$, only one arc arrives. Constraint (5) ensures the arcs in the destroyed matrix cannot be moved by man. Finally, Constraints (5) and (6) define x_{ik} as binary, and y_{ijk} are non-negative variables.

3.2. The traveling salesman problem in post-disaster

A multi-commodity flow formulation (Wong, 1980) was proven to support a strong linear relaxation to solve the TSP. In this paper, we utilize these flow formulations to solve the TSPPD. Let V_C be a set of vertices that include the depot, 0, and the others $1, 2, \ldots, n$. Let c_{ij}, and w_{ij} be the traveling time and debris removal time of arc (i, j), respectively, while let $D = \{e_1 = (i, j), e_2, \ldots, e_l\}$ be a set of the arcs that man cannot move because of priority level. We have several decision variables as follows:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is in the solution} \\ 0 & \text{otherwise} \end{cases}$$

To prevent subtour, additional variables, y_{ijk}, are included in the model. In the multi-commodity formulation, we have $n-1$ commodities with $k = 2, 3, \ldots, n$, and a non-negative decision variable y_{ijk} representing the flow on the arc $(i, j) \in E$ for the commodity k from vertex 1 to k. We have:

$\min z = \sum_{i,j=1}^{n} (c_{ij} + w_{ij}) \times x_{ij}$

Subject to:

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1; (i = 1, 2, \ldots, n) \tag{8}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1; (j = 1, 2, \ldots, n) \tag{9}$$

$$\sum_{i \in V \setminus \{0\}} y_{0jk} = 1; (k = 1, 2, \ldots, n) \tag{10}$$

$$\sum_{i \in V \setminus \{0\}} y_{i0k} = 0; (k = 1, 2, \ldots, n) \tag{11}$$

$$\sum_{i \in V} y_{ikk} = 1; (k = 1, 2, \ldots, n) \tag{12}$$

$$\sum_{i \in V} y_{kjk} = 0; (k = 1, 2, \ldots, n) \tag{13}$$

$$\sum_{i \in V} y_{ijk} - \sum_{i \in V} y_{jik} = 0; (j, k = 1, 2, \ldots, n, j \neq k) \tag{14}$$

$$\sum_{(i,j) \in D} x_{ij} = 0; \tag{15}$$

$$x_{ij} \in \{0, 1\}; (i = 0, 1, \ldots, n, j = 0, 1, 2, \ldots, n) \tag{16}$$

$$0 \leq y_{ijk} \leq x_{ij}; (i = 0, 1, \ldots, n) \tag{17}$$

Constraints (8) and (9) show that the man leaves and arrives only once at each vertex. Constraint (10) shows that one unit of each commodity flows in at vertex 0, while constraint (11) avoids any commodity out at vertex 0. Constraints (12) and (13) guarantee that one unit of commodity k flows in vertex k and it does not flow out the vertex k. Constraint (14) forces balance for all commodities at each vertex, apart from vertex 0, and for commodity k at vertex k. Constraint (15) ensures the arcs in the destroyed matrix cannot be moved by man. Finally, constraints (16) and (17) define x_{ij} are binary, and y_{ijk} are non-negative variables.

4. The proposed algorithm

4.1. The basic multifactorial evolutionary algorithm

The theory of multifactorial optimization (MFO) is proposed in Osaba et al. (2020, 2013), Xu et al. (2021). This paper briefly describes how to combine the concept of MFO in EA. We have k optimization problems that need to be minimized simultaneously. The task jth, defined T_j, with objective function $f_j : X_j \rightarrow R$, where x_j is its solution space. We need to find k solutions $\{x_1, x_2, \ldots, x_{k-1}, x_k\} = \min\{f_1(x), f_2(x), \ldots, f_{k-1}(x), f_k(x)\}$, in which x_j is a solution in X_j. Each f_j is an additional parameter impacting the evolution of a single population. Therefore, it is called the $k-$ factorial problem. For a composite problem, a method to compare individuals is important. Each solution $p_i (i \in \{1, 2, \ldots, |P|\})$ in the population P has several properties: *factorial cost*, *factorial rank*, *scalar-fitness*, and *skill-factor*. These properties allow us to rank and select them.

- Factorial cost c_j^i of the solution p_i is its fitness value for task T_j $(1 \leq j \leq k)$.

- Factorial rank r^i_j of p_i on the task T_j is its index in the list of the population ranked in ascending order with respect to c^i_j.
- Scalar-fitness ϕ_i of p_i is given by its best factorial rank overall tasks as $\phi_i = \frac{1}{\min_{j\in 1,\dots,k} r^i_j}$.
- Skill-factor ρ_i of p_i is the one task, amongst all other tasks, on which the individual is most effective, *i.e.*, $\rho_i = argmin_j\{r^i_j\}$ where $j \in \{1, 2, \dots, k\}$.

The main steps of basic MFEA are as follows. Firstly, a unified search space (USS) for different tasks is created so that the transfer of information between tasks can occur in this space. Secondly, a set of *SP* solutions (*SP* is the population's size) is initialized in the USS. Each solution is then evaluated by calculating its skill-factor. After that, an iteration begins to build offspring and assign them skill-factors. Selection ensures the skill-factor of offspring is selected randomly among those of their parents. The offspring and parent are merged to generate a new population. The evaluation for each individual is taken, and their new skill-factors are updated. The Elitist method keeps the *SP* best individuals for the next generation. Following these above steps, MFEA allows solving many tasks simultaneously using unified space and knowledge transfer between tasks.

4.2. The proposed algorithm

Fig. 2 shows the general flow of the proposed MFEA-TS, and Algorithm 1 presents the proposed MFEA-TS in detail. The first step of MFEA-TS creates a unified search space for two problems. The population is then generated in the second step. All solutions in the population must be feasible by using the Fix_Invalid algorithm. After that, an iteration begins until the termination criterion is satisfied. Parents are selected to create offspring using assortative mating and then assign skill-factor values to them. The offspring are added to the current population. The individuals of the population are re-evaluated to update their skill-factors. The best solution is picked using skill-factor and converted to each task's representation. It then is the input of the TS to find the best solution for each task. This step tries to exploit good solution space. The outputs of the TS are then converted to the unified search space. After that, they will be added to the population. The Elitist strategy keeps the *SP* solutions for the next generation. Next, the *rmp* value is updated to encourage or discourage knowledge transfer between tasks when the transfer is taking advantage or does not bring profit. After the number of generations (Ng), the best solution has not been improved, and the algorithm stops. We show that combining MFEA and TS brings a good balance between exploration and exploitation in Appendix. The details of these steps in our MFEA-TS are presented as follows.

4.2.1. Individual representation and evaluation

The permutation representation is used, in which an individual is encoded as a set of vertices as following $(v_1, v_2, \dots, v_k, \dots, v_n)$, where v_k is the kth vertex to be visited. Fig. 3 describes this encoding for two problems.

We need a fitness function to evaluate each individual in the population. The scalar-fitness is computed for each individual. Better solutions will be ones with higher scalar-fitness.

4.2.2. Population initialization

An insertion heuristic for creating an initial population is described in Algorithm 2. We consider a partial solution and define $\overline{V}$ as a set of non-visited nodes. To improve the partial tour, a vertex from $\overline{V}$ is inserted. Among all cases, we create a list of pair (v, j) (notation: L) in which v is not yet in the partial tour and a position j in the partial tour so that after the insertion, the constraints are satisfied. If the list L is not empty, we randomly select a pair in the list to insert. Otherwise, we select a pair (v, j) so that the insertion leads to the lowest increase in the cost of the solution. When the solution is feasible,

Algorithm 1: MFEA-TS

begin
- $P \leftarrow$ Initialize the population according to Algorithm 2;
- **while** *the termination criterion is not satisfied* **do**
  - Offspring population $O(t) \leftarrow \emptyset$;
  - **while** $|O(t)| < |P|$ **do**
    - Select NG individuals in the population randomly;
    - $\mathbf{p}_1, \mathbf{p}_2 \leftarrow$ Select two individuals that have the best in terms of formula (18);
    - **if** *(M and F have the same skill-factor) or (rand(1) $\leqslant$ rmp)* **then**
      - $\mathbf{c}_1, \mathbf{c}_2 \leftarrow$ Perform knowledge transfer method for M and F according to Algorithm 3;
    - **else**
      - $\mathbf{c}_1$, $\mathbf{c}_2 \leftarrow$ Mutation($\mathbf{p}_1$), Mutation($\mathbf{p}_2$) and assign skill-factor, respectively;
    - **end**
    - $\mathbf{c}_1, \mathbf{c}_2 \leftarrow$ Transform to feasible solutions according to Algorithm 4;
    - Evaluate and add $\mathbf{c}_1, \mathbf{c}_2$ to *O(t)*;
  - **end**
  - $\mathbf{o}_1, \mathbf{o}_2 \leftarrow$ Select the best individuals from *O(t)* for each task;
  - $\mathbf{o}'_1, \mathbf{o}'_2 \leftarrow$ Implement Tabu Search for each task according to Algorithm 6;
  - Convert($\mathbf{o}'_1, \mathbf{o}'_2$) to unified representation as Figure 3 and add them to *O(t)*;
  - $P \leftarrow P \cup O(t)$;
  - Update scalar-fitness and skill-factor for all individuals in P;
  - Elitism-Selection(P);
  - Update *rmp* according to Algorithm 5;
  - Update the current best solution if found;
- **end**
- **return** The best solutions;

end

it is added to the population. Conversely, a Fix_Invalid algorithm is implemented. The algorithm includes two steps: (1) finding a feasible solution from an infeasible one by removing a vertex and inserting it into a suitable position; (2) improving the obtained solution from the previous step with many neighborhoods (Lian et al., 2019). The Fix_Invalid algorithm is illustrated in Algorithm 4. This step stops until we obtain the population with *SP* feasible individuals.

This paper applies some popular neighborhoods ($N_i, i = 1, \dots, 5$) including swap-adjacent (N_1), swap (N_2), remove-insert (N_3), 2-opt (N_4), and or-opt (N_5) (Mladenovic and Hansen, 1997).

4.2.3. Selection operator

We propose a new selection for the MFEA-TS that balances scalar-fitness and diversity. The scalar-fitness effectively transfers elite genes between tasks and the good solutions are kept in each task. That means there is a large accumulation of good genes. However, diversity is important when diversity loss can make a bottleneck against the genetic information transfer. For each solution T, we consider both its scalar-fitness and diversity in a set of solutions as follows:

$$R(T) = \alpha \times (SP - RF(T) + 1) + (1 - \alpha) \times (SP - RD(T) + 1) \quad (18)$$

where $SP, \alpha \in [0, 1], RF(T)$, and $RD(T)$ are the population size, weight value, the rank of T based on scalar-fitness, and its diversity, respectively. The *RD* value of each solution is calculated as follows:

$$\overline{RD}(T) = \frac{\sum_{k=1}^{n} d(T, T_i)}{n} \quad (19)$$

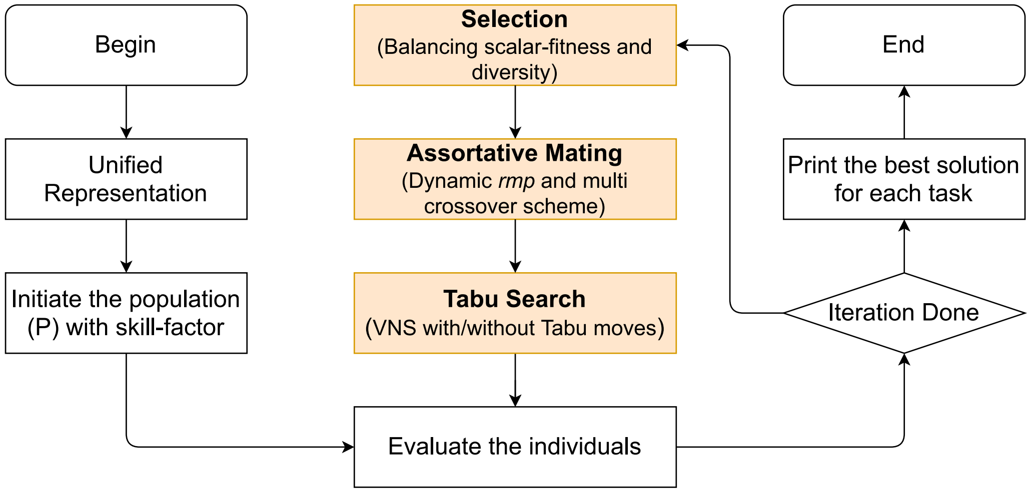


Fig. 2. Overview of the proposed algorithm.

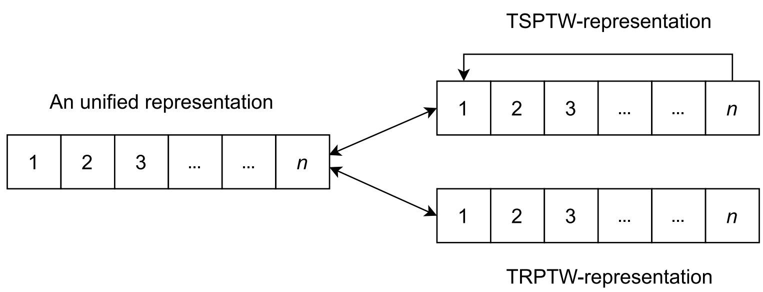


Fig. 3. The unified representation for each task.

$d(T, T_i)$ is the metric distance between T, and T_i (see in Section 3.1), and $\overline{RD}(T)$ is the average distance of T in the population. The larger $\overline{RD}(T)$ is, the higher its rank is. The larger R is, the better solution T is.

To select parents, a group of *NG* individuals is selected randomly in the same group. Then, two individuals with the largest R values are chosen to become parents.

4.2.4. Crossover operator

During the multitasking evolutionary process, the knowledge across tasks is transferred via crossover operators. There are two types of crossover: intra-task between parents of the same skill factor and inter-task with the predefined probability (*rmp*) between candidate solutions associated with different skill factors.

In Osaba et al. (2013), the crossovers are divided into three main types. We found no logical investigation showing which operator brings the best performance in the literature. In a pilot study, the following operators are selected from the study to balance solution quality and complexity time:

- Position-based crossovers (PMX, CX).
- Alternation-based without genes' repetition (EXX, EAX).
- Order-based crossovers (SC, MC).

In the beginning, a crossover is chosen at random. Along with the running, the others can be chosen, allowing repetitions. If an improvement is found, the current crossover is used. Otherwise, another crossover is selected randomly. As a result, multiple crossovers make the population diverse. Therefore, the algorithm prevents premature convergence. When the offsprings are infeasible, the Fix_Invalid algorithm repairs them to feasible ones. The offspring's skill-factors are randomly set to those of parents if inter-crossover is applied. Otherwise, they are set to those of parents respectively. The pseudocode of the crossover is given in Algorithm 4.

The *rmp* value may be dynamic or static. A larger *rmp* encourages the transfer of knowledge between tasks because the transfer is taking advantage. On the other hand, the smaller value suggests a reduction in knowledge transfer when the transfer does not bring profit. In the paper, we propose a dynamic *rmp* that controls the transfer rate between two tasks. The pseudocode of updating *rmp* value is given in Algorithm 5. If a better solution is found in any task, a current *rmp* is continued to use. It is added to the candidate value list L when it is not in L. Otherwise, the *rmp* value is updated by selecting a value from the L randomly. A Gaussian noise value is then added to create a new *rmp*.

4.2.5. Mutation operator

A mutation is used to maintain the diversity of the population. The operation randomly picks two vertices and then inverts the list of vertices between them. After the operator, two offspring are created from the parents. If the offspring is infeasible, the Fix_Invalid algorithm converts them to feasible ones. Their skill-factors are set to those of parents, respectively.

4.2.6. Tabu search

The combination of MFEA and TS utilizes transferrable knowledge between tasks from MFEA and the ability to exploit good solution spaces from TS. In this step, we convert a solution from a unified representation to a separate representation for each task, as Fig. 2. The proposed TS then applies to each task separately. The main characteristics of the TS: (1) various neighborhoods are used to exploit good solution spaces; (2) Tabu status is restarted after θ iterations; (3) only feasible solution is considered in neighborhood search; (4) a tabu movement is avoided from being applied if it is not better than the best solution. It prevents the search from getting trapped in

Algorithm 2: Insertion-based Construction

Input: v_1, V' are a depot, and the set of vertices, respectively.
Output: An initial population P.

```
begin
    P ← ∅;
    while (|P| < SP) do
        Initial individual p ← v_1;
        while (|p| < n) do
            L = List of pairs (v, j) so that Insert(p, j, bib11v)
             satisfies constraints and v ∉ the partial tour;
            /* Insert(p, j, v) aims to insert v to p at
               position j − th                                  */
            if (|L| > 0) then
                Randomly select a pair in L to insert into p;
            else
                Randomly select a vertex v (∉ in the partial tour)
                 so that p = Insert(p, j, v) has a minimal cost;
                /* if the constraints are not
                   satisfied, p is infeasible                   */
            end
        end
        if (p is infeasible) then
            p ← Convert p to feasible one according to
             Algorithm 4;
        end
        P ← P ∪ {p};
    end
    return P;
end
```

Algorithm 3: Knowledge transfer method

Input: Two parent individuals $\mathbf{p}_1$ and $\mathbf{p}_2$.
Output: Two children individuals $\mathbf{c}_1$ and $\mathbf{c}_2$.

```
begin
    if p_1 and p_2 have the same skill-factor then
        c_1, c_2 ← Crossover according to Algorithm 5;
        c_1, c_2's skill-factors are set to the skill-factors off p_1 or
         p_2, respectively;
    else
        if rand(1) ⩽ rmp then
            c_1, c_2 ← Crossover according to Algorithm 5;
            c_1, c_2's skill-factors are set to the skill-factors of p_1 or
             p_2 randomly;
        end
    end
    return two offspring c_1, c_2;
end
```

Algorithm 4: Fix_Invalid

Input: $T, N_i(i = 1, \ldots, 5)$ are the infeasible solution and a set of neighborhoods, respectively.
Output: An feasible solution T.

```
begin
    while T is infeasible do
        /* Finding a feasible solution                     */
        for i from 1 to |T| − 1 do
            for j from i + 1 to |T| do
                T' ← copy(T);
                Remove T'_i and insert it into j − th position in T';
                if (T' is feasible) then
                    T ← T';
                end
            end
        end
    end
    while The improvement is not found do
        /* Improving solution using local search */
        T ← Find a feasible solution with the lowest cost in
         neighborhoods N_i(T), i = 1, ..., 5;
    end
    return T;
end
```

Algorithm 5: Crossover

Input: $\mathbf{p}_1, \mathbf{p}_2$ are the parents, respectively.
Output: A new child $\mathbf{c}$.

```
begin
    if the current best solution is improved then
        A current crossover continues to use;
    else
        /* Choose a crossover randomly                     */
        opt ← rand(3);
        if (opt==1) then
            c ← PMX(p_1, p_2);
        end
        if (opt==2) then
            c ← EXX(p_1, p_2);
        else
            C ← SC(p_1, p_2);
        end
    end
    return c;
end
```

a cycle. After that, the solutions whose costs are less than γ (γ is a parameter) times the cost of the optimal solution are added to the promising list L. These solutions are applied to neighborhood search for all possible moves (without tabu list). If a new best solution is improved, it replaces the current best solution. The step helps the search to enhance intensification. The best solution for the TS of each task is converted into a unified representation. They are then added to the current population.

For this step, we use $N_i(i = 1, \ldots, 8)$ neighborhoods and a tabu list for each neighborhood, such as remove-insert (N_1), move-up (N_2), move-down (N_3), shift (N_4), swap-adjacent (N_5), exchange (N_6), 2-opt (N_7), and or-opt (N_8) in Mladenovic and Hansen (1997), Salehipour et al. (2011), Silva et al. (2012). The pseudocode of the TS algorithm is given in Algorithm 6.

4.2.7. Elitism operator

The operator guarantees that good solutions have a higher chance of keeping in the population. We pick 15% of the best solutions moving to the next generation, and the remaining individuals are randomly selected from P. The value 15% is suitable as shown in Reeves (1999).

5. Computational evaluations

The experiments are conducted on a personal computer with a Xeon E-2234 CPU and 16 GB of RAM. The metaheuristic was coded in C language, and mixed integer formulations were implemented using the state-of-the-art Gurobi MIP solver in Python-MIP. These parameters have been selected to find the best solution by the proposed algorithm: $SP = 50, NG = 5, \alpha = 10, \gamma = 1.2$, and $\delta = 0.1$.

Algorithm 6: Modified Tabu search

Input: T is a current solution.
Output: A new solution.

begin
  Initialize the Neighborhood List *NL*;
  /* Randomized VNS with Tabu list */
  while $NL \neq \emptyset$ **do**
    Choose a neighborhood N in *NL* at random;
    $T' \leftarrow \arg\min N(T)$;
    /* Neighborhood search */
    if ($L(T') < L(T)$ *and* T' *is feasible and not tabu)* **then**
      Update Tabu list;
      $T \leftarrow T'$ and Update *NL*;
      /* Built up promising solution */
      if T *is not less than the* γ *times current best solution* **then**
        $L \leftarrow L \cup \{T\}$;
      end
    else
      Remove N from the *NL*;
    end
  end
  /* Additional intensification */
  while $L \neq \emptyset$ **do**
    Select randomly a solution $\bar{T}$ (and remove it) from L;
    Perform Neighborhood Search without using tabu list restrictions on the solution $\bar{T}$;
  end
end
if *improvement is found* **then**
  Update the current best solution;
end
return the current best solution;

Algorithm 7: Update-rmp

Input: rmp, L, Θ are the current *rmp* value, the candidate value list, and maximum number of elements in L, respectively.
Output: updated *rmp*.

begin
  if *the better solution is found for any task* **then**
    if $|L| \geq \Theta$ **then**
      Remove a random value from L;
    end
    $L \leftarrow L \cup \{rmp\}$;
  else
    $rmp \leftarrow$ Select random a value from L;
    $rmp \leftarrow rmp + \delta \times N(0,1)$;
  end
  end
  return *rmp*;
end

Table 1
Kartal dataset.

SOE	BER	Kartal	Description
1	0%–20%	K1, …, K5	the first severity level
2	20%–40%	K6, …, K10	the second severity level
3	40%–70%	K10, …, K15	the third severity level
4	70%–100%	K16, …, K20	the fourth severity level

5.1. Dataset

We used three data sets to evaluate the exact and metaheuristic algorithm. The first one is from the Kartal district in Istanbul (Boese, 1995; Silva et al., 2012). The second is based on a simplified variant of the Istanbul road network (Ajam et al., 2019), and the third is based on the TSP and TRP benchmarks in Salehipour et al. (2011).

5.1.1. Kartal dataset

The Kartal data is based on a complete network with 45 vertices (Boese, 1995). They select subsets of size 14, 21, 22, 26, 33, 41, and 43 (subset 1) or 4, 5, 10, 14, 16, 21, 22, 26, 30, 33, 36, 38, 41, 43, and 44 (subset 2) of the critical nodes in the runs involving this network. Detailed instances about the creation of this dataset can be seen in Boese (1995), Silva et al. (2012). The dataset includes 20 instances (K1, … , K20) in which the set of critical vertices, traveling times, and the number of blocked edges are different. The travel times are the time to travel from one vertex to another. The matrix distance is symmetric, and the triangular inequality holds. To generate different scenarios in disaster situations, Silva et al. (2012) assumed four levels of earthquake severity (SOE), which vary from 1 to 4. Since the level is 1, there is a less severe earthquake. On the other hand, when the level is 4, it yields the highest severe earthquake. Table 1 illustrates the broken edge ratios (BER) according to the severe earthquake. Therefore, the debris removal times are calculated according to: (1) $w(v_i, v_j)$ = SOE $\times$ $c(v_i, v_j)$ (low debris removal time); (2) $w(v_i, v_j)$ = SOE $\times$ $c(v_i, v_j)$ + $U[0, \max_{(i,j)\in C} c(v_i, v_j)]$ (high debris removal time). Therefore, the cost to travel from v_i to v_j is the sum of $w(v_i, v_j)$ and $c(v_i, v_j)$. To assign a priority value to each vertex, they use the population parameter. That means a vertex that belongs to a crowded district has a priority value more than one with a small population. In total, we have 100 instances of different disaster situations from 7 to 15 critical vertices. In Table 1, we presents the SOE and the number of blocked edges in the instances. For instance, K1, … , K5 have 124 blocked edges when SOE is 1.

5.1.2. Instabul dataset

The Istanbul datasets are generated from the road network of 74 vertices and 179 edges for the simplified dataset, and 250 vertices and 539 edges for the southwestern dataset by Ajam et al. (2019), respectively. Detailed instances about creating these datasets can be found in Ajam et al. (2019). All edges are divided into three categories: low, medium, and high-risk edges. The probability of a blocked edge after an earthquake for the low, medium, and high-risk edges is 0.1, 0.2, and 0.3, respectively. The depot and critical vertices are selected randomly among the potential ones with equal probabilities to obtain an instance. Moreover, the blocked edges are picked randomly among all vertices with equal probability values. The travel time is the time to travel from one vertex to another. The debris time $w(v_i, v_j) = X \times c(v_i, v_j)$, where X has a uniform distribution between 100 and 300. To set a priority value for each vertex, we randomly generate a value from 1 to 4 corresponding to four levels of earthquake severity. That means a location with more earthquake severity should be supported immediately. In Ajam et al. (2019), they create 80 instances from 15 to 30 critical vertices. To evaluate the efficiency of the proposed algorithm in the larger instances, 40 instances from 50 to 100 are generated.

5.1.3. TRP dataset

The TRP dataset is found in Salehipour et al. (2011). Specifically, we select two of these sets, where each of them is composed of 50, and 100 customers, respectively. For the dataset, we evaluate the efficiency of MFEA-TS in the case of TSP and TRP.

Table 2
The algorithms' description.

Algorithm	Description
MFEA-TS	The proposed algorithm
MFEA-NR	The MFEA-TS using scalar-fitness-based selection
MFEA-No-TS	The proposed algorithm without TS
CEA	The canonical evolutionary algorithm with a single task only
TS	Tabu Search to solve each problem
MA	The GRASP+VNS algorithm of M. Ajam et al. Ajam et al. (2019)
BE	The heuristic algorithm of N. Berktas et al. Berktas et al. (2016)
SA	The heuristic algorithm of H. Sahina et al. Sahina et al. (2016)
BP	The MFEA+VNS algorithms of Ban et al. Ban and Dang-Hai (2022)
OA	The MFEA algorithm of E. Osaba (Osaba et al., 2020)
YA	The MFEA algorithm of Y. Yuan et al. Yuan et al. (2016)
TS-VNS	The Tabu with VNS algorithm of Ban et al. Ban and Nguyen (2017)
SA-ST	The VNS with single-start algorithm of A. Salehipour et al. Salehipour et al. (2011)
SA-MT	The VNS with multi-start algorithm of A. Salehipour et al. Salehipour et al. (2011)
MS	The GRASP+RNVD algorithm of M. Silva (Silva et al., 2012)

5.2. Metrics

No prior algorithms have been identified for addressing the TSPPD and TRPPD problems in existing literature. Consequently, our proposed algorithm cannot be directly benchmarked against existing solutions. With respect to our methodology, we have applied our formulations exclusively to instances characterized by small sizes, typically ranging between 7 and 30 vertices. Regarding MFEA approaches, the solutions are compared to the optimal solutions from the exact approach in the small instances, while for large instances, they are compared to the upper bounds. Moreover, we adopt the proposed algorithm to solve some close variants: In the TSP and TRP, the proposed algorithm (MFEA-TS) directly compares with the MFEA (Ban and Dang-Hai, 2022), OA (Osaba et al., 2020), and YA (Yuan et al., 2016) while in the TSPPD and TRPPD without priority constraints, it compares with the state-of-the-art metaheuristics in Ajam et al. (2019), Berktas et al. (2016), Sahina et al. (2016). More details of these algorithms are provided in Table 2.

In this paper, some metrics can be used to evaluate the efficiency of the metaheuristic algorithm as follows:

$$gap_1[\%] = \frac{Best.Sol - KBS(OPT)}{KBS(OPT)} \times 100\% \tag{20}$$

$$gap_2[\%] = \frac{Best.Sol - UB}{UB} \times 100\% \tag{21}$$

We have some notations as follows:

- *Best.Sol* is the best solution found by the proposed algorithm.
- *OPT* and *KBS* are the optimal solution and the known best solution of the previous algorithms, respectively.
- *UB* is the best solution of Insertion-based Construction.
- *Aver.Sol* is the average solution after 30 runs.
- $Time$ is the running time by second such that the proposed algorithm reaches the best solution.

The proposed algorithm is compared with the other algorithms showed in Table 1: To compare the effectiveness of the algorithms, the non-parametric statistic is selected to analyze the obtained results. There are two major steps in the comparison process:

- Statistical methods such as Friedman, Aligned Friedman, and Quade (Carrasco et al., 2020) are used to evaluate the differences among results obtained by the aforementioned algorithms.
- Once the hypothesis of equivalence of means of results obtained by algorithms in the first step is rejected, the post-hoc statistical procedures (Carrasco et al., 2020) are used to compute the concrete differences among algorithms and compare them. In this step, the control algorithm is the MFEA-TS.

5.3. Experimental scenarios

In this section, the effectiveness of the proposed formulations and the proposed algorithm MFEA-TS is evaluated in several experiments as follows:

- *Experiment 1:* analyze the effectiveness of two proposed formulations of TSPPD and TRPPD using an exact optimizer to find the optimal solutions.
- *Experiment 2:* evaluate the performance of MFEA-TS for TSPPD and TRPPD by comparing to the upper bound, optimal value, and a single-task algorithm to prove the effectiveness of the proposed multi-task algorithm.
- *Experiment 3:* conduct an ablation study to indicate the effectiveness of each new component in our MFEA-TS, including the selection method, the rmp update method, and the hybridization of MFEA and TS.
- *Experiment 4:* evaluate the effectiveness of the proposed MFEA-TS applied to similar problems, *i.e.*, TRP and TSP and TSPPD and TRPPD without priority constraints, by comparing to the solid previous algorithms to demonstrate the good generalization of our MFEA-TS applications.

5.4. Experimental results and discussions

5.4.1. Analysis of proposed formulations of TRPPD and TSPPD

In Tables 3 to 5, the first column shows the size of instances. The second and third columns show the results of Berktas et al.'s formulation (BE) (Salehipour et al., 2011) when using Gurobi and Cplex solvers, respectively. The OPT column indicates the optimal solution found, while the Time column describes the running time by second. We limit 3 h for each run. After a limited time, the formulations stop running. The diff[%] shows the difference between the optimal solutions when a priority constraint is included (p column) or not (no-p column).

Tables 3 and 4 show that the formulations for the TRPPD and TSPPD solved optimality instances with 7 and 15 vertices in less than one second. On the other hand, Berktas et al.'s formulation consumes much time to obtain the optimal solution in the instances with 7 vertices. In addition, it fails to solve exactly the instances with 15 vertices. Table 3 also shows that the optimal solutions for the TSPPD or TRPPD in the case of small blocking arcs (K1-K5) are the same. Specifically, the optimal values for K1-K5 instances (SOE = 1) for both the TSPPD and TRPPD are 106 and 509, respectively. It indicates that the optimal solutions remain unchanged when the number of blocking arcs is small. However, when the number of blocking arcs increases, the difference between them is significant. Namely, from K6 to K20 (SOE = 2, 3, 4), their optimal values are significantly different in both TSPPD and

Table 3
The results of formulations for Kartal datasets with low debris removal time.

n	7									15							
Instances	BE	TSPPD				TRPPD				TSPPD				TRPPD			
		no-p	p			no-p	p			no-p	p			no-p	p		
	CPU time (Gurobi)	OPT	OPT	diff [%]	Time	OPT	OPT	diff [%]	Time	OPT	OPT	diff [%]	Time	OPT	OPT	diff [%]	Time
K1	95.83	65	106	146.51	0.01	180	509	182.78	0.01	101	183	115.29	0.08	642	1512	135.51	0.06
K2	124.7	65	106	146.51	0.01	180	509	182.78	0.01	101	183	115.29	0.07	642	1512	135.51	0.06
K3	102.15	65	106	146.51	0.01	180	509	182.78	0.01	101	183	115.29	0.07	642	1512	135.51	0.07
K4	99.36	65	106	146.51	0.01	180	509	182.78	0.01	101	183	115.29	0.07	642	1512	135.51	0.06
k5	137.65	65	106	146.51	0.01	180	509	182.78	0.01	116	183	115.29	0.08	642	1512	135.51	0.06
average				146.51	0.01			182.78	0.01			115.29	0.074			135.51	0.062
K6	387.90	71	107	122.92	0.01	198	514	159.60	0.01	117	192	97.94	0.08	731	1573	115.18	0.06
K7	426.63	68	108	134.78	0.01	192	525	173.44	0.01	114	198	100.00	0.08	740	1644	122.16	0.07
K8	431.03	71	111	131.25	0.01	204	538	163.73	0.02	112	195	95.00	0.07	740	1600	116.22	0.06
K9	358.51	69	108	129.79	0.01	204	538	163.73	0.01	113	194	100.00	0.08	738	1621	119.65	0.07
K10	422.57	70	108	125.00	0.02	198	512	158.59	0.01	117	189	90.91	0.07	716	1554	117.04	0.07
average				128.75	0.012			163.81	0.012			96.77	0.076			118.05	0.066
K11	722.86	74	114	128.00	0.02	225	552	145.33	0.02	130	198	94.12	0.08	816	1619	98.41	0.06
K12	806.2	83	123	101.64	0.02	266	599	125.19	0.01	120	217	85.47	0.09	866	1784	106.00	0.07
K13	624.41	80	119	88.89	0.02	272	575	111.40	0.01	115	220	101.83	0.08	848	1864	119.81	0.07
K14	605.93	69	112	143.48	0.01	197	542	175.13	0.02	113	200	108.33	0.09	721	1664	130.79	0.08
K15	513.3	70	111	136.17	0.02	198	524	164.65	0.02	113	197	107.37	0.09	715	1611	125.31	0.07
average				119.64	0.018			144.34	0.016			99.42	0.086			116.07	0.07
K16	1741.48	120	111	23.33	0.02	430	682	58.60	0.02	170	294	104.17	0.1	1197	2389	99.58	0.08
K17	1495.75	95	154	105.33	0.02	382	653	70.94	0.03	171	274	85.14	0.1	1201	2140	78.18	0.09
K18	1956.3	114	145	62.92	0.03	401	678	69.08	0.03	194	264	53.49	0.1	1309	2262	72.80	0.08
K19	1717.03	104	148	87.34	0.02	401	678	69.08	0.03	161	245	71.33	0.09	1085	2053	89.22	0.08
K20	1615.06	110	141	71.95	0.03	349	737	111.17	0.02	199	273	51.67	0.09	1268	2226	75.55	0.08
average			157	70.18	0.024			75.78	0.026			73.16	0.096			83.07	0.082

Table 4
The results of formulations for Kartal datasets with high debris removal time.

n	7										15							
Instances	BE		TSPPD				TRPPD				TSPPD				TRPPD			
	CPU time (Gurobi)	CPU time (Cplex)	no-p	p			no-p	p			no-p	p			no-p	p		
			OPT	OPT	diff [%]	Time	OPT	OPT	diff [%]	Time	OPT	OPT	diff [%]	Time	OPT	OPT	diff [%]	Time
K1	77.9	70.25	66	106	140.91	0.01	185	509	175.14	0.01	107	183	110.34	0.09	655	1512	130.84	0.08
K2	79.92	94.15	65	107	148.84	0.01	180	509	182.78	0.01	103	184	109.09	0.09	666	1512	127.03	0.07
K3	170.13	88.21	65	108	145.45	0.01	186	511	174.73	0.01	106	184	109.09	0.08	654	1512	131.19	0.08
K4	180.57	60.27	65	106	146.51	0.01	180	509	182.78	0.01	104	189	110.00	0.08	677	1547	128.51	0.08
K5	173	60.57	65	106	146.51	0.01	180	509	182.78	0.01	119	187	112.50	0.07	663	1539	132.13	0.07
average	136.30	74.69			145.64	0.01			179.64	0.01			110.21	0.08			129.94	0.08
K6	381.84	476.06	71	107	122.92	0.01	198	514	159.60	0.01	117	204	104.00	0.08	752	1620	115.43	0.08
K7	336.59	314.06	72	108	116.00	0.02	208	525	152.40	0.01	114	204	101.98	0.07	759	1698	123.72	0.09
K8	340.94	332.62	74	117	129.41	0.01	208	568	173.08	0.01	116	200	100.00	0.11	740	1639	121.49	0.07
K9	423.35	741.95	71	112	128.57	0.02	208	568	173.08	0.01	113	200	98.02	0.07	778	1670	114.65	0.07
K10	384.81	430.96	70	108	125.00	0.01	198	512	158.59	0.01	119	189	90.91	0.08	716	1554	117.04	0.07
average	373.51	459.13			124.38	0.01			163.35	0.01			98.98	0.08			118.46	0.08
K11	490.67	1460.1	77	117	120.75	0.02	243	564	132.10	0.01	136	201	93.27	0.09	836	1641	96.29	0.1
K12	503.7	311.29	85	128	103.17	0.02	279	621	122.58	0.02	121	225	82.93	0.09	932	1857	99.25	0.08
K13	721.46	1246.04	84	123	80.88	0.02	287	596	107.67	0.02	115	227	102.68	0.1	861	1942	125.55	0.07
K14	654.75	379.09	69	112	143.48	0.01	197	542	175.13	0.01	113	205	113.54	0.09	721	1679	132.87	0.08
K15	709.89	245.1	70	111	136.17	0.01	198	524	164.65	0.02	113	198	106.25	0.1	718	1614	124.79	0.08
average	616.09	728.32			116.89	0.016			140.42	0.016			99.73	0.094			115.75	0.082
K16	1668.84	5874.1	140	172	50.88	0.02	554	750	35.38	0.02	179	327	106.96	0.1	1358	2598	91.31	0.1
K17	1536.69	4067.4	102	160	95.12	0.01	410	724	76.59	0.02	189	325	95.78	0.12	1293	2477	91.57	0.09
K18	2187.08	7200	137	161	42.48	0.03	488	734	50.41	0.03	209	279	53.30	0.1	1387	2422	74.62	0.09
K19	1334.81	7200	119	163	75.27	0.03	488	734	50.41	0.03	180	277	77.56	0.1	1193	2265	89.86	0.1
K20	1298.63	2286.1	131	203	89.72	0.01	441	967	119.27	0.02	228	317	53.14	0.09	1505	2554	69.70	0.1
average	1605.21	5325.52		157	70.69	0.02			66.41	0.024			77.35	0.102			83.41	0.096

TRPPD. Even with the high debris removal time in Table 4, their optimal values also show a relative difference in the case of $SOE = 1$. Therefore, a large number of blocking arcs strongly affects the optimal solutions for both two problems. In addition, we compare the optimal solutions for both problems since their priority constraints are included or not. The results in Tables 3 and 4 show that the priority constraints clearly change the optimal solutions for both problems. Specifically, the average difference between them is from 70.18% to 146.51%.

To evaluate the efficiency of our formulations, we run them with larger datasets (Simplified and Southwestern). In Tables 5 and 6, the running time increases exponentially when the number of vertices increases. For example, the formulations spend 0.33 (2.66) seconds

Table 5
The results of formulations for Simplified Istanbul datasets.

n	15				20				25				30			
Instances	TSPPD		TRPPD		TSPPD		TRPPD		TSPPD		TRPPD		TSPPD		TRPPD	
	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time
K1	60	0.07	336	6.4	36	1.71	320	4.88	126	3	1355	187.8	66	7.89	730	1045.8
K2	50	0.02	342	1.6	36	0.76	327	4.88	74	10.2	719	321.6	59	69.88	749	1021
K3	50	0.03	301	4.2	99	1.25	749	16.45	57	4.11	744	61.8	45	15.63	546	396
K4	61	0.02	551	1.47	49	0.59	829	446	104	1.92	1306	470.4	133	25.83	2089	153.6
K5	33	0.04	156	3.45	62	0.75	369	17.58	87	5.55	1108	43.09	31	28.35	287	455.4
K6	65	0.02	558	1.89	59	3.55	700	8.85	152	6.86	970	138	81	27.91	1021	966.6
K7	33	0.02	214	1.37	113	4.32	926	5.36	97	3.38	1081	140.4	131	9.92	1679	182.4
K8	72	0.02	462	2.37	82	1.26	853	13.86	110	2.37	1553	85.8	140	10.96	1678	976.8
K9	78	0.07	634	2.61	106	2.15	822	7.19	54	2.37	651	1273.2	125	11.36	2483	216
K10	81	0.02	714	1.28	37	1.4	441	5.06	101	2.54	1001	196.8	93	31.36	1291	505.2
average		0.03		2.66		24.51		53.01		4.23		291.89		23.91		591.88

Table 6
The results of formulations for Southwestern datasets.

n	15				20				25				30			
Instances	TSPPD		TRPPD		TSPPD		TRPPD		TSPPD		TRPPD		TSPPD		TRPPD	
	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time	OPT	Time
K1	21	0.04	212	3.49	13	0.85	160	38.7	15	2.92	174	76.8	66	12.23	730	1045.8
K2	10	0.02	80	3.61	17	0.78	97	9.91	15	13.19	136	232.3	10	10.93	172	687
K3	13	0.08	108	1.22	15	0.7	141	25.18	21	3.23	235	61.2	9	12.97	173	698
K4	16	0.03	128	0.97	11	1.03	103	26.92	22	4.08	306	205.2	11	17.08	157	2427
K5	12	0.02	95	1.61	23	0.79	283	10.88	9	15.16	91	247.8	13	23.7	209	985.2
K6	12	0.04	80	1.89	17	1.52	188	14.3	18	6.08	239	73.8	18	21.83	266	552
K7	15	0.08	101	1.41	20	0.63	157	8.1	18	1.55	197	63.6	13	43.25	176	307.8
K8	11	0.02	107	3.61	11	2.71	64	7.45	10	5.85	151	131.4	17	7.15	267	383.4
K9	7	0.02	58	0.98	13	0.78	133	8.06	23	4.38	323	70.8	15	6.99	261	504.6
K10	5	0.03	34	1.65	20	0.73	133	26.88	29	6.94	349	257.4	17	109.36	198	2102
average		0.04		2.04		1.05		17.64		6.34		142.03		26.55		969.28

Table 7
The results of MFEA-TS for Istanbul and Southwestern datasets.

Instance	TSPPD			TRPPD		
	n	gap_1	Time	*n*	gap_1	Time
Simplified Istanbul	15	0	0	15	0	0
	20	0	1	20	0	1
	25	0	2	25	0	2
	30	0	2.5	30	0	2.5
Southwestern	15	0	0	15	0	0
	20	0	1	20	0	1
	25	0	2	25	0	2
	30	0	2.5	30	0	2.5
Istanbul	7-high	0	0	7-high	0	0
	7-low	0	0	7-low	0	0
	15-high	0	0	15-high	0	0
	15-low	0	0	15-low	0	0

for 15 vertices, 1.77 (53.01) seconds for 20 vertices, 4.23 (291.89) seconds for 25 vertices, and 23.91 (591.88) seconds for 30 vertices in Simplified Istanbul. Similarly, the formulations spend 0.41 (2.04), 1.05 (17.64), 6.34 (142.03), and 26.55 (969.28) seconds for 15, 20, 25, and 30 vertices in Southwestern, respectively. In Tables 4 and 5, our formulation can reach the optimal solutions in all cases. Otherwise, Berktas et al.'s formulation cannot find the optimal solutions within the limited time.

We realize that the formulation of the TSPPD consumes less time than the one of the TRPPD. Solving the TSPPD exactly is easier than the TRPPD because of the non-local objective function in the TRPPD (Abeledo et al., 2013; Ban and Nguyen, 2017).

5.4.2. Analysis of the performance of MFEA-TS for TSPPD and TPRPD

Comparisons with Upper Bound and Optimal Value

In this experiment, we evaluate the improvement of the MFEA-TS compared to the optimal value and upper bound. The results can be seen in Tables 7 to 11. The statistical results are shown in Tables 12 to 13.

For small instances, the optimal solutions are found by the proposed formulations. Therefore, the efficiency of the algorithm can be evaluated exactly. Otherwise, the proposed algorithm's efficiency is relatively considered for larger instances. The experimental results in Table 7 show that the proposed MFEA-TS can find the optimal solutions reasonably for all instances with up to 30 vertices in some seconds. For larger instances in Tables 8–11, the improvement of the MFEA-TS upon the *UB* is significant when the best average values of gap_2 are 32.05% to 31.95% for the TSPPD and TRPPD, respectively. The statistical results in Tables 12 and 13 also indicate the dominance of the MFEA-TS compared to the *UB* values.

Comparisons with a single-task algorithm

In this experiment, we compare the results between single-task and multi-task algorithms to verify the performance of our multitasking approach. The results can be seen in Tables 8 to 11. The statistical results are shown in Tables 12 to 13.

The result of the CEA is obtained by running the MFEA-TS with the only task. Table 12 shows the ranking of the MFEA-TS is lower than the one of single-task, while Table 13 indicates the MFEA-TS is better than the CEA, and its result is statistically significant. The results show knowledge transfer between two tasks brings benefits.

When multitasking is run with the same number of generations as single-tasking, on average, it only consumes $\frac{1}{k}$ computational effort for each task (k is the number of tasks). Therefore, we consider the worst-case situation when the number of generations for multitasking is k times the one for single-tasking. If multitasking obtains better solutions than single-tasking in this case, we can say that multitasking obtains benefits. The result shows that multitasking obtains better solutions than single-tasking. It indicates the efficiency of knowledge transfer between tasks.

Table 8
The experimental results of MFEA+TS for the TSPPD with 50-x.

Instances	UB		MFEA-NR	MFEA-No-TS	CEA	TS	MFEA-TS		
	Best.Sol	gap2 [%]	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Aver.Sol	Time
50-1	235 981	39.5	154 053	165 596	149 780	149 780	142 753	143 989	12
50-2	254 527	39.8	165 973	178 618	159 440	162 810	153 292	154 922	12
50-3	307 200	39.7	201 297	217 357	192 497	198 705	185 198	187 143	13
50-4	294 542	39.6	190 760	209 173	183 985	190 760	177 865	181 002	12
50-5	257 662	39.8	162 099	182 342	162 099	162 099	155 183	158 566	13
50-6	308 924	39.2	204 144	220 970	196 568	196 568	187 909	192 160	12
50-7	294 258	39.5	192 955	208 913	187 365	190 464	178 012	178 986	13
50-8	306 216	39	202 690	219 642	196 240	199 396	186 844	191 385	14
50-9	313 625	38.1	210 448	227 646	197 537	197 537	194 265	201 654	13
50-10	324 638	39.5	209 962	229 277	202 282	209 962	196 478	201 178	14
50-11	246 455	37.7	165 159	180 273	161 487	164 395	153 437	156 565	13
50-12	335 069	40	215 856	234 344	207 818	215 856	201 087	206 168	14
50-13	247 717	39.7	162 230	174 830	157 271	160 374	149 420	152 678	14
50-14	279 986	39.6	183 541	197 196	177 327	178 165	169 053	174 057	14
50-15	288 503	39.3	186 268	194 890	180 144	186 268	175 207	177 846	14
50-16	257 886	40	166 836	181 384	162 411	165 030	154 796	159 135	13
50-17	272 687	40	176 673	186 684	171 214	171 214	163 691	175 282	12
50-18	252 800	39.4	165 859	178 937	160 841	163 233	153 181	156 481	14
50-19	276 117	39.5	181 177	195 807	174 511	179 097	166 917	167 378	13
50-20	305 457	39.9	194 246	212 230	183 547	194 246	183 547	194 854	14

Table 9
The experimental results of MFEA+TS for the TRPPD with 50-x.

Instances	UB		MFEA-NR	MFEA-No-TS	CEA	TS	MFEA-TS		
	Best.Sol	gap2 [%]	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Aver.Sol	Time
50-1	6 080 981	39.3	4 005 300	4 325 339	3 962 098	4 097 709	3 692 387	3 711 904	14
50-2	6 638 406	38.8	4 306 322	4 843 049	4 306 322	4 306 322	4 062 897	4 081 171	12
50-3	7 107 284	39.8	4 628 451	5 127 652	4 599 701	4 741 901	4 278 164	4 387 289	12
50-4	7 966 498	39.9	5 202 904	5 757 629	5 132 789	5 310 755	4 788 320	4 847 465	14
50-5	6 791 026	39.8	4 437 392	4 618 240	4 381 111	4 534 195	4 087 043	4 099 890	12
50-6	8 538 476	39.7	5 489 662	6 134 266	5 489 662	5 489 662	5 146 606	5 247 384	12
50-7	7 263 151	39.8	4 752 153	5 214 786	4 617 865	4 825 016	4 373 371	4 402 019	14
50-8	7 824 652	38.2	5 256 379	5 810 213	5 139 493	5 360 317	4 838 545	4 951 029	12
50-9	6 422 149	39.1	4 101 198	4 101 198	4 101 198	4 101 198	3 910 787	3 980 310	14
50-10	7 246 558	40	4 538 765	4 885 717	4 538 765	4 745 767	4 350 515	4 380 405	12
50-11	6 299 119	39.9	4 112 925	4 555 432	4 019 817	4 207 351	3 788 536	3 918 720	13
50-12	8 663 625	39.7	5 647 433	6 201 197	5 586 917	5 756 225	5 220 605	5 235 148	13
50-13	6 210 437	39.5	4 078 041	4 332 095	3 977 327	4 108 422	3 760 359	3 780 892	14
50-14	7 354 910	39.5	4 806 887	5 326 512	4 757 116	4 942 263	4 453 144	4 474 603	14
50-15	7 611 028	38.2	5 075 175	5 573 050	5 054 200	5 222 499	4 705 136	4 809 496	12
50-16	6 323 649	39.8	4 130 599	4 571 705	3 898 364	4 219 212	3 804 147	3 856 347	13
50-17	6 811 803	37.8	4 538 371	5 097 578	4 538 371	4 538 371	4 239 276	4 315 343	13
50-18	6 331 719	39.8	4 141 096	4 436 108	4 086 751	4 216 937	3 813 681	3 869 880	14
50-19	7 375 267	38.9	4 879 902	5 397 160	4 824 263	4 924 734	4 507 705	4 524 135	14
50-20	8 394 429	39.8	5 463 200	6 077 710	5 410 185	5 583 490	5 057 494	5 176 389	14

Table 10
The experimental results of MFEA+TS for the TSPPD with 100-x.

Instances	UB		MFEA-NR	MFEA-No-TS	CEA	TS	MFEA-TS		
	Best.Sol	gap2 [%]	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Aver.Sol	Time
100-1	381 273	24.9	301 215	335 792	307 712	314 958	286 356	288 351	133
100-2	367 316	23.3	295 930	314 588	297 298	297 298	281 625	285 058	140
100-3	404 285	25	318 588	324 817	324 817	324 817	303 346	307 684	139
100-4	351 578	24.7	267 239	311 109	284 435	294 107	264 844	273 628	135
100-5	379 862	24.8	299 173	333 407	307 199	313 130	285 820	292 382	141
100-6	362 385	24.9	285 964	315 848	291 652	301 376	272 196	273 211	145
100-7	423 565	24.8	334 111	374 473	342 106	351 577	318 336	319 260	117
100-8	355 035	24.7	280 606	313 433	284 367	296 501	267 209	268 802	151
100-9	367 533	25	289 387	320 327	291 350	305 852	275 652	280 512	140
100-10	373 781	24.9	294 798	330 197	301 385	310 397	280 800	283 709	119
100-11	350 967	24.8	277 162	309 216	277 162	277 162	263 923	270 073	123
100-12	417 894	24.7	328 984	365 591	336 588	349 304	314 524	320 401	127
100-13	320 723	24.7	254 104	280 130	257 449	267 007	241 527	245 390	121
100-14	370 816	24.9	286 252	325 877	298 704	308 866	278 357	279 391	118
100-15	348 077	24.8	272 468	307 985	279 985	290 715	261 915	266 841	130
100-16	373 168	24.2	294 548	332 628	302 914	314 245	282 951	284 870	144
100-17	330 501	24.7	248 756	290 033	248 756	268 266	248 756	266 113	128
100-18	356 400	24.8	280 417	283 182	283 182	283 182	267 988	276 107	131
100-19	341 690	23.7	274 149	305 380	275 517	275 517	260 714	262 321	115
100-20	344 006	25	271 225	303 007	276 743	280 538	258 089	259 496	123

Table 11
The experimental results of MFEA+TS for the TRPPD with 100-x.

Instances	UB		MFEA-NR	MFEA-No-TS	CEA	TS	MFEA-TS		
	Best.Sol	gap2 [%]	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Best.Sol	Aver.Sol	Time
100-1	19 701 162	23.2	16 611 026	17 763 267	15 893 159	16 800 523	15 123 495	15 129 263	124
100-2	19 041 426	24.8	15 050 709	15 050 709	15 050 709	15 050 709	14 320 185	14 438 380	126
100-3	22 184 417	25	18 218 171	19 398 593	17 181 430	18 448 881	16 642 691	16 687 134	132
100-4	17 677 320	24.4	14 642 421	15 698 945	14 068 931	14 848 014	13 366 215	13 419 299	143
100-5	19 445 140	23.4	16 336 279	17 512 478	15 608 839	16 539 207	14 888 412	15 361 077	123
100-6	19 043 432	24.7	15 729 728	16 614 015	14 479 296	15 927 717	14 336 787	14 376 290	115
100-7	21 197 212	23.8	17 742 144	18 961 402	16 924 755	17 914 939	16 161 999	16 200 148	149
100-8	18 040 875	24.9	14 743 768	15 906 571	14 231 592	15 042 317	13 539 849	13 891 184	142
100-9	18 395 395	24.8	15 193 776	16 277 673	14 226 667	15 193 776	13 839 118	14 056 154	134
100-10	19 475 134	24.7	16 061 881	17 182 678	15 395 142	16 259 255	14 672 566	14 677 904	117
100-11	17 645 405	25	13 418 669	15 148 845	13 418 669	13 418 669	13 234 577	14 066 141	135
100-12	21 520 395	24.8	16 854 040	19 035 996	16 854 040	17 978 792	16 190 510	16 202 579	137
100-13	18 268 534	24.8	14 899 620	16 127 986	14 449 604	14 899 620	13 733 131	13 744 283	118
100-14	18 156 978	24.8	14 914 561	16 025 364	13 652 605	15 150 436	13 652 605	14 587 526	127
100-15	17 570 417	24.7	14 446 116	15 526 818	13 878 303	14 672 223	13 223 492	13 524 377	140
100-16	19 615 829	23.4	16 397 486	17 617 861	15 792 968	16 671 890	15 017 576	15 034 498	126
100-17	16 258 585	24.9	12 971 680	14 043 818	12 204 540	12 971 680	12 204 540	12 778 748	147
100-18	18 314 303	24.2	15 172 286	15 342 467	14 561 872	15 342 467	13 876 452	13 883 402	136
100-19	18 417 920	24.9	14 378 858	14 378 858	14 378 858	14 378 858	13 839 463	15 120 058	143
100-20	18 508 123	24.6	15 232 670	16 387 122	14 681 403	15 338 070	13 947 650	13 958 039	152

Table 12
Average rankings achieved by the Friedman, Friedman Aligned, and Quade tests in both the TSPPD and TRPPD.

Algorithm	TSPPD			TRPPD		
	Friedman	Friedman Aligned	Quad	Friedman	Friedman Aligned	Quad
UB	1.00	20.5	1.0	1.0	10.5	0.99
MFEA-NR	3.5	121.0	3.29	3.02	51.7	3.01
MFEA-N-TS	2.02	60.54	2.01	2.0	30.5	1.99
CEA	3.51	121.9	3.71	1.9	71.3	4.02
MFEA-TS	4.96	178.4	4.97	4.0	88.4	4.95

Table 13
The z-values and p-values of the Friedman procedures (MFEA-TS is the control algorithm) in both the TSPPD and TRPPD.

i	Algorithm	TSPPD				TRPPD			
		z	p	Holm	Holland	z	p	Holm	Holland
4	UB	11.24	2.50E−29	0.0125	0.012	11.53	8.50E−31	0.0125	0.012
3	MFEA-No-TS	8.20	2.35E−16	0.016	0.017	8.20	2.32E−16	0.016	0.017
2	MFEA-NR	5.37	7.70E−8	0.025	0.025	4.89	9.93E−7	0.025	0.025
1	CEA	3.11	0.0012	0.05	0.050	2.14	0.03	0.05	0.050

Comparisons with TS algorithm

The proposed algorithm based on the MFEA framework allows us to solve two problems simultaneously with knowledge transfer. The TS is incorporated in the framework to improve exploitation capacity. When only TS is used without MFEA, we need to adapt it to solve each problem independently. In this case, there is no knowledge transfer between problems, which is the main focus of this paper. The experimental results are shown in Tables from 8 to 11.

Tables from 8 to 11 show MFEA-TS obtains better results than TS in both problems for most instances and the results are statistically significant. The results also show the approach based on the MFEA framework takes advantages with knowledge transfer.

5.4.3. Ablation study on MFEA-TS performance

Evaluating the efficiency of selection

In this experiment, we evaluate the ability of the selection operator in the MFEA-TS algorithm to balance knowledge transfer and diversity. The detailed results can be seen in Tables 8 and 11. The statistical results are shown in Tables 12 to 13.

In Tables from 8 to 11, the MFEA-TS obtains better solutions than the MFEA-NR regarding the average *gap* value for both problems. In Table 12, the ranking obtained by the Friedman, Friedman Aligned, and Quade tests strongly suggest the MFEA-TS algorithm has a slower ranking than the MFEA-NR. Table 13 confirms that the MFEA-TS is better than the MFEA-NR, and its result is significant.

Obviously, the proposed selection considering both scalar-fitness and diversity in selecting parents is more effective than the other. The scalar-fitness-based criterion for effectively transferring elite genes between tasks while diversity is important since it becomes a bottleneck against genetic knowledge transfer.

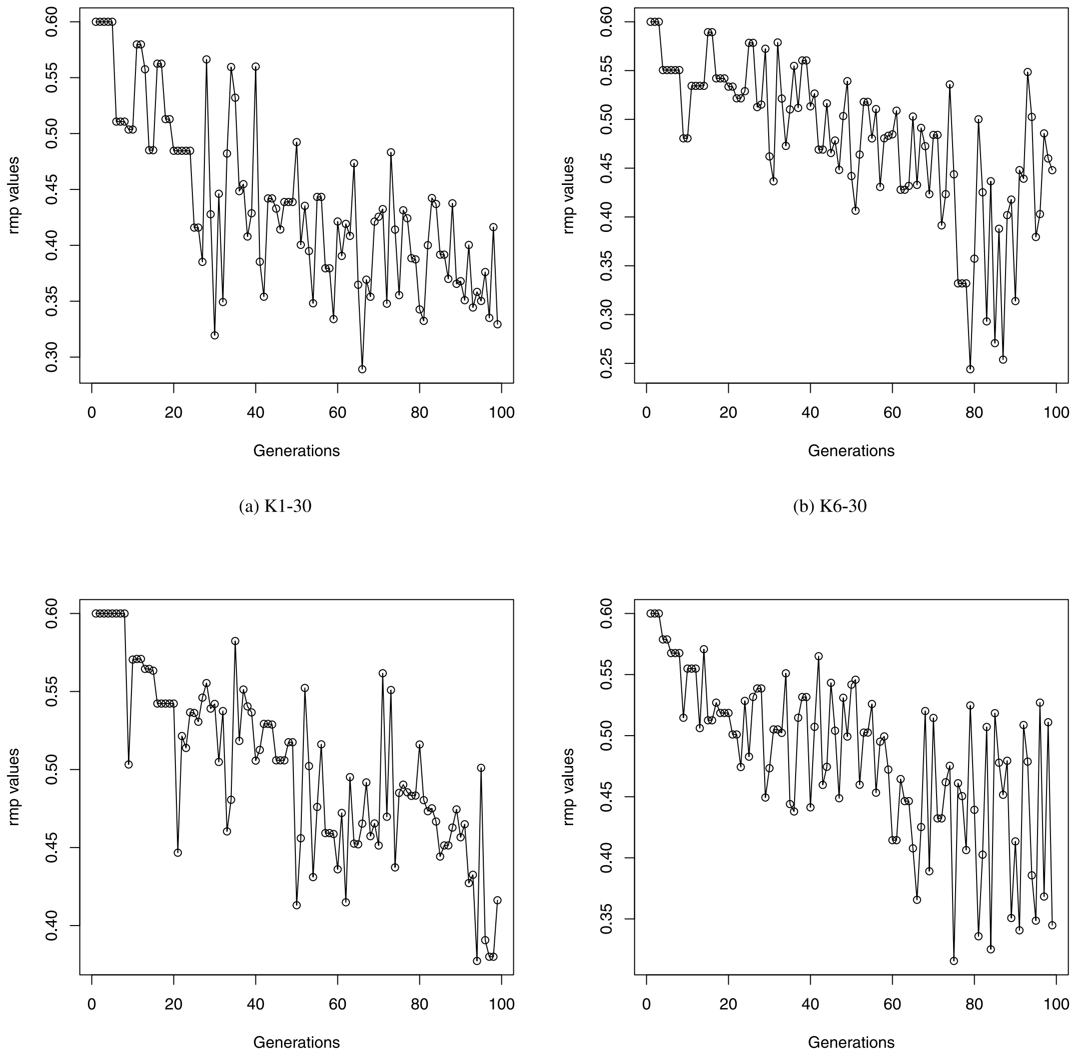
Evaluating the *rmp* values

In this experiment, the changes of the *rmp* values are evaluated. We choose some instances (K1-30, K6-30, K11-30, and K16-30) to visualize the changes of *rmp* over successive generations. The result is shown in Fig. 4. In Fig. 4, the horizontal axis is the number of generations while the vertical axis is the *rmp* value.

Fig. 4 indicates the changes of *rmp* value over 100 generations. A larger *rmp* encourages knowledge transfer between tasks because the knowledge transfer may be positive. Conversely, the smaller value shows a reduction in knowledge transfer when the transfer may be negative. It is the benefit of dynamic *rmp* in the proposed algorithm.

Evaluating the balance between exploration and exploitation

Generally speaking, algorithms get stuck into local optimum because there is a lack of balance between exploration and exploitation. Exploration helps the search to explore extension spaces on a global scale, while exploitation helps the search to focus on local space



(a) K1-30 (b) K6-30

(c) K11-30 (d) K16-30

Fig. 4. The average *rmp* values over generations.

by exploiting the information that a current solution is reached in this space. In this experiment, the balance between exploration and exploitation is considered. To study the ability to balance exploration and exploitation of the search space, we implement an experimental study on the distribution of locally optimal solutions. We choose two instances (sw-50-1, and sw-50-2) to perform one execution of our algorithm and record the distinct local optima encountered in some generations. We then plot the normalized tour's cost versus its average metric distance to all other local minima (the distance metric and its average is defined in Section 3.1). The results are illustrated in Figs. 5 to 8. The black "x" points indicate the result of the MFEA, while the red "x" points show the results of the TS. The normalized cost can be used as follows:

$$\overline{f_j} = \frac{(f_j - f_{min})}{(f_j^{max} - f_j^{min})}, \tag{22}$$

where $j = 1, 2$ is the j-task and f_j^{min}, f_j^{max} are the minimum and maximum cost values for all runs, respectively.

Figs. 5 to 8 show that the black "x" points are spread quite widely, which implies that our algorithm has the power to search over a wide region of the solution space. It is the capacity for exploration of the MFEA. On the other hand, the red "x" points are concentrated in the regions containing the good solution space. It shows the search tends to exploit the good solution spaces explored by the MFEA. It is the exploitation capacity of the TS. As a result, the algorithm maintains the right balance between exploration and exploitation.

In Tables 8 and 9, the original MFEA column is the results of the MFEA without the TS (MFEA-No-TS), while the MFEA-TS column is the results of the proposed MFEA-TS. The statistical results are shown in Tables 10 to 11. In Tables 8 and 9, for both problems, the proposed MFEA-TS obtains much better solutions than the MFEA-No-TS on average gap value. The ranking obtained in Tables 10 and 11 strongly suggests the significant differences in comparison with the MFEA-No-TS. Table 9 shows that MFEA-TS outperforms the MFEA-No-TS in the level of significance. It indicates that tabu lists prevent the search from getting trapped in a cycle. It enhances the ability to exploit good solution spaces.

5.4.4. Analysis of the generalization of our MFEA-TS for TSPPD and TRPPD without priority constraints

We adopt the MFEA-TS algorithm to solve the TSPPD and TRPPD simultaneously without priority constraints. Therefore, we can compare our results with MA (Ajam et al., 2019), BE (Berktas et al., 2016), and SA (Sahina et al., 2016) directly. The results are presented in Tables 14 and 15. The results of MA (Ajam et al., 2019) and BE (Berktas et al., 2016) are extracted from Ajam et al. (2019) while we coded the heuristic SA in Sahina et al. (2016) and ran it on the same dataset.

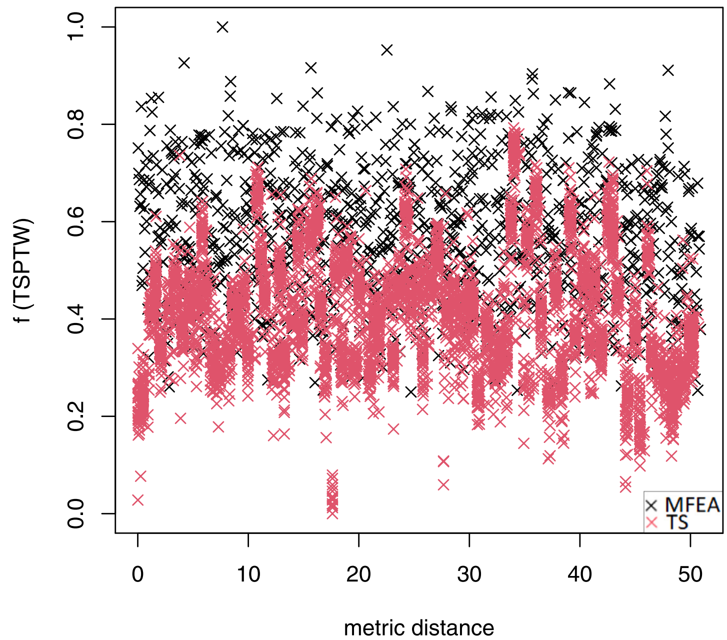


Fig. 5. The average distance to the other local optima in 50-1 instance (TSPPD).

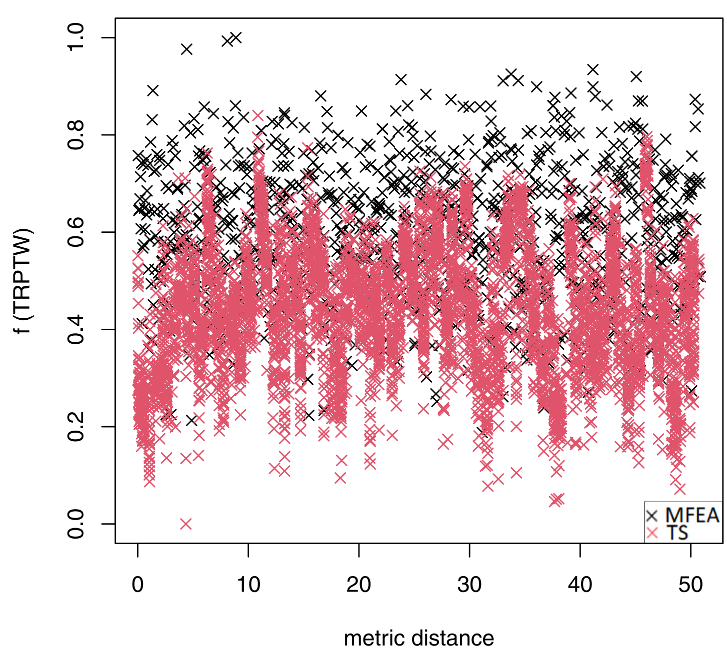


Fig. 6. The average distance to the other local optima in 50-1 instance (TRPPD).

In Table 14, the average gap_1 of SA is from 1.3% to 9.5% for low and high debris removal time, respectively. On the other hand, our average gap_1 is always 0 in all cases. Obviously, MFEA-TS obtains better results than SA (Sahina et al., 2016).

In Table 15, for the instances with 11 critical vertices, three algorithms obtain the optimal solutions in all cases. However, our average running time is much better than the others. In the instances with 15 critical vertices, our solution results still outperform the others when our algorithm obtains the optimal solutions in all cases, while the average MA, and BE's gap_1 is 4.9%, and 63.4%, respectively. Obviously, the proposed algorithm applied to the two variants well.

5.4.5. Analysis of the generalization of our MFEA-TS for TSP and TRP

In the experiment, we show the generalization of the proposed algorithm by solving two problems such as TSP and TRP though it is not developed to solve them. For TSP and TRP, many effective algorithms Ban and Dang-Hai (2022), Ban and Nguyen (2017), Osaba et al. (2020), Salehipour et al. (2011), Silva et al. (2012), Yuan et al. (2016), and Concorde tool (https://www.math.uwaterloo.ca/tsp/concorde.html) were proposed to solve in the literature. We divide them into two types: (1) algorithms Ban and Nguyen (2017), Salehipour et al. (2011), Silva et al. (2012), and Concorde tool (https://www.math.uwaterloo.ca/tsp/concorde.html) were developed to solve each problem independently and separately; (2) algorithms based on MFEA approach Ban and Dang-Hai (2022), Osaba et al. (2020), Yuan et al. (2016) to solve two problems at the same time.

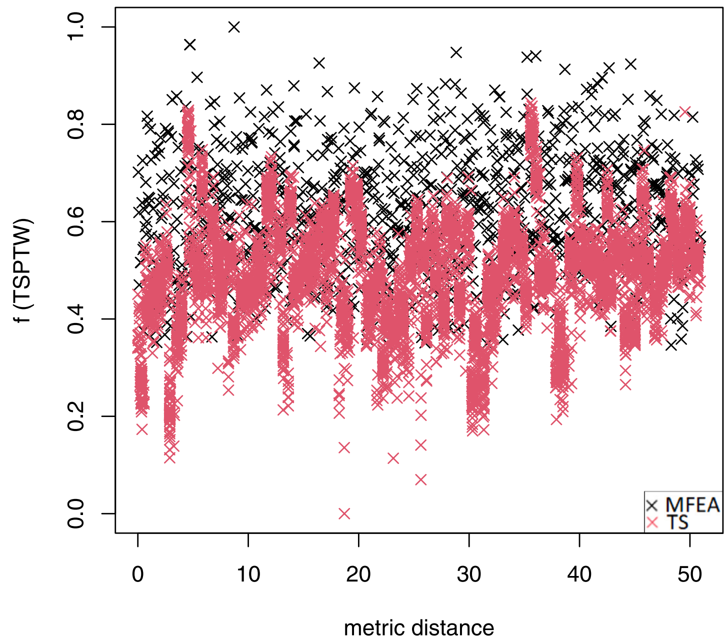


Fig. 7. The average distance to the other local optima in 50-2 instance (TSPPD).

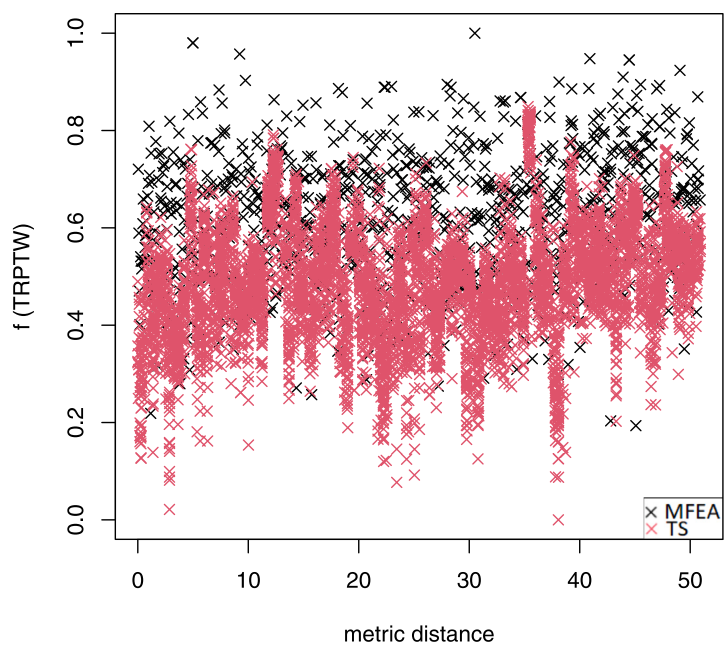


Fig. 8. The average distance to the other local optima in 50-2 instance (TRPPD).

Comparisons with algorithms based on MFEA

We adopt the MFEA-TS algorithm to solve TSP and TRP problems simultaneously. Tables 16 and 17 compare our results to BP (Ban and Dang-Hai, 2022), OA (Osaba et al., 2020), and YA (Yuan et al., 2016) in both the TSP and TRP problems. To compare fairly, we fix the maximum number of fitness function evaluations. In this work, the maximum number of fitness evaluations is $2 \times n \times 10^4$. All algorithms are run with the same maximum number of fitness evaluations.

The average gap between our result and the optimal value is below 2.59%. It indicates that our solutions are near-optimal ones. In addition, the MFEA-TS algorithm obtains the optimal solutions for the instances with up to 76 vertices. Obviously, the MFEA-TS solves well in the case of the TSP and TRP.

Table 14
Comparisons with SA for the TSPPD without priority constraints.

n	7		15	
	$gap_1[\%]$		$gap_1[\%]$	
Debris removal time	LOW	HIGH	LOW	HIGH
SA	1.3	2.17	7.3	9.5
MFEA-TS	0	0	0	0

A non-parametric test (Friedman, Aligned Friedman, and Quad test) is carried out in the group of the algorithms ((Ban and Dang-Hai, 2022), OA (Osaba et al., 2020), and YA (Yuan et al., 2016)) to check if a significant difference between them is found. Table 18 illustrates

Table 15
Comparisons with MA, and BE for the TRPPD without priority constraints.

n	11						15					
Instances	MA		BE		MFEA-TS		MA		BE		MFEA-TS	
	gap_1 [%][%]	Time	gap_1 [%][%]	Time	gap_1 [%]	Time	gap_1 [%]	Time	gap_1 [%]	Time	gap_1 [%]	Time
K1	0	0.5	0	345.3	0	0	7.8	29.7	64.2	–	0	1
K2	0	0.4	0	419.7	0	0	7.4	27.9	52	–	0	1
K3	0	0.4	0	254	0	0	6	28.3	67.8	–	0	1
K4	0	0.5	0	321	0	0	4.6	28.9	63.5	–	0	1
K5	0	0.5	0	319.3	0	0	4.6	28.4	63.4	–	0	1
K6	0	0.5	0	380.8	0	0	3.2	22.8	64.5	–	0	1
K7	0	0.4	0	1213.7	0	0	6	29.1	67.9	–	0	1
K8	0	0.4	0	683.9	0	0	4.7	24.6	49.5	–	0	1
K9	0	0.4	0	605.1	0	0	5.1	27.8	64.9	–	0	1
K10	0	0.4	0	477.3	0	0	2.6	26.4	64	–	0	1
K11	0	0.6	0	604.4	0	0	5	25.8	68.6	–	0	1
K12	0	0.4	0	677.9	0	0	3.4	26.2	67.3	–	0	1
K13	0	0.4	0	651.5	0	0	3.9	29.1	65.8	–	0	1
K14	0	0.3	0	413.5	0	0	3.5	24.3	54.1	–	0	1
K15	0	0.5	0	831.4	0	0	4.2	27.4	63.7	–	0	1
K16	0	0.5	0	1236.4	0	0	4.7	24.9	72	–	0	1
K17	0	0.5	0	573.9	0	0	5.4	24.2	67.6	–	0	1
K18	0	0.6	0	660.9	0	0	5.9	23.2	60.1	–	0	1
K19	0	0.5	0	508.2	0	0	2.1	24	67.3	–	0	1
K20	0	0.5	0	536	0	0	6.7	21.5	58.8	–	0	1
average		0.5		585.7		0	4.9	26.2	63.4			1

Table 16
The comparison with other algorithms with TRP-50-x.

Instances	OPT		YA		OA		BP		MFEA-TS		MFEA-TS (3-opt)	
	TSP	TRP	TSP	TRP	TSP	TRP	TSP	TRP	TSP	TRP	TSP	TRP
			best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol
TRP-50-1	602	12 198	641	13 253	634	13 281	610	12 330	608	12 198	608	12 198
TRP-50-2	549	11 621	583	12 958	560	12 543	560	11 710	549	11 621	549	11 621
TRP-50-3	584	12 139	596	13 482	596	13 127	592	12 312	584	12 139	584	12 139
TRP-50-4	603	13 071	666	14 131	613	15 477	610	13 575	603	13 575	603	13 575
TRP-50-5	557	12 126	579	13 377	578	14 449	557	12 657	557	12 126	557	12 126
TRP-50-6	577	12 684	602	13 807	600	13 601	588	13 070	577	12 684	577	12 684
TRP-50-7	534	11 176	563	11 984	555	12 825	547	11 793	534	11 176	534	11 176
TRP-50-8	569	12 910	629	14 043	609	13 198	572	13 198	569	12 910	569	12 910
TRP-50-9	575	13 149	631	14 687	597	13 459	576	13 459	575	13 149	575	13 149
TRP-50-10	583	12 892	604	14 104	602	13 638	590	13 267	583	12 892	583	12 892
TRP-50-11	578	12 103	607	13 878	585	12 124	585	12 124	578	12 103	578	12 103
TRP-50-12	500	10 633	521	11 985	508	11 777	604	11 305	500	10 633	500	10 633
TRP-50-13	579	12 115	615	13 885	601	13 689	587	12 559	579	12 115	579	12 115
TRP-50-14	563	13 117	612	14 276	606	14 049	571	13 431	563	13 117	563	13 117
TRP-50-15	526	11 986	526	12 546	526	12 429	526	12 429	526	11 986	526	11 986
TRP-50-16	551	12 138	577	13 211	564	12 635	551	12 417	551	12 138	551	12 138
TRP-50-17	550	12 176	601	13 622	585	13 342	564	12 475	550	12 475	550	12 475
TRP-50-18	603	13 357	629	14 750	625	14 108	603	13 683	603	13 683	603	13 683
TRP-50-19	529	11 430	595	12 609	594	12 899	539	11 659	529	11 430	529	11 430
TRP-50-20	539	11 935	585	13 603	575	12 458	539	12 107	539	11 935	539	11 935

the rankings achieved by the Friedman, Aligned Friedman, and Quade tests. The results show significant differences between the algorithms. Because the other algorithms have a larger ranking, the MFEA-TS is selected as the control algorithm. After that, we compare the control algorithm with the others by statistical tests. Table 19 shows the MFEA-TS outperforms the remaining algorithms with a level of significance $\alpha = 0.05$.

The OA and YA were developed based on the MFEA framework. The exploration ability of the MFEA is shown well (in Section 5.6, we indicate good exploration ability of the MFEA). Nevertheless, there is a lack of a mechanism to exploit the good solution space explored by the MFEA. Therefore, these algorithms cannot effectively balance exploration and exploitation. Recently, BP has successfully applied the MFEA with RVNS. The algorithm obtains better solutions than the OA and YA because of better exploration and exploitation balance. However, the search can return the explored solution space, and the BP may get stuck into local optima. The MFEA-TS not only maintains the exploration and exploitation balance by combining the MFEA and TS but also prevents the search from getting trapped into a cycle by using Tabu list. Therefore, the chance of finding better solutions is higher than the others.

Comparisons with the other algorithms

In fact, comparison between the proposed algorithm and the algorithms in Ban and Nguyen (2017), Salehipour et al. (2011), Silva et al. (2012), and Concorde tool (https://www.math.uwaterloo.ca/tsp/concorde.html) is not actually fair because these algorithms are developed to solve a specific problem but they cannot solve two problems well at the same time. In Ban and Dang-Hai (2022), Salehipour et al. (2011), they also showed that efficient algorithms for TSP may not be good for TRP. They tested two algorithms on the same instances. On average,

Table 17
The comparison with other algorithms with TRP-100-x.

Instances	OPT	KBS	YA		OA		BP		MFEA+TS		MFEA-TS (3-opt)	
	TSP	TRP	TSP	TRP	TSP	TRP	TSP	TRP	TSP	TRP	TSP	TRP
			best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol	best.sol
TRP-100-1	762	32 779	830	36 012	806	36 869	791	35 785	767	34 629	767	33 242
TRP-100-2	771	33 435	800	39 019	817	37 297	782	35 546	782	35 546	782	33 929
TRP-100-3	746	32 390	865	38 998	849	34 324	767	34 324	748	33 734	748	33 734
TRP-100-4	776	34 733	929	41 705	897	38 733	810	37 348	785	36 655	785	35 612
TRP-100-5	749	32 598	793	40 063	899	37 191	774	34 957	757	36 899	757	34 352
TRP-100-6	807	34 159	905	40 249	886	40 588	854	36 689	838	35 469	838	35 469
TRP-100-7	767	33 375	780	38 794	849	39 430	780	35 330	767	34 989	767	34 733
TRP-100-8	744	31 780	824	38 155	845	35 581	763	34 342	744	33 265	744	33 265
TRP-100-9	786	34 167	863	39 189	858	41 103	809	35 990	786	35 625	786	35 625
TRP-100-10	751	31 605	878	36 191	831	37 958	788	33 737	751	33 390	751	32 879
TRP-100-11	776	34 188	831	39 750	876	41 153	814	36 988	776	36 815	776	35 245
TRP-100-12	797	32 146	855	39 422	855	40 081	823	34 103	797	32 146	797	32 146
TRP-100-13	753	32 604	772	37 004	772	40 172	771	35 011	753	32 604	753	32 604
TRP-100-14	770	32 433	810	40 432	810	36 134	800	34 576	770	32 433	770	32 433
TRP-100-15	776	32 574	953	38 369	878	38 450	810	35 653	776	32 574	776	32 574
TRP-100-16	775	33 566	838	40 759	835	38 549	808	36 188	775	33 566	775	33 566
TRP-100-17	805	34 198	939	39 582	881	42 155	838	36 969	805	34 198	805	34 198
TRP-100-18	785	31 929	876	38 906	836	37 856	814	34 154	785	31 929	785	31 929
TRP-100-19	780	33 463	899	39 865	881	40 379	797	35 669	780	35 669	780	34 649
TRP-100-20	775	33 632	816	41 133	905	40 619	808	35 532	775	35 532	775	35 532

Table 18
Average rankings achieved by the Friedman, Friedman Aligned, and Quade tests in both the TSPPD and TRPPD.

Algorithm	TSPPD			TRPPD		
	Friedman	Aligned Friedman	Quade	Friedman	Aligned Friedman	Quade
YA	3.85	64.35	3.89	3.75	64.55	3.66
OA	2.89	50.65	2.86	3.14	54.94	3.24
MFEA	2.075	29.47	2.16	2.02	29.27	2.02
MFEA-TS	1.175	17.52	1.07	1.075	13.22	1.06

Table 19
The z-values and p-values of the Friedman procedures ((MFEA-TS is the control algorithm) in both the TSPPD and TRPPD.

i	Algorithm	TSPPD				TRPPD			
		z	p	Holm	Holland	z	p	Holm	Holland
3	YA	6.55	5.66E−11	0.016	0.016	6.55	5.66E−11	0.016	0.016
2	OA	4.22	2.38E−5	0.025	0.025	5.08	3.72E−5	0.025	0.025
1	MFEA	2.20	0.027	0.05	0.05	2.32	0.02	0.05	0.05

Table 20
Average results for algorithms in TSP and TRP.

Algorithms	TSP		TRP	
	gap_1		gap_2	
	TRP-50-x	TRP-100-x	TRP-50-x	TRP-100-x
SA+ST	–	–	−6.87%	−10.54%
SA+MT	–	–	−9.67%	−11.56%
MS	–	–	−11.01%	−13.00%
TS-VNS	–	–	−11.01%	−13.00%
MFEA-TS	–	–	−11.01%	−9.57%
MFEA-TS (using 3-opt)	0.00%	0.00%	−11.01%	−10.90%

the optimal solution for TSP using the TRP objective function is 18% worse than the optimal solution for TRP. Similarly, the optimal solution for TRP using the TSP objective function is 15% worse than the optimal solution for TSP. On the other hand, if our results are good on average for TSP and TRP at the same time, we say that the proposed algorithm for multitasking is beneficial.

In the case of TSP and TRP, to exploit better solution space, additional neighborhoods are used. We consider additional neighborhoods with larger sizes such as 3-opt, 4-opt though the complexity of time to explore these neighborhoods consumes much time in the general case. The reasons to explain why we consider their use in improving exploitation capacity are as follows: For TSP and TRP, the main operation in exploring these neighborhoods is the calculation of a neighboring solution's cost. In a straightforward way, it takes $O(n)$ time. In Ban and Nguyen (2017), Silva et al. (2012), by using the known cost of the current solution, we show that it can be done in constant time. Therefore, the fitness evaluations do not completely dominate the internal workings of the algorithm. Moreover, for TSP and TRP, all solutions are feasible and checking feasibility is not necessary.

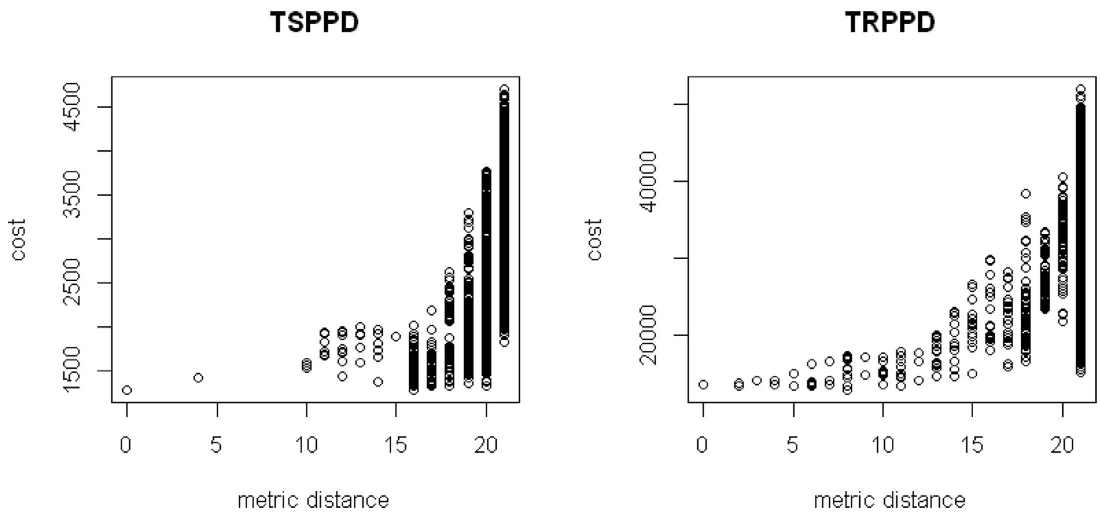


Fig. 9. The metric distance to the optimal solution in MLP-20-1 instance.

Therefore, the running time of the proposed algorithm in the case of TSP and TRP is not too time-consuming. Nevertheless, in the pilot study, 4-opt requires much time to run but it does not bring any benefit. To balance between running time and solution quality, we only use 3-opt. By using 3-opt, we use nine neighborhoods in total. To compare directly to algorithms in Ban and Nguyen (2017), Salehipour et al. (2011), Silva et al. (2012), and Concorde tool (https://www.math.uwaterloo.ca/tsp/concorde.html), another termination criterion is used. The algorithm stops if no improvement is found after 50 loops. The same termination criteria are also used in Ban and Nguyen (2017), Salehipour et al. (2011), Silva et al. (2012).

In Tables 16 and 17, MFEA-TS and MFEA-TS with 3-opt columns are our algorithms without or with using 3-opt. The average results are shown in Table 20. In TRP, UB is the result of the nearest neighbor heuristic Salehipour et al. (2011) while the optimal solutions in TSP come from Concorde Tool (https://www.math.uwaterloo.ca/tsp/concorde.html). The results show that MFEA-TS and MFEA-TS with 3-opt are comparable with the state-of-the-art algorithms for TRP while they obtain the optimal solutions for all instances in TSP. Reaching good solutions simultaneously for both two problems indicates that the proposed algorithm is beneficial.

6. Conclusions

Compared to the previous MFEA frameworks in the literature, the proposed scheme consists of new features. Firstly, we propose two formulations to solve the TSPPD and TRPPD. The formulations can solve the instances with up to 30 vertices exactly. Secondly, we propose a new selection operator that balances skill-factor and population diversity. The skill-factor effectively transfers elite genes between tasks, while diversity in the population is important when it meets a bottleneck against the information transfer. Thirdly, a multiple crossover scheme helps the proposed algorithm to maintain diversity. The combination of the MFEA with the TS has good transferrable knowledge between tasks from the MFEA and the ability to exploit good solution spaces from the TS. Extensive numerical experiments on benchmark instances show that our formulations find the optimal solutions for two problems with 30 vertices simultaneously. For larger instances, the MFEA-TS obtains better solutions than the state-of-the-art MFEA in many cases. However, the running time needs to be improved and we leave the further improvements in the future works.

CRediT authorship contribution statement

Ha-Bang Ban: Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Huynh Thi Thanh Binh:** Funding acquisition, Methodology, Project administration, Software, Supervision, Writing – original draft, Writing – review & editing. **Tuan Anh Do:** Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Cong Dao Tran:** Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Su Nguyen:** Methodology, Software, Writing – original draft, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgments

This research was funded by Vingroup Innovation Foundation (VINIF), Vietnamunder project code VINIF.2022.DA00183.

Appendix. The global search structure

In this paper, we investigate the global structure of solution space by selecting a representative instance (such as MLP-20-1 instance (Salehipour et al., 2011)) and applying Variable Neighborhood Search (VNS) on the instance. We give a definition of a measure of distance between two tours T_1 and T_2 of the problem. Naturally, the distance is defined as the minimum number of transformations from T_1 to T_2, denoted $d(T_1, T_2)$. Since there is no polynomial algorithm to compute $d(T_1, T_2)$, we define $d(T_1, T_2)$ to be n minus the number of vertices with the same position in both T_1 and T_2 (Boese, 1995).

The result of the global structure investigation is shown in Fig. 9. In this figure, the x-axis is the evaluation of the local optimum, while the y-axis is the distance from the global optimum as measured by the metric distance. The neighborhoods seem to have a big-valley structure in which the evaluation of solutions is positively correlated to the metric distances. The big valley structure often has a big valley in which local optima are spread and surround the global optimum. In Fig. 10, we also realize that the search can return the previous solution spaces explored before (cycle issue). The global structure investigation and cycle issue suggest that a good balance of exploration and exploitation, e.g. by combining the TS and MFEA, is needed. The MFEA explores extensive local optima, while the TS is attracted to the big valley area by not only exploiting good solution spaces but also preventing the search from getting trapped into cycles.

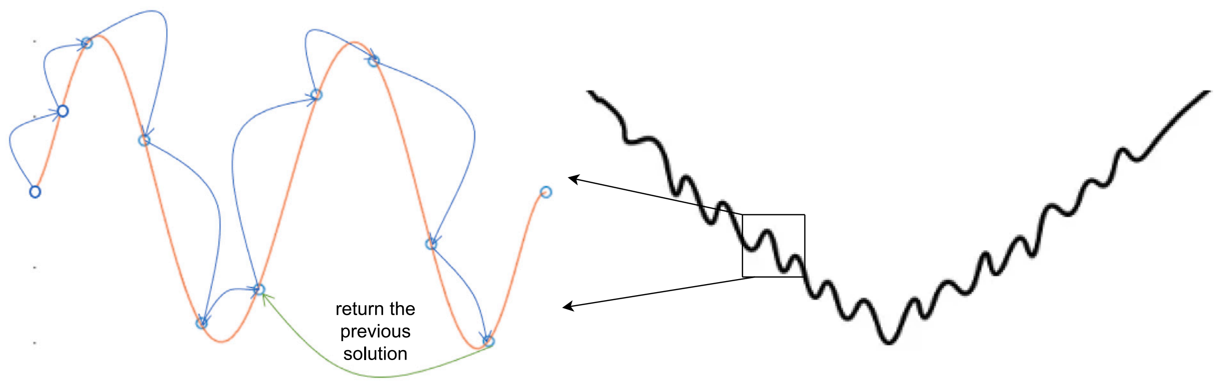


Fig. 10. The global search structure and cycle issue.

References

Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E., 2013. The time-dependent traveling salesman problem: polyhedra and algorithm. J. Math. Program. Comput. 5, 27–55.

Ajam, M., Akbari, V., Sibel Salman, F., 2019. Minimizing latency in post-disaster road clearance operations. Eur. J. Oper. Res. 277 (3), 1098–1112.

Akbari, V., Salman, F.S., 2017. Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. Eur. J. Oper. Res. 257 (2), 625–640.

Anon, 2020. Disaster Year in Review 2019, Cred Crunch, 2020 pp. 1–2. (58) Centre For Research On The Epidemiology Of Disasters (Cred) https://cred.be/sites/default/files/CC58.pdf.

Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., 2006. The Traveling Salesman Problem: A Computational Study. Princeton University Press.

Bali, K.K., Ong, Y.S., Gupta, A., Tan, P.S., 2019. Multifactorial evolutionary algorithm with online transfer parameter estimation: Mfea-II. J. IEEE Trans. Evol. Comput. 24 (1), 69–83.

Ban, Ha-Bang, 2021. Applying metaheuristic for time-dependent traveling salesman problem in postdisaster. Int. J. Comput. Intell. Syst. 14 (1), 1087–1107.

Ban, Ha-Bang, Dang-Hai, Pham, 2022. Multifactorial evolutionary algorithm for simultaneous solution of TSP and TRP. J. CAI 40 (6), 1370–1397.

Ban, Ha-Bang, Nguyen, Duc-Nghia, 2017. A meta-heuristic algorithm combining between tabu and variable neighborhood search for the minimum latency problem. J. Fundam. Inform. 156 (1), 21–41.

Ban, Ha-Bang, Nguyen, K., Ngo, M.C., Nguyen, D.N., 2013. An efficient exact algorithm for minimum latency problem. J. Inf. Prog. 10, 167–174.

Berktas, N., Kara, B.Y., Karaşan, O.E., 2016. Solution methodologies for debris removal in disaster response. EURO J. Comput. Optim. 4, 403–445.

Boese, K., 1995. Cost Versus Distance In the Traveling Salesman Problem.

Carrasco, J., Garcia, S., Rueda, S., Herrera, F., 2020. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. J. Swarm Evol. Comput. 100665.

D'Angelo, G., Palmieri, F., 2021. GGA: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems. J. Inf. Sci. 547, 136–162.

Feo, T.A., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. J. Glob. Opt. 109–133.

Fetter, G., Rakes, T., 2012. Incorporating recycling into post-disaster debris disposal. Soc. Econ. Plan. Sci. 46 (1), 14–22.

Gavish, B., Graves, S., 1978. The Traveling Salesman Problem and Related Problems, Working Paper GR-078-78. Operations Research Center, Massachusetts Institute of Technology.

Gupta, A., Ong, Y.S., Feng, L., 2016. Multifactorial evolution: toward evolutionary multitasking. J. IEEE Trans. Evol. Comput. 20 (3), 343–357.

Lian, Y.C., Huang, Z.X., Zhou, Y.R., Chen, Z.F., 2019. Improve theoretical upper bound of Jumpk function by evolutionary multitasking. Proc. HPCCT 22–24, 44–50.

Lucena, A., 1990. Time-dependent traveling salesman problem - the deliveryman case. J. Netw. 20, 753–763.

M. Çelik, M., Ergun, O., Keskinocak, P., 2015. The post-disaster debris clearance problem under incomplete information. Oper. Res. 63 (1), 65–85.

Mladenovic, N., Hansen, P., 1997. Variable neighborhood search. J. Comput. Oper. Res. 24 (11), 1097–1100.

Osaba, E., Martinez, A.D., Galvez, A., Iglesias, A., Del Ser, J., 2020. dMFEA-II: An adaptive multifactorial evolutionary algorithm for permutation-based discrete optimization problems. Proc. GECCO 1690–1696.

Osaba, E., Onieva, E., Carballedo, R., Diaz, F., Perallos, A., 2013. An adaptive multi-crossover population algorithm for solving routing problems. In: Nature Inspired Cooperative Strategies for Optimization, Vol. 512, G. Terrazas and F. Otero and A. Masegosa NICSO 2013, Springer, Cham, Switzerland, pp. 113–125.

Pramudita, A., Taniguchi, E., Qureshi, A.G., 2014. Location, and routing problems of debris collection operation after disasters with a realistic case study. Proc. Soc. Behav. 445–458.

Reeves, C.R., 1999. Landscapes, operators and heuristic search. Ann. Oper. Res. 86, 473–490.

Ruland, K., 1995. Polyhedral Solution to the Pickup and Delivery Problem (Ph.D. thesis). Washington University, Saint Louis, MO.

Sahina, H., Karab, B.Y., Karasanb, O.E., 2016. Debris removal during disaster response: a case for Turkey. J. Socio-Econ. Plan. Sci. 53, 49–59.

Salehipour, A., Sorensen, K., Goos, P., Braysy, O., 2011. Efficient grasp+vnd and grasp+vns meta-heuristics for the traveling repairman problem. J. Oper. Res. 9, 189–209.

Shuanglin, L., Zujun, Ma, Kok, T.L., 2020. A new model for road network repair after natural disasters: Integrating logistics support scheduling with repair crew scheduling and routing activities. J. Comput. Ind. Eng. 145, 106506.

Silva, M., Subramanian, A., Vidal, T., Ochi, L., 2012. A simple and effective metaheuristic for the minimum latency problem. J. Oper. Res. 221, 513–520.

Thang, T.B., Dao, T.C., Long, N.H., Binh, H.T.T., Parameter adaptation in multifactorial evolutionary algorithm for many-task optimization. J. Memet. Comput. 13 (4), 433–446.

Thang, T.B., Long, N.B., Hoang, N.V., Binh, H.T.T., 2021. Adaptive knowledge transfer in multifactorial evolutionary algorithm for the clustered minimum routing cost problem. J. Asoc. 105 (3), 1–10.

Wong, R., 1980. Integer programming formulations of the traveling salesman problem. Proc. Circuits Comput. 149.

Xu, Q., Wang, N., Wang, L., Li, W., Sun, Q., 2021. Multi-task optimization and multi-task evolutionary computation in the past five years: A brief review. J. Math. 9 (864), 1–44.

Yuan, Y., Ong, Y.S., Gupta, A., Tan, P.S., Xu, H., 2016. Evolutionary multitasking in permutation-based combinatorial optimization problems: realization with tsp, qap, lop, and jsp. Proc. TENCON 3157–3164.